

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

MODELAGEM DA INTERAÇÃO DO USUÁRIO COM O SISTEMA EM
MÉTODOS ÁGEIS

Cecilia Giuffra Palomino

Cecilia Giuffra Palomino

MODELAGEM DA INTERAÇÃO DO USUÁRIO COM O SISTEMA EM
MÉTODOS ÁGEIS

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como
parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação

Orientadora:
Prof^ª. Dr^ª. Patrícia Vilain

Florianópolis – SC
03/10/2009

MODELAGEM DA INTERAÇÃO DO USUÁRIO COM O SISTEMA EM MÉTODOS ÁGEIS

Cecilia Giuffra Palomino

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciências da Computação e aprovado em sua forma final pelo Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

Data de aprovação: 13 de Outubro de 2009.

Prof. Dr. Luís Fernando Friedrich.
Universidade Federal de Santa Catarina
Coordenador do Curso

Banca Examinadora

Prof^ª. Dr^ª. Patrícia Vilain
Universidade Federal de Santa Catarina
Orientadora

Prof. Dr. Vitório Bruno Mazzola
Universidade Federal de Santa Catarina

Prof. Dr. Ricardo Pereira e Silva
Universidade Federal de Santa Catarina

Ao meu pai, minha mãe, meu irmão, meu avô,
minha avó e minha família, que é o mais
importante para mim.

AGRADECIMENTOS

Ao meu pai, por acreditar em mim e me apoiar em tudo, pelas palavras de ânimo em todo momento, por estar sempre aí. Ao meu avô e minha avó, por me darem a força para continuar, e por aceitarem o fato de eu estar longe. À toda minha família, pelo apoio constante e pela confiança.

Ao Alex, por me ajudar tanto e sempre, pelas longas conversas, os trabalhos, as madrugadas e os dias que vimos o sol nascer, sentados nos nossos computadores. Pela solidariedade.

Ao Edu, por sua incessante vontade de me fazer aprender coisas novas, pelas horas que eu falei sozinha no MSN, e pelas vezes que ele me respondeu. Pela companhia nos momentos difíceis.

À ambos, por essa grande amizade que nasceu no primeiro dia de aula.

Ao Glauco, por todo aquele apoio no final, pela compreensão, por acreditar em mim e na minha capacidade.

Aos meus amigos em geral, pelas vezes que precisei daquela palavra de conforto.

À minha orientadora, a Prof^a. Patrícia Vilain, pela dedicação, paciência e constantes revisões.

À Universidade Federal de Santa Catarina, pela oportunidade.

"Noventa por cento do sucesso se
baseia simplesmente em insistir."

Woody Allen

SUMÁRIO

LISTA DE FIGURAS	9
RESUMO	11
ABSTRACT	12
1. INTRODUÇÃO	13
1.1. Objetivo Geral	14
1.2. Objetivos Específicos	14
1.3. Justificativa do Trabalho	14
1.4. Estrutura do Trabalho	14
2. MÉTODOS ÁGEIS	16
2.1. Extreme Programming – XP	17
2.1.1. <i>Processo do XP</i>	17
2.1.2. <i>Valores do XP</i>	19
2.1.3. <i>Princípios do XP</i>	20
2.1.4. <i>Práticas do XP</i>	20
2.1.5. <i>Papéis do XP</i>	23
2.2. Scrum	24
2.2.1. <i>Processo do Scrum</i>	24
2.2.2. <i>Práticas do Scrum</i>	26
2.2.3. <i>Equipe do Scrum</i>	28
2.3. Feature Driven Development – FDD	28
2.3.1. <i>Processo do FDD</i>	30
2.3.2. <i>Práticas do FDD</i>	32
2.3.3. <i>Equipe do FDD</i>	34
2.4. Dynamic System Development Method – DSDM	36
2.4.1. <i>Processo do DSDM</i>	37
2.4.2. <i>Princípios do DSDM</i>	40
2.4.3. <i>Práticas do DSDM</i>	41
2.4.4. <i>Equipe do DSDM</i>	42
3. DIAGRAMA DE INTERAÇÃO DO USUÁRIO – UID	44
3.1. Definição	44
3.2. Características	44

3.3. Notação	44
3.3.1. <i>Item de dado</i>	45
3.3.2. <i>Estrutura</i>	45
3.3.3. <i>Conjunto</i>	45
3.3.4. <i>Dado opcional</i>	46
3.3.5. <i>Entrada do usuário</i>	46
3.3.6. <i>Entrada do usuário enumerada</i>	46
3.3.7. <i>Saída do sistema</i>	47
3.3.8. <i>Texto</i>	47
3.3.9. <i>Estado da iteração</i>	47
3.3.10. <i>Estado inicial da iteração</i>	47
3.3.11. <i>Estados alternativos da iteração</i>	47
3.3.12. <i>Sub-estados de um estado da iteração</i>	47
3.3.13. <i>Chamada de outro UID</i>	48
3.3.14. <i>Chamada a partir de outro UID</i>	48
3.3.15. <i>Transição do estado da iteração</i>	48
3.3.16. <i>Pré e pós condições</i>	48
3.3.17. <i>Notas textuais</i>	49
3.4. Relacionamentos entre UIDs	49
3.4.1. <i>Relacionamento de inclusão (inclui)</i>	49
3.4.2. <i>Relacionamento de extensão (estende)</i>	49
3.4.3. <i>Relacionamento de precedência (precede)</i>	49
3.5. UIDs no processo de software	49
4. INCLUSÃO DE UIDs	51
4.1. Inclusão dos UIDs no XP	51
4.2. Inclusão dos UIDs no SCRUM	52
4.3. Inclusão dos UIDs no FDD	53
4.4. Inclusão dos UIDs no DSDM	54
5. ESTUDO DE CASO	56
6. CONCLUSÕES	91
7. REFERÊNCIAS.....	94
8. ANEXOS.....	97

LISTA DE FIGURAS

Figura 1. Processo do XP	18
Figura 2. Fases do <i>Scrum</i>	26
Figura 3. Exemplo de <i>Product Backlog</i>	27
Figura 4. Fases do Processo FDD	30
Figura 5. Processo de desenvolvimento do DSDM	37
Figura 6. A equipe do projeto do DSDM	43
Figura 7. Processo do XP, com inclusão de UIDs	52
Figura 8. Fases do <i>Scrum</i> , com inclusão de UIDs.....	53
Figura 9. Fases do Processo FDD, com inclusão de UIDs.....	54
Figura 10. Processo de desenvolvimento do DSDM, com inclusão de UIDs.....	55
Figura 11. Casos de Uso Sistemas de Venda Fábrica de Café.....	58
Figura 12. Diagrama de classes inicial.....	59
Figura 13. Diagrama de Seqüência – Cadastro de cliente.....	59
Figura 14. Diagrama de classes com os métodos.....	60
Figura 15. Cadastro de cliente.....	62
Figura 16. UID Cadastro novo tipo café torrado.....	64
Figura 17. UID Diminuição estoque de café torrado.....	65
Figura 18. UID Exclusão tipo de café torrado.....	65
Figura 19. UID Registro de pedido e venda.....	69
Figura 20. UID Registrar pagamento.....	70
Figura 21. UID Cadastro de contas fixas.....	79
Figura 22. UID Atualização de contas fixas.....	80
Figura 23. UID Relatório de contas fixas.....	81
Figura 24. UID Relatório de vendas.....	81
Figura 25. Diagrama de classes – Sistema Administrativo.....	85
Figura 26. UID Relatório vendas mês a mês	87
Figura 27. UID Relatório total separado por mês das despesas.....	88
Figura 28. UID Saldo vendas.....	88
Figura 29. Diagrama de Seqüência – Exclusão de cadastro de cliente.....	97
Figura 30. Diagrama de Seqüência – Cadastro de café verde.....	97
Figura 31. Diagrama de Seqüência – Insere estoque café verde.....	97

Figura 32. Diagrama de Seqüência – Cadastro novo café torrado.....	98
Figura 33. Diagrama de Seqüência – Diminui estoque café torrado.....	98
Figura 34. Diagrama de Seqüência – Exclusão de café torrado.....	98
Figura 35. Interface inicial do sistema.....	99
Figura 36. Fazer pedido	99
Figura 37. Diminui estoque de café torrado.....	100
Figura 38. Inserção de cliente no banco de dados.....	100
Figura 39. Inserção de café verde no banco de dados	100
Figura 40. Cadastro de café torrado	100
Figura 41. Escolha do ID do café a ser excluído.....	101
Figura 42. Diminui estoque de café torrado.....	101
Figura 43. Exclusão de café torrado.....	101
Figura 44. Verificando cliente para fazer pedido.....	102
Figura 45. Ordem de processamento.....	102
Figura 46. Protótipo cadastro de funcionários.....	102
Figura 47. Protótipo relatório vendas.....	103
Figura 48. Cadastro funcionário ou administrador.....	103
Figura 49. Relatório estoque café torrado.....	104
Figura 50. Relatório vendas.....	104
Figura 51. Interface geral do sistema	105
Figura 52. Relatório folhas de pagamento dos funcionários.....	105
Figura 53. Atualiza funcionário	105
Figura 54. Relatório de vendas com impostos.....	106
Figura 55. Cadastro de despesas de contas fixas	106
Figura 56. Relatório das despesas de contas fixas (Todas)	107
Figura 57. Insere despesa variável ao sistema.....	107
Figura 58. Cadastra nova despesa variável.....	108
Figura 59. Relatório das contas variáveis de transporte.....	108
Figura 60. Relatório vendas mês a mês	109
Figura 61. Relatório despesas mês a mês.....	108
Figura 62. Cálculo do lucro.....	110

RESUMO

Dentre os processos de desenvolvimento de software, os métodos ágeis vêm sendo uma opção bastante explorada. Desde a publicação do Manifesto Ágil em 2001 até a atualidade, a quantidade de seguidores destes métodos tem aumentado significativamente. Princípios ágeis como o comportamento iterativo e incremental e a entrega de uma versão funcional no final de cada iteração, além da comunicação constante com o usuário, obtendo-se uma resposta rápida de sua percepção do sistema, facilitam a detecção de erros e a execução de mudanças durante todo o desenvolvimento do software.

É importante que essa resposta do usuário, incluindo a descrição de sua interação com o sistema, seja rápida e aproveitada da melhor forma. Para tanto, fazer uma modelagem da interação do usuário com o sistema, que facilite o entendimento do fluxo de dados, e incluí-la como uma das práticas dos métodos ágeis, pode trazer benefícios na hora da comunicação, e posterior desenvolvimento.

Esse trabalho faz uma análise de como a modelagem da interação entre o usuário e o sistema é realizada em alguns métodos ágeis atualmente e propõe a inclusão desta modelagem, através do uso dos UIDs, como prática opcional, com o intuito de melhorar o entendimento do usuário, facilitando a comunicação entre ele e o desenvolvedor.

Palavras chave: Métodos Ágeis, Modelagem da Interação entre Usuário e Sistema, Diagrama de Interação de Usuário, *Extreme Programming*, *Scrum*, *Feature Driven Development*, *Dynamic System Development Method*.

ABSTRACT

Among processes for software development, agile methods have become an option that is frequently explored. Since the publication of The Agile Manifest in 2001 to the present date, the number of followers of these methods has increased significantly. Agile principles such as iterative and incremental development and the delivery of a functional version at the end of each iteration, while communicating constantly with users in order to obtain a rapid response of their perception of the system, facilitate the detection of errors and the execution of changes during the entire software development cycle.

It is important that such a user response, including the description of their interaction with the system, be efficient and utilized in the best way. Therefore, to make a model of the interaction between the user and the system that helps with the understanding of the data flow, and include it as one of the practices of agile methods, may bring benefits during communication and later during development.

This work analyses how the modeling of the interaction between users and the system can be carried out in some of the agile methods by means of the utilization of UIDs as an optional practice, aiming to facilitate the communication between developers and users and to improve their understanding about the system.

Keywords: Agile Methods, User Interaction Modeling, User Interaction Diagram, Extreme Programming, Scrum, Feature Driven Development, Dynamic System Development Method.

1. INTRODUÇÃO

Os métodos ágeis são processos de software bastante utilizados atualmente. Eles surgiram no final dos anos 90 e propõem agilizar o processo de desenvolvimento de software. Esses métodos possuem em comum o fato de serem aplicados em projetos de baixa complexidade, utilizando ciclos iterativos curtos, planejamento guiado por funcionalidades, retro alimentação constante, tolerância a mudanças, proximidade da equipe, intimidade com o cliente e um foco no ambiente geral de trabalho da equipe [9]. Entre os métodos ágeis podemos citar: XP (Extreme Programming), Scrum, Crystal, FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method), entre outros.

Esses métodos oferecem às organizações uma grande variedade de práticas e enfoques para o desenvolvimento de software, propiciando que as organizações escolham o método que melhor supra suas necessidades. Entretanto, apesar destes métodos serem guiados por funcionalidades, onde um conjunto de funcionalidades é desenvolvido em cada iteração, eles especificam estas funcionalidades de diferentes maneiras. Além disso, por não fazer parte dos princípios ágeis, a modelagem da interação entre o usuário e o sistema nem sempre é realizada nos diferentes métodos ágeis. Entretanto, para aplicações interativas que apresentam bastante troca de informações a modelagem da interação entre o usuário e o sistema poderia facilitar a comunicação entre o usuário e o desenvolvedor. Aplicações interativas podem ser vistas como aquelas que apresentam muita interação, que é a comunicação entre o usuário e o sistema. [15]

Uma técnica de representação da interação entre o usuário e o sistema são os Diagramas de Interação do Usuário (UIDs – User Interaction Diagrams). Eles representam a interação entre o usuário e o sistema necessária para realizar as tarefas desejadas pelo usuário de uma aplicação [21]. Os UIDs podem ser usados em conjunto com os casos de uso, portanto, podem ser usados em qualquer processo de software que os utilize. Para cada caso de uso pode ser definido um UID composto por um conjunto de estados de interação, que incluem as informações trocadas durante a interação usuário-sistema, conectados através de transições.

1.1 Objetivo geral

Analisar a modelagem da interação entre usuário e sistema nos processos dos métodos ágeis e a vantagem da inclusão dos UIDs como ferramenta para modelar esta interação.

1.2 Objetivos específicos

- Estudar as características e vantagens dos métodos ágeis em geral e estudar, detalhadamente, os métodos ágeis XP, Scrum, FDD e DSDM, principalmente a modelagem da interação entre usuário e sistema.
- Estudar os UIDs, utilidades e conceitos.
- Propor a modelagem da interação entre usuário e sistema a través dos UIDs em cada um dos métodos ágeis estudados.
- Utilizar a proposta feita em um ou mais estudos de caso.
- Analisar as vantagens da modelagem da interação entre usuário e sistema através da inclusão dos UIDs nos métodos ágeis utilizados.

1.3 Justificativa do trabalho

Com o crescimento dos métodos ágeis e sua inclusão no mercado, surgiu a proposta de analisar como é feita a modelagem da interação entre o usuário e o sistema nestes métodos, e se, para o desenvolvimento de aplicações interativas, existe vantagens na inclusão dos UIDs nos processos destes métodos ágeis, com o objetivo de modelar tal interação.

1.4 Estrutura do trabalho

Esse trabalho está estruturado da seguinte maneira. No capítulo 2 são apresentados os conceitos básicos dos métodos ágeis assim como uma explicação mais detalhada dos métodos *Extreme Programming (XP)*, *Scrum*, *Feature Driven Development (FDD)* e *Dynamic System Development Method (DSDM)*.

No capítulo 3 são apresentados os conceitos dos Diagramas de Iteração do Usuário (UID).

No capítulo 4 é feita a inclusão dos UIDs no processo de cada um dos quatro métodos ágeis apresentados no capítulo 2.

No capítulo 5 são desenvolvidas duas aplicações utilizando os quatro métodos ágeis juntamente com os UIDs, de acordo com o que foi especificado no capítulo 4.

No capítulo 6, para concluir, são apresentadas as considerações finais do trabalho e os trabalhos futuros relacionados com o tema.

2. MÉTODOS ÁGEIS

Os métodos ágeis surgiram como uma alternativa aos métodos tradicionais para o desenvolvimento de software.

Em 2001 foi publicado o Manifesto Ágil, por um grupo que praticava preceitos ágeis, que tem como princípios [14]:

- Indivíduos e interações são mais importantes que processos e ferramentas.
- Software funcionando é mais importante do que documentação completa e detalhada.
- Colaboração com o cliente é mais importante do que negociação de contratos.
- Adaptação a mudanças é mais importante do que seguir o plano inicial.

Os métodos ágeis são caracterizados pelas repetidas iterações, enquanto o software é desenvolvido. Cada iteração apresenta uma entrega para o cliente, onde é informado o que foi feito até o momento, com o intuito de intercambiar opiniões e idéias, tanto entre os programadores e projetistas, quanto com os clientes.

Se o cliente mudar de opinião constantemente, por não saber bem o que quer, a utilização dos métodos ágeis é bastante indicada. Nos métodos tradicionais qualquer mudança depois do projeto já estar encaminhado pode causar um atraso muito grande, e pode até inviabilizar o projeto. Com os métodos ágeis é possível adaptar essas mudanças, sem prejudicar o desenvolvimento do projeto, porém esses métodos são indicados, em sua maioria, para projetos e equipes de desenvolvimento não muito grandes.

Os métodos ágeis priorizam o desenvolvimento rápido do projeto, pois visam ter um cliente satisfeito, tanto com o software quanto com o prazo de entrega deste. Uma característica importante e positiva são as entregas contínuas de software funcionando que permitem ao cliente ver o programa sem ter que esperar muito por isso [13].

A seguir serão explicados em detalhe quatro métodos ágeis, XP, SCRUM, FDD e DSDM, os quais serão usados junto com os UIDs para o desenvolvimento desse trabalho.

2.1 *Extreme Programming* – XP

O XP (*Extreme Programming*), criado por Kent Beck no final da década de 90, é o método ágil mais conhecido atualmente. Assim como outros métodos ágeis, entre suas características podemos citar: grupos pequenos de desenvolvedores; softwares desenvolvidos em um tempo menor do que os que utilizam métodos tradicionais; requisitos dinâmicos que não estão totalmente definidos e que mudam constantemente [6]. “No XP o cliente define as funcionalidades do sistema que será desenvolvido por meio das chamadas estórias do usuário, e as prioriza em seguida” [7].

Este método ágil está baseado em um conjunto de valores, princípios e práticas, apresentados adiante.

2.1.1 **Processo do XP**

O processo do XP (Figura 1) tem cinco fases que são explicadas a seguir [6].

2.1.1.1 Exploração: Fase na qual se deve entender o escopo do sistema. Começa com a escrita das estórias de usuário, que dependendo do tamanho devem ser divididas em estórias menores, para que o trabalho seja mais rápido. As estórias devem ser sucintas. Nesta fase, os desenvolvedores também testam as tecnologias e configurações para serem usadas no sistema, e pode-se convidar um especialista na tecnologia escolhida para resolver pequenos problemas que possam surgir. Isto ajuda a ter uma idéia da implementação no planejamento. Além disso, se faz a estimativa do tempo que será gasto na tarefa que será realizada no futuro, dependendo das estórias de usuário.

2.1.1.2 Planejamento: Definição, de acordo com estimativas entre o cliente e os desenvolvedores, da data e o conjunto de estórias de usuário que serão entregues na primeira vez. Passos para auxiliar nessa fase:

- Classificação pelo cliente das estórias de usuário em valor alto, médio e baixo.
- Classificação pelo desenvolvedor das estórias de usuário em risco alto, médio e baixo.
- Estimativa, baseada na experiência, do tempo para desenvolvimento das estórias de usuário, pelos programadores.

- Escolha pelo cliente das histórias de usuário para a próxima entrega.

2.1.1.3 Iterações para entrega: Duram de uma a quatro semanas e contêm várias histórias de usuário para serem desenvolvidas nesse tempo. São escritos os testes de aceitação pelo cliente para cada história e são escritos os testes de unidade pelos desenvolvedores, antes da implementação. “A primeira iteração mostra como a arquitetura do sistema irá se comportar” [6]. No final de cada iteração, feitos todos os testes de aceitação e de unidade, o cliente receberá uma versão nova e funcional do sistema.

2.1.1.4 Produção: Começa no final da última iteração. Novos testes devem ser criados para provar a estabilidade do sistema, que será lançado para que o cliente possa usá-lo no seu ambiente de trabalho. O sistema continuará evoluindo, pois não terminou de ser implementado, podem existir muitas histórias para serem desenvolvidas ainda ou o cliente pode incluir alguma outra funcionalidade. Nesse caso, o processo volta de novo à fase de planejamento.

2.1.1.5 Fim do projeto: Ocorre quando não existem mais histórias para implementar. Escreve-se um relatório das funcionalidades do software para poder alterá-las no futuro, se for necessário. E se faz uma reunião entre todos os que participaram do projeto para analisar o que foi feito e o que pode ter causado algum problema.

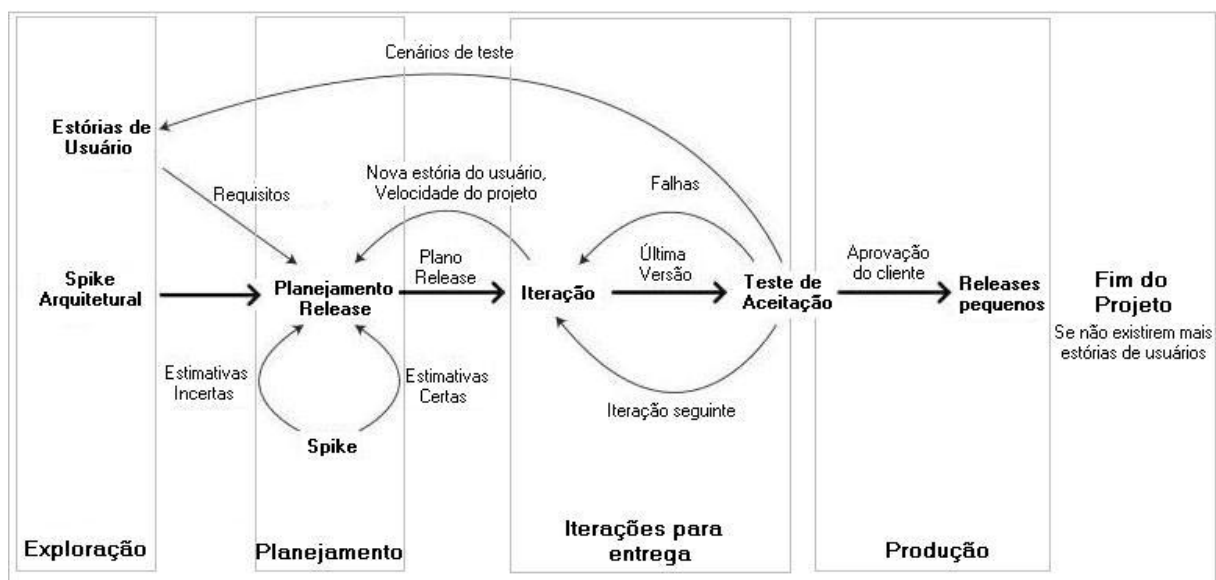


Figura 1. Processo do XP (Adaptado de [5] e [18])

2.1.2 Valores do XP

Os valores do XP existem para tentar fazer baixar o custo das alterações de requisitos que são realizadas [10] e [12]. A seguir são explicados os 5 valores do XP.

- **Comunicação:** Esse valor tem como objetivo promover a comunicação entre os clientes e os desenvolvedores, mantendo o melhor relacionamento possível entre eles. Essa comunicação tem que ser rápida e eficaz. Para isto é dada maior prioridade às conversas pessoais, que são melhores que os telefonemas que, por sua vez, são melhores que emails. Além disso, trabalhar na mesma sala é melhor que fazê-lo em salas separadas.
- **Simplicidade:** Valor relacionado com o desenvolvimento do software, simplicidade é fazer um software simples, com poucas classes e métodos. Isto significa não implementar o que talvez possa servir no futuro, mas implementar só o necessário, o que o cliente tem certeza que vai usar. Isso deve ser definido nas contínuas comunicações entre o cliente e o desenvolvedor.
- **Feedback:** Visa diminuir os problemas na hora da entrega do produto final. O desenvolvedor deve testar o código constantemente para verificar os erros. Além disso, são feitas várias entregas do software totalmente funcional ao cliente, que revisará se está tudo como ele pediu e, no caso de não estar, as mudanças deverão ser feitas rapidamente.
- **Coragem:** Nem todo mundo consegue se expressar com a mesma facilidade, portanto é necessário coragem para conseguir se comunicar da maneira certa. Fazer mudanças no software pode trazer resultados bons ou não, e precisa-se de coragem para experimentar essas mudanças. Para ter o *feedback* necessário também é importante ter coragem, pois os desenvolvedores devem aceitar os erros e explicar ao cliente no caso de surgir alguma dúvida.
- **Respeito:** Esse é o valor mais importante e básico de todos, sem ele é difícil que um projeto possa ser concluído. É necessário saber ouvir, compreender e respeitar o

pensamento de todos os que trabalham no projeto. Isso ajuda na comunicação e se traduz no sucesso do projeto.

2.1.3 Princípios do XP

Os princípios servem de ponte entre os valores e as práticas e estão apresentados a seguir.

- *Feedback* rápido
- Simplicidade
- Mudanças incrementais
- Abraçar mudanças
- Trabalho de qualidade

2.1.4 Práticas do XP

As práticas do XP servem para aplicar os valores. O XP possui 13 práticas, apresentadas a seguir [3] e [6].

- **Jogo de planejamento:** É feito várias vezes durante o projeto, em iterações semanais. O cliente define através de estórias, descritas em pequenos cartões e com uma linguagem simples, o que ele quer que seja implementado, o escopo, a prioridade e as datas de entrega. A equipe estima o tempo e custo de desenvolvimento e planeja os *releases*, que são intervalos de tempo de no máximo dois meses para entregar funcionalidades ao cliente, e que estão divididos em iterações, que contêm um conjunto de estórias para serem desenvolvidas dentro de uma e três semanas. Depois de cada iteração o cliente analisa e fornece o *feedback* correspondente. É no jogo de planejamento que se decide o que deve ser feito e o que pode ser adiado.
- **Pequenas versões:** Depois de cada iteração são entregues versões pequenas e funcionais do software ao cliente, para que ele possa avaliar constantemente verificando que o programa está de acordo com os requisitos especificados por ele.

Isso é bom tanto para o cliente, que recebe uma parte do programa, quanto para o programador, que pode corrigir algum erro de maneira rápida.

- **Metáfora:** Para simplificar a comunicação entre cliente e desenvolvedor são usadas metáforas, que são descrições do software com uma visão mais próxima à realidade do cliente, para que ele possa entender melhor.
- **Projeto simples:** O programa desenvolvido deve ser o mais simples possível. Devem ser implementadas as funcionalidades necessárias no momento, sem considerar as que possam ser úteis no futuro.
- **Testes:** São criados antes do código e servem para que os programadores possam ver se o programa tem erros ou não, e possam analisar se o programa faz realmente o que tem que ser feito. Existem dois tipos de testes, os de unidade, que são escritos pelos desenvolvedores e verificam se cada componente do sistema funciona corretamente, e os de aceitação, que são escritos pelos clientes e testam a interação entre os componentes.
- **Refatoração:** É a limpeza ou reestruturação contínua do código para se ter uma compreensão melhor do que ele faz. Muda-se, não o que ele faz e sim como ele o faz, ou seja, muda-se a estrutura do código e não a funcionalidade. Assim é mais fácil reutilizá-lo, pois fica mais simples e ajuda a tirar as repetições.
- **Programação em pares:** O código é escrito e revisado sempre por dois programadores, revezados constantemente no computador, que compartilham o conhecimento e criam uma solução mais simples, comparando com a que teria sido criada por um programador sozinho. O intercâmbio de pensamentos e maneira de ver o programa de duas pessoas ajuda na implementação dele, pois enquanto um digita e está concentrado na linha que está digitando, o outro consegue enxergar os erros rapidamente. Com isto, existe uma inspeção contínua do código, o que diminui a ocorrência de erros e faz com que os desenvolvedores produzam mais, ficando mais concentrados no que estão fazendo, não querendo decepcionar a outra pessoa.

- **Propriedade coletiva:** O código pode ser alterado a qualquer momento, sem deixar de fazer os testes previstos por qualquer desenvolvedor que faz parte do projeto. Isto ajuda na simplificação do código, na detecção de erros e, no caso de alguma pessoa deixar o projeto, isto não prejudicará o desenvolvimento, pois todos vão conhecer o código o suficiente para continuar o desenvolvimento. O código é propriedade de todos e todos são responsáveis por ele.
- **Integração contínua:** Cada parte do software, que é desenvolvido pelos pares de programadores, deve ser integrada ao resto do software. Esta integração tem que ser serial e contínua: cada par integra sua parte sem que outros o façam no mesmo momento e, depois disso, testa o software. Se tiver algum erro ou problema de compatibilidade, quem integrou o seu código por último tem que corrigir ou resolver. A integração é feita várias vezes ao dia e é feita, normalmente, usando-se uma única máquina de integração.
- **Semana de 40 horas:** O trabalho dos desenvolvedores deve ser de 40 horas na semana. No caso de ser necessário pode-se exceder essa quantidade, fazendo horas extras, mas não por um tempo maior que uma semana, pois quando uma pessoa não descansa o suficiente ou trabalha de mais não produz da mesma maneira quanto se trabalhasse o horário normal. Se o projeto obriga a ter mais do que 40 horas semanais por mais de uma semana é sinal que o planejamento inicial deve ser melhorado.
- **Cliente presente:** Para que o projeto seja desenvolvido de uma melhor forma é recomendável que o cliente esteja sempre por perto. As comunicações entre o cliente e os desenvolvedores devem ser contínuas para que o cliente possa resolver as possíveis dúvidas dos programadores. Isto aumenta a velocidade do trabalho e diminui as implementações erradas, que acontecem pelo pouco entendimento de algum requisito pedido. Se for possível, o cliente deve fazer parte da equipe de trabalho, pois ter uma comunicação constante é muito importante.
- **Padronização do código:** O código deve ser escrito por todos os programadores seguindo um padrão comum, sem importar qual seja, pois o fato de ter o código

padronizado ajuda na compreensão do código e na comunicação entre os desenvolvedores. Com isso parece que o código foi escrito por uma pessoa só.

- **Reuniões em pé:** São reuniões rápidas feitas no início do dia de trabalho. Duram mais ou menos 20 minutos. O seu objetivo é “atualizar a equipe sobre o que foi implementado no dia anterior e trocar experiências das dificuldades enfrentadas. Neste momento também são decididas as histórias que serão implementadas no dia e em conjunto são definidos os responsáveis por cada uma delas” [16].

As práticas do XP devem ser usadas em conjunto, pois existe uma relação direta entre elas, e todas devem ser aplicadas para garantir a agilidade do processo. “A maioria das práticas do XP causa polêmica a primeira vista e muitas não fazem sentido se aplicadas isoladamente.” [4]

2.1.5 Papéis do XP [6]

Os papéis do XP são sete:

- **Programador:** É quem analisa, projeta e implementa o sistema, estima os prazos das histórias criadas pelos clientes, escreve e realiza os testes de unidade, trabalha em par, e solicita, no caso de ter dúvidas, esclarecimentos aos clientes.
- **Cliente:** Define as histórias nas quais escreve as funcionalidades que o software precisa, indicando a prioridade delas. Escreve e executa os testes de aceitação. Se for solicitado, esclarece as dúvidas que os programadores possam ter, pois ele sabe o que quer e o que deve ser feito no software.
- **Testador:** Ajuda o cliente a escrever os testes de aceitação. Após a implementação, o testador verifica se o software atende todos os testes e, então, publica os resultados. É recomendável que o testador não seja desenvolvedor.
- **Rastreador:** Coleta sinais vitais do projeto uma ou duas vezes por semana. Mantém a equipe informada do que acontece e toma atitudes quando alguma coisa parece ir mal.

- **Treinador:** Responsável pelo desenvolvimento do projeto. É recomendável que ele seja quem conhece mais sobre o processo, os valores e as práticas do XP. Ele olha os erros e avisa aos desenvolvedores sobre eles. Ajuda no que puder para manter a visão do projeto. Delega as tarefas entre os desenvolvedores.
- **Consultor:** Membro externo da equipe, com o conhecimento necessário para auxiliar em alguns problemas específicos.
- **Gerente:** Responsável pela administração do projeto, ele agenda as reuniões de planejamento e escreve um relatório sobre elas. Tem um contacto direto com o cliente, para que o mesmo participe no desenvolvimento.

2.2 *Scrum*

Scrum, cuja definição foi formalizada por Ken Schwaber nos anos 90, é um método ágil direcionado para a gerência de projetos. O objetivo desse método é a entrega, dentro de iterações, de um software de qualidade que seja útil para o cliente. Essas iterações são formadas por *Sprints*, que são ciclos de 30 dias, onde se trabalha para alcançar objetivos específicos. Estes objetivos são representados por uma lista de funcionalidades, definidas pelo Dono do Produto (*Product Owner*), que é constantemente atualizada e priorizada, chamada *Product Backlog* [17].

O Scrum adota uma abordagem empírica, está baseado em pequenas equipes auto-organizadas e estabelece um conjunto de práticas de gerenciamento.

2.2.1 *Processo do Scrum*

O processo do Scrum está dividido em três fases: *PreGame* ou Pré-Planejamento, *GamePhase* ou Desenvolvimento e *PostGame* ou Pós-Planejamento.

O Ciclo de vida do *Scrum* está detalhado a seguir.

- São feitas reuniões diárias, *Standup Meeting* ou *Daily Meeting*, de aproximadamente 15 minutos, onde o *Scrum Master* se reúne com a equipe para saber o que fez no dia anterior, comentar os problemas que surgiram e fala sobre o que será feito no dia que se inicia.
- No início de cada *Sprint* é feita uma reunião onde a equipe decide o que vai ser implementado nela, depois do *Product Owner* ter priorizado os itens do *Product Backlog* (lista das funcionalidades, feita pelo cliente). A reunião é chamada *Sprint Planning Meeting*. “As tarefas alocadas em um *Sprint* (período de até 30 dias) são transferidas do *Product Backlog* para o *Sprint Backlog* (lista de funcionalidades)” [11]. Essas tarefas são responsabilidade da equipe, que decide como serão executadas.
- Quando termina um *Sprint* é realizada uma reunião onde a equipe apresenta as funcionalidades implementadas ao cliente. Essa reunião é chamada *Sprint Review Meeting*. Depois disso se faz um *Sprint Retrospective*, “que serve para identificar o que funcionou bem, o que pode ser melhorado e que ações serão tomadas para melhorar” [11], e se reinicia o ciclo. Inicia-se outro *Sprint*.

As fases (Figura 2) do *Scrum* são:

- ***PreGame***: Os requisitos são descritos e priorizados no *Product Backlog* e é feita a estimativa de esforço para cada um deles. É definido o sistema, a equipe, as ferramentas. É feita uma avaliação dos riscos e treinamentos necessários, e uma proposta de desenvolvimento baseada nos itens da lista. “O *Product Backlog* é constantemente atualizado com novos itens ou com itens mais detalhados, bem como com estimativas mais precisas e novas ordens de prioridade.” [6]
- ***GamePhase***: Nessa fase o software é desenvolvido através de vários *Sprints* [6] e são produzidos novos incrementos de funcionalidade do produto. São realizadas quase todas as práticas.

- **PostGame:** Entrega final do sistema. Requisitos estão completos, não é possível adicionar mais. São realizados os testes para avaliar o sistema. É feita uma documentação.

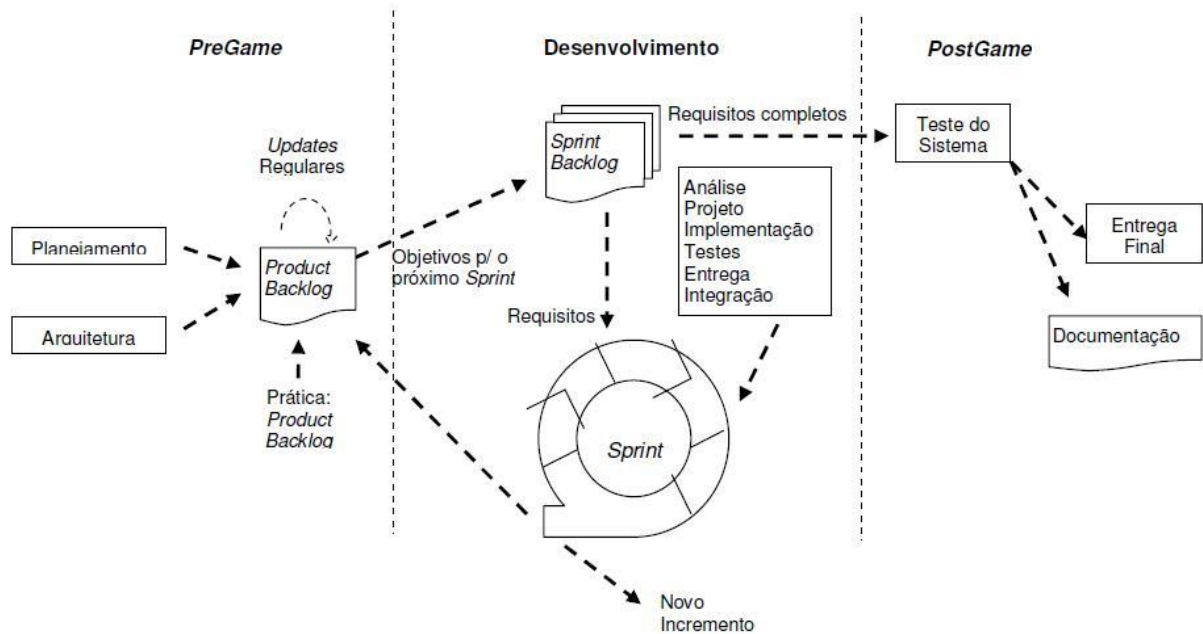


Figura 2. Fases do Scrum [6]

2.2.2 Práticas do Scrum

“O Scrum estabelece um conjunto de regras e práticas gerenciais que devem ser adotadas para o sucesso de um projeto que utilize o método” [6]. As práticas são:

- **Product Backlog** (Figura 3): É uma lista feita pelo cliente com as funcionalidades que serão desenvolvidas no projeto, ordenada por prioridade de execução e constantemente atualizada. É o ponto de partida do ciclo do Scrum, onde se definem também a tecnologia e a estratégia a serem usadas. Cada item da lista deve ter especificações tais como descrição, estimativa em horas, responsável e prioridade.

Prioridade	Item	Descrição	Tempo Estimado	Responsável
Muito Alta				
	1	Conexão com o Banco de Dados	40	Ana
	2
	3	Cadastro de Usuários	...	João
Alta				
	4			
	5	..		
	6			
Média				
	7	Impressão de Relatórios		
	8
	9			

Figura 3. Exemplo de *Product Backlog* [6]

- ***Sprint***: São períodos de, no máximo, 30 dias onde são executados alguns dos itens que foram definidos no *Product Backlog*, e que são listados no *Sprint Backlog*. São feitas reuniões diárias para decidir o que vai ser feito, mas o *Sprint* não tem um processo definido. Quando termina o *Sprint* se cria uma versão pequena do software, que logo é integrada com o que já foi feito e são realizados testes. Depois de um *Sprint* terminar começa outro com uma nova lista de itens.
- ***Sprint Backlog***: É uma lista de funcionalidades escolhidas do *Product Backlog* pela equipe, o *Scrum Master* e o cliente. Essas funcionalidades são escolhidas dependendo das suas prioridades e serão implementadas no próximo *Sprint*.
- **Reunião de planejamento do *Sprint***: Aqui se define o que vai ser priorizado no *Product Backlog*, para depois colocar os itens no *Sprint Backlog*.
- **Reuniões Diárias do *Scrum***: São reuniões com duração de quinze a trinta minutos onde os membros da equipe falam sobre o que foi feito desde a última reunião, os problemas que surgiram e o que será feito até a próxima reunião. No caso dos problemas, o jeito de solucionar cada um deles não é abordado nessa reunião, senão em outra específica para isso, onde só participam as pessoas envolvidas em cada um deles [6].
- **Revisão do *Sprint***: É feita uma reunião na qual a equipe junto com o *Scrum Master* apresenta para o cliente o que foi realizado no *Sprint*. Os resultados são avaliados e se

decide o que vai ser feito a seguir. Novos itens podem ser adicionados ao *Product Backlog*.

- **Estimativa de esforço:** Realizada freqüentemente para ter uma idéia mais exata do tempo que vai ser necessário para a realização de uma tarefa.

2.2.3 Equipe do *Scrum*

- ***Scrum Master*:** Organiza as reuniões diárias. Representa o gerente do projeto. Atua protegendo a equipe de qualquer coisa que impeça continuar com o desenvolvimento e não permita a entrega do software. Além disso, ele se encarrega de ver se as equipes estão usando o *Scrum* corretamente, e as motiva mantendo o foco na entrega da *Sprint*.
- **Dono do Produto (Product Owner):** Responsável pelo *Product Backlog*. Escolhe o que será trabalhado em cada *Sprint*, e calcula o esforço de cada item. Geralmente é um papel desempenhado pelo cliente.
- **Equipe *Scrum*:** Grupo de, no máximo, sete pessoas. Se for necessário ter um grupo maior é recomendável dividir em grupos menores. No caso de equipes múltiplas, o projeto inicia-se com uma única equipe e o resto é incorporado depois da primeira *Sprint*. A equipe atua junto com o *Scrum Master* para atingir os objetivos de cada *Sprint*. Define o *Sprint Backlog* e sugere o que precisa ser acrescentado ou removido do projeto. É responsável por entregar soluções.
- **Cliente:** Participa na elaboração do *Product Backlog*.

2.3 *Feature Driven Development* – FDD

O *Feature Driven Development* (FDD – Desenvolvimento Dirigido a Funcionalidades) é um método ágil para gerenciamento e desenvolvimento de software, criado em 1997 por Peter Coad e Jeff De Luca, num grande projeto em Java para o United Overseas Bank, em Singapura. “Combina as melhores práticas do gerenciamento ágil de projetos com uma

abordagem completa para Engenharia de Software orientada por objetos, conquistando os três principais públicos de um projeto de software: clientes, gerentes e desenvolvedores.” [8]

O FDD é um método iterativo que está focado em pequenas iterações que resultam em entregas de parte do sistema funcionando, para monitorar o progresso do projeto. O seu desenvolvimento é voltado à *Feature*, ou funcionalidade, que representa um requisito funcional do sistema, solicitado pelo cliente. É apropriado para todo tipo de projeto como projetos iniciais, atualização de código existente, criação de uma segunda versão, ou substituição de um sistema inteiro em partes.

O FDD estabelece um conjunto de práticas que devem ser seguidas para o sucesso da metodologia. Os envolvidos no projeto devem segui-las, evitando a busca de alternativas próprias que poderiam dificultar o andamento do mesmo. Seus princípios e práticas proporcionam um equilíbrio entre as filosofias tradicionais e as mais extremas, proporcionando uma transição mais suave para organizações mais conservadoras.

O lema do FDD é: “Resultados freqüentes, tangíveis e funcionais.”

Algumas características do FDD são [8]:

- Resultados úteis a cada duas semanas ou menos.
- Blocos bem pequenos de funcionalidade valorizada pelo cliente, chamados “*Features*”.
- Não existem restrições quanto à complexidade do sistema e tamanho da equipe.
- Planejamento detalhado e guia para medição.
- Rastreabilidade e relatórios precisos.
- Monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes, tudo em termos de negócio.
- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, se o plano e a estimativa são sólidos.

O FDD é um método muito objetivo, e está focado principalmente em:

- **Concepção & Planejamento (Projeto):** Aqui se inicia o desenvolvimento do sistema, dura de uma a duas semanas, e são desenvolvidas três das cinco fases que compõem o

FDD (desenvolver um modelo, construir uma lista de características e planejar cada uma delas).

- **Construção:** Aqui são desenvolvidas as outras duas fases (projeto e construção de cada característica), que são a parte iterativa do FDD. As iterações são, normalmente, de duas semanas.

2.3.1 Processo do FDD

O processo do FDD tem cinco fases (Figura 4) bem definidas e integradas.

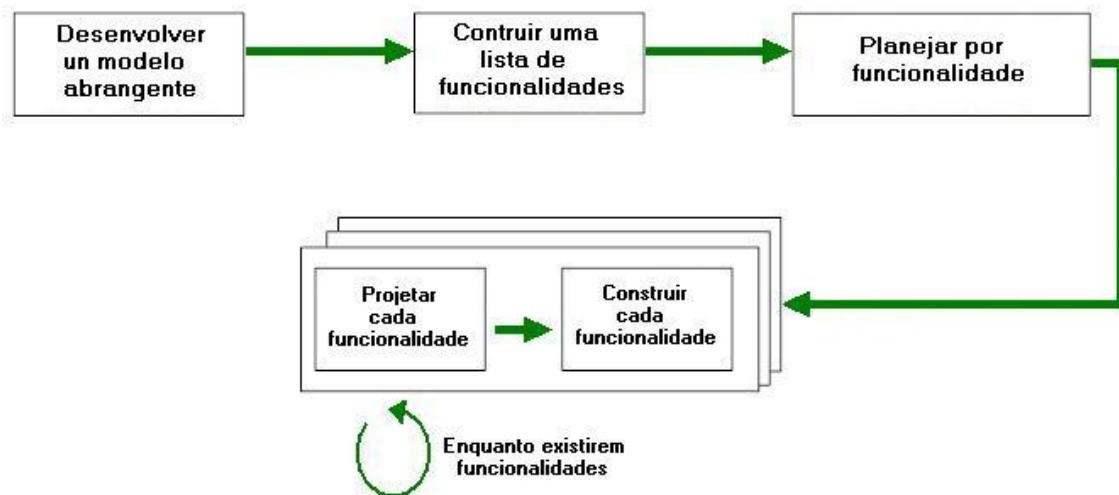


Figura 4. Fases do Processo FDD. (Adaptado de [6])

2.3.1.1 Desenvolver um Modelo Abrangente (DMA): Definição do domínio do sistema, contexto e requisitos para a construção, assim como uma documentação em forma de casos de uso ou uma especificação das funcionalidades. O domínio do projeto é dividido em domínios menores e mais específicos, que serão modelados por um grupo de desenvolvedores. As principais pessoas envolvidas nesta etapa são o Especialista do Domínio e o Projetista. Principal tarefa nesta etapa é a modelagem do domínio da aplicação, construção do diagrama de classe UML, diagrama(s) de seqüência UML e uma lista informal das funcionalidades.

2.3.1.2 Construir a Lista de Funcionalidades (CLF): Construção de uma lista de funcionalidades, importantes para o cliente, do produto a ser desenvolvido. Nela, a equipe de desenvolvimento apresenta cada função esperada pelo cliente baseada na

lista de funcionalidades gerada na etapa anterior, podendo ser necessária a inclusão de novas classes no modelo de domínio, que deverá também ser refeito, apresentando agora os atributos e os métodos de cada classe.

A lista principal é dividida em listas menores que são agrupadas. Cada conjunto de funcionalidades é uma função para uma área de domínio estudado, sendo que a execução de cada grupo de funcionalidades não deverá exceder duas semanas. A lista é revisada pelos usuários e patrocinadores do sistema para sua aprovação.

2.3.1.3 Planejar por Funcionalidade (PPF): Construção de um plano de alto nível, de execução dos conjuntos de funcionalidades. Cada atividade de negócio é atribuída a um chefe de programação e cada classe é atribuída a um programador. A equipe de planejamento formada pelo Gerente do Projeto, Programador Chefe e Gerente de Desenvolvimento, é a responsável pela elaboração do plano de quais funcionalidades serão desenvolvidas.

2.3.1.4 Projetar cada Funcionalidade (DPF): É uma das partes iterativas do método. O chefe de programação realiza o projeto, resultando o diagrama de seqüência detalhado, diagrama de classes atualizado, além das funcionalidades com toda documentação. Nesta etapa são executadas as seguintes tarefas pela equipe e o programador chefe:

- Estudo da documentação existente;
- Desenvolvimento do diagrama de seqüência para o conjunto de funcionalidades;
- Refinamento do modelo de objetos;
- Reescrita das classes e dos métodos;
- Inspeção do projeto.

2.3.1.5 Construir por Funcionalidade (CPF): Última etapa de cada iteração do processo no FDD. Implementa classes e métodos, faz a inspeção do código e o teste de unidade. Algumas tarefas requeridas nessa etapa, executadas pela equipe e pelo programador chefe, são:

- Implementação das classes e métodos;

- Inspeção de código;
- Teste de unidade;
- Integração;
- Testes de integração;
- Entrega do incremento.

Na parte iterativa do método (Detalhar por funcionalidade e Construir por funcionalidade), um pequeno grupo de funcionalidades é selecionado, assim como as funções necessárias para o desenvolvimento desse grupo. Esses processos podem levar, no máximo, duas semanas para serem realizados.

Todos os conjuntos de funcionalidades devem passar pelas etapas de projeto e construção até que o sistema esteja concluído.

2.3.2 Práticas do FDD

As práticas do FDD são detalhadas a seguir:

- **Modelagem dos Objetos de Domínio:** Construção de diagramas de classes UML (*Unified Modeling Language*) que descrevem os objetos relevantes dentro do domínio do problema, bem como os relacionamentos entre eles. Para complementar os diagramas de classe UML, são desenvolvidos diagramas de seqüência UML que descrevem explicitamente como os objetos interagem para cumprir suas responsabilidades.
- **Desenvolvendo através de funcionalidades:** É feita a identificação das funcionalidades do sistema (definidas pelo cliente). Após isso, inicia-se o projeto e a construção de cada uma delas. Uma vez identificadas, as funcionalidades serão utilizadas para guiar o desenvolvimento no FDD, tendo como objetivo mostrar o progresso através da implementação das mesmas.

A execução das funcionalidades, ou conjunto delas, não deve exceder de duas semanas.

Segundo Palmer (2003 apud FAGUNDES[6], 2005), toda característica poderia ser descrita utilizando o seguinte padrão: <ação> <artigo> <resultado> <preposição> <artigo> <objeto>. Por exemplo, “calcular o total da venda”.

- **Propriedade Individual da Classe:** Cada classe ou conjunto de classes é de responsabilidade de um indivíduo.

Algumas vantagens da propriedade individual da classe:

- Quando novos métodos forem adicionados à classe o proprietário terá segurança de que o objetivo da classe será mantido e que as modificações serão feitas corretamente. O proprietário do código pode executar uma atualização mais rapidamente do que outro desenvolvedor que não esteja familiarizado com ele.
- O proprietário do código tende a fazer sempre melhor o que é de sua total responsabilidade.

Algumas desvantagens da propriedade individual da classe:

- Pode acontecer de um desenvolvedor necessitar fazer modificações, na sua classe, que dependem de mudanças que estão sendo feitas nas classes de outro desenvolvedor. Isto poderia resultar em perda de tempo até que os dois desenvolvedores entrassem em sintonia.
 - Caso o proprietário da classe saia do projeto por alguma razão, isso acarretará na necessidade de um entendimento do funcionamento desta por outro desenvolvedor, ocasionando, também, perda de tempo.
- **Equipes de Funcionalidades:** A prática da propriedade individual da classe atribui classes a desenvolvedores específicos. Contudo, sabe-se que o desenvolvimento deve ser por funcionalidade. Por isso, são definidas equipes, com seus respectivos desenvolvedores líderes, onde os componentes possuem as propriedades das classes, e é atribuído a eles um conjunto de funcionalidades.

Algumas considerações sobre as equipes de funcionalidades:

- Uma equipe de funcionalidades deve ter de três a seis membros.

- Por definição, uma equipe de funcionalidades compreende todos os proprietários das classes relacionadas ao desenvolvimento de uma funcionalidade em particular.
 - Cada membro de uma equipe de funcionalidades contribui no desenvolvimento de uma delas, sob a orientação de um desenvolvedor mais experiente.
 - Eventualmente, um proprietário da classe pode trabalhar para outra equipe de funcionalidades. Os desenvolvedores podem pertencer a duas ou três equipes de funcionalidades simultaneamente por um curto período de tempo.
 - Os programadores principais devem trabalhar em conjunto para resolver conflitos e para evitar sobrecarregar qualquer desenvolvedor em particular.
- **Inspeções:** Devem ser feitas durante e ao final de cada iteração, para assegurar a qualidade do projeto e do código. O objetivo principal das inspeções é a detecção de defeitos. É uma ferramenta para eliminação de erros e uma grande oportunidade de aprendizado.
 - **Construções Regulares:** Devem ocorrer durante as iterações, na execução de um conjunto de funcionalidades, para detectar, prematuramente, erros de integração. Uma construção regular assegura também que haja sempre um sistema atual e executável para ser apresentado ao cliente.
 - **Administração de Configuração:** Utilização de um sistema de controle de versões para datar e manter um histórico das alterações feitas em cada classe. Bem como, no que se refere aos requisitos, análise e o projeto de modo que facilite a visualização das modificações feitas.
 - **Relatório dos resultados:** O FDD sugere que todos os resultados ocorridos durante o projeto sejam disseminados para todos os membros da equipe e clientes.

2.3.3 Equipe do FDD

Segundo Palmer (2002 apud FAGUNDES[6], 2005), o FDD classifica os papéis da equipe em três categorias: Chave, Suporte e Adicional.

- **Papéis chave:** Gerente de Projeto, Arquiteto Principal, Gerente de Desenvolvimento, Programador Chefe, Proprietário de Classe e Especialista no Domínio.
 - **Gerente de Projeto:** Responsável financeiro e administrativo do projeto. Uma de suas responsabilidades é gerenciar a viabilidade do projeto oferecendo todas as condições necessárias à equipe para o desenvolvimento do trabalho. No FDD, a “última palavra” é dada por ele.
 - **Arquiteto Principal:** Elabora o projeto geral do *software* a ser desenvolvido, tomando decisões finais em relação ao projeto técnico. Essa função poderá ser dividida entre o Projetista de Domínio e o Projetista Técnico.
 - **Gerente de Desenvolvimento:** Responsável por gerenciar as atividades diárias do projeto resolvendo problemas que poderão ocorrer com a equipe. Pode combinar as atividades desenvolvidas pelo Arquiteto Principal e Gerente de Projeto.
 - **Programador Chefe:** Geralmente é o programador com maior experiência dentro da equipe, participando na análise dos requisitos e no projeto do *software*. Considerado como um dos papéis mais importantes no projeto FDD. Atua principalmente nas duas últimas etapas do processo.
 - **Proprietário de Classe:** Subordinado do Programador Chefe, tendo como tarefas projetar, codificar, testar e documentar. Responsável pelo desenvolvimento das classes atribuídas a ele.
 - **Especialista no Domínio:** Pode ser um usuário, analista de negócios, cliente ou qualquer pessoa que conheça bem o domínio do problema. Sua tarefa é informar as funcionalidades que deverão ser atendidas pelo *software* e entender como os requisitos estão sendo desenvolvidos.

- **Papéis de suporte:** Gerente de Domínio, Gerente de Versão, Especialista na Linguagem, Coordenador de Configuração, *Toolsmith* e Administrador de Sistema.
 - **Gerente de domínio:** Conduz os peritos de domínio a resolver as diferenças de opinião relativa aos requisitos do sistema.
 - **Gerente de Versão:** Responsável por controlar o progresso no desenvolvimento através de constantes revisões em conjunto com o Programador Chefe. Informa suas atividades ao Gerente de Projeto.

- **Especialista na Linguagem:** Membro da equipe responsável por possuir um conhecimento completo de uma linguagem de programação específica ou tecnologia. Este papel é particularmente importante quando é usada uma nova tecnologia.
 - **Coordenador de Configuração:** Pessoa responsável pelas tarefas de administração do sistema de controle de versão e a publicação da documentação, durante a atividade de construção.
 - **Toolsmith¹:** Responsável por construir ferramentas de suporte para o desenvolvimento, teste e conversão de dados no projeto. Também pode trabalhar com modelagem e manutenção de bancos de dados e *websites* para propósitos específicos do projeto.
 - **Administrador de Sistema:** Possui a tarefa de configurar, administrar e diagnosticar os servidores, estações de trabalho e desenvolvimento e testar os ambientes usados pela equipe do projeto.
- **Papéis adicionais:** Testadores, Desenvolvedores e Escritor Técnico.
 - **Testadores:** Verificam se o sistema que está sendo produzido satisfará os requisitos do cliente.
 - **Desenvolvedores:** Responsáveis por converter dados existentes ao formato requerido pelo novo sistema e participar no desenvolvimento de novos lançamentos.
 - **Escritor técnico:** Responsável pela documentação de usuário.

2.4 *Dynamic System Development Method - DSDM*

O DSDM, desenvolvido na década dos 90 na Inglaterra, é um framework para o desenvolvimento ágil de software, portanto uma de suas características é a aceitação de mudanças. Ele é uma extensão do RAD (*Rapid Application Development*) e tem como objetivo principal gerenciar as ações dentro dos limites desejados a respeito do tempo e do orçamento [20].

¹ **Toolsmith:** Aquele que se especializa em fazer ferramentas com as quais outros programadores criarão aplicações. <http://www.brunocarvalho.com/06/07/2007/toolsmith/>

Alguns pré-requisitos para a utilização do DSDM podem ser citados:

- Interatividade entre equipe de desenvolvimento, usuário final e gerente do projeto.
- Facilidade de decomposição do projeto em partes.
- Possibilidade de definição dos requisitos de uma forma clara e com opção de priorizá-los.
- O produto final não deve ter um estado de segurança crítica, pois para isso precisará de testes intensivos o que pode fazer que o tempo e orçamento sejam ultrapassados.
- “Projetos que apontam para componentes de software reutilizáveis podem não ser apropriados para um desenvolvimento utilizando o DSDM” [20], pois eles precisam ter um nível de perfeição que o DSDM não pode oferecer.

2.4.1 Processo do DSDM

O processo do DSDM (Figura 5) tem três fases: pré-projeto, ciclo de vida do projeto ou projeto, e pós-projeto. O projeto, por sua vez, está dividido em cinco etapas.

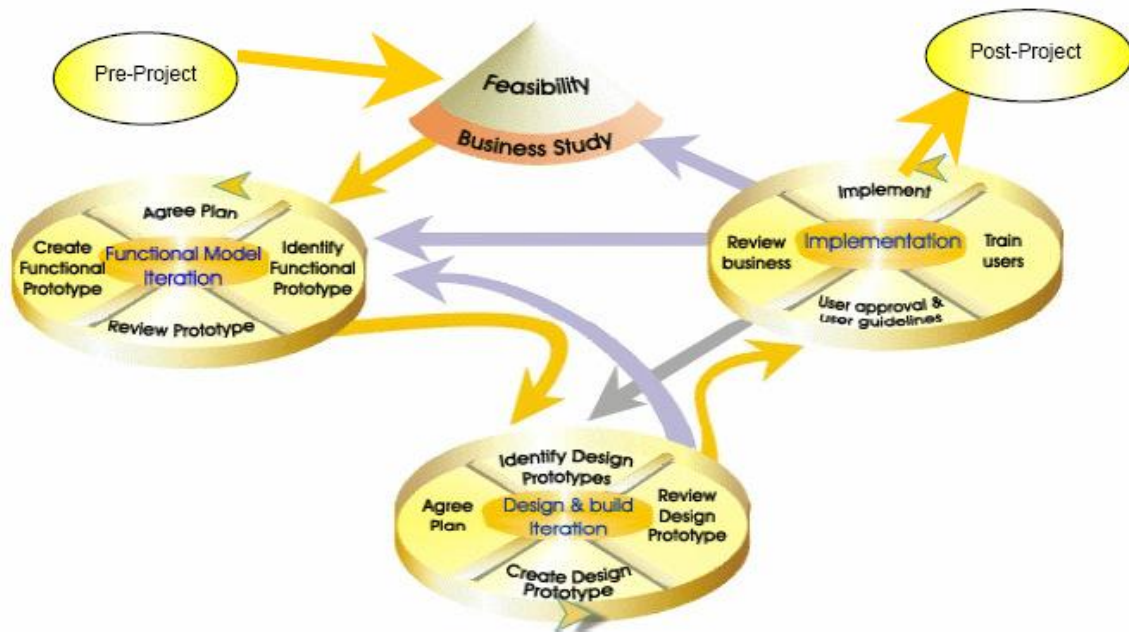


Figura 5 – Processo de desenvolvimento do DSDM [9]

2.4.1.1 Pré-projeto: É identificado o projeto, se realiza o plano inicial para o estudo de viabilidade, é feito o plano de financiamento calculando o orçamento, e a equipe se compromete na realização do projeto. Tudo isto é feito no início para evitar problemas numa etapa avançada do desenvolvimento.

2.4.1.2 Ciclo de vida do projeto: Tem cinco etapas que são: estudo de viabilidade, estudo de negócio, modelagem funcional, projeto e construção, e implementação. As duas primeiras são sequenciais e se complementam. Depois de elas serem concluídas o sistema se desenvolve de forma iterativa e incremental nas outras três etapas.

- **Estudo de Viabilidade:** Nessa etapa é estudado o projeto para ver se é viável sua execução utilizando DSDM. Esse estudo é feito antes de iniciar o projeto e entre as iterações, usando principalmente a prática de *workshop*. As questões revisadas referem-se, por exemplo, aos riscos envolvidos e à adequação do projeto ao DSDM. São preparados um relatório e um protótipo de viabilidade para serem entregues ao cliente, neles está escrita a viabilidade do projeto, além de um esboço do plano para a realização do projeto e um estudo dos riscos mais importantes.
- **Estudo do Negócio:** Depois de verificar a viabilidade de execução do projeto com o DSDM, é feito um estudo mais detalhado do sistema a ser desenvolvido. São definidos e entregues ao cliente, os requisitos, dando-lhes prioridades usando a prática *Moscow*; a área de negócio; o plano de desenvolvimento, usando o *Timeboxing* e calculando-se os prazos e orçamentos de cada parte a ser implementada; o processo de financiamento; e a arquitetura do sistema. É atualizada a lista dos riscos. Tudo isso é definido nas discussões conhecidas como *workshops*.
- **Modelagem Funcional:** Os requisitos, funcionais e não funcionais, que foram definidos anteriormente, são demonstrados por meio de protótipos. Em geral, os requisitos não funcionais não podem ser demonstrados usando protótipos, nesses casos eles são enumerados e é usada a documentação, mas se for possível se prefere ter protótipos deles também. Em cada iteração, e para ter certeza da qualidade do sistema, se implementam testes. Esta etapa de modelagem tem quatro partes: identificar o protótipo funcional, onde são definidas as funcionalidades a serem implementadas no protótipo da iteração corrente; combinar o plano, onde se define como e quando

desenvolver as funcionalidades identificadas na etapa anterior; criar protótipo funcional; e revisar o protótipo, verificando com a ajuda de testes feitos pelos usuários finais, junto com a documentação, se existe alguma coisa que deve ser corrigida. Nesta etapa são entregues ao cliente o modelo funcional e o protótipo funcional. Além disso, se atualiza a lista de requisições e são apagados os itens já implementados.

- **Projeto e Construção (*Design and Build*):** Esta etapa tem como objetivo integrar os componentes funcionais da etapa anterior no sistema, segundo o que o usuário necessita. Também são realizados testes, que são muito importantes. Está dividida em quatro partes: Identificar protótipos do projeto, onde se identificam os requisitos funcionais e não funcionais necessários no sistema testado; combinar o plano, onde se define como e quando se desenvolvem os requisitos; criar protótipo do projeto; e revisar protótipo do projeto, onde se verifica se o sistema projetado é o que se quer. Nesta etapa se entrega ao cliente o protótipo do projeto, para que ele possa testá-lo.
- **Implementação:** Nessa etapa os protótipos testados na etapa anterior são melhorados e entregues aos usuários finais, que são treinados para fazer uso deles. Os próximos requisitos a serem desenvolvidos são definidos. A implementação está dividida também em quatro partes: aprovação do usuário e linhas de guia do usuário, o sistema testado é aprovado pelo usuário para ser implementado e são criadas as linhas de guia para implementação e uso do sistema; treinamento dos usuários para usar o sistema; implementação, o sistema é implementado e implantado no local onde os usuários trabalham e revisar o negócio, onde é visto o impacto do sistema, e se ele alcança os objetivos que foram definidos no início do projeto, se não for assim se volta a uma etapa anterior. No caso de ter os objetivos cumpridos o projeto passa para a fase seguinte, o pós-projeto. Terminando esta etapa deve ser entregue e instalado o sistema, e deve ser feita, em detalhe, a documentação de utilização dele.

2.4.1.3 Pós- Projeto: Começa depois de o sistema ter sido entregue. O objetivo dessa fase é dar suporte e manutenção para que o sistema continue funcionando de maneira eficiente. Nessa fase também podem se fazer relatórios da manutenção, projetos para atualização ou requisição de alterações.

2.4.2 Princípios do DSDM

O DSDM se baseia em nove princípios básicos.

- Envolver aos usuários é a chave para ter um projeto eficiente. Assim, tanto os desenvolvedores quanto os usuários podem chegar a uma decisão mais precisa.
- A equipe DSDM deve ter poder para decidir, sem precisar da aprovação de superiores.
- O foco é a entrega freqüente de produtos, para poder verificar e testar o produto em cada entrega.
- O sistema deve adequar-se às necessidades atuais do negócio. Não é necessário entregar um sistema perfeito e sim centralizar o esforço para que o sistema alcance os objetivos do projeto.
- O desenvolvimento deve ser iterativo e incremental, para chegar a uma solução precisa.
- Mudanças durante o desenvolvimentos são reversíveis.
- Requisitos devem ser escritos em alto nível.
- Os testes são realizados durante o ciclo de vida do projeto. Isso é feito para evitar um custo elevado extra para manutenção e consertos do sistema depois de ter sido entregue.
- Todos os membros da equipe devem colaborar e se comunicar continuamente para o sucesso do projeto.

Além desses princípios, existem outros nos que o DSDM também se apóia.

- “Nenhum sistema é construído perfeitamente na primeira tentativa. Num processo de desenvolvimento de um sistema informatizado 80% da solução podem ser desenvolvidos em 20% do tempo necessário para encontrar a solução perfeita. Para aperfeiçoar a parte final poderá ser necessário que o projeto ultrapasse o tempo e orçamento estipulados” [20]. Já que o DSDM se caracteriza por fazer só o que a empresa precisa, não é necessário obter a solução perfeita.
- A entrega do produto deve ser no tempo desejado, respeitando o orçamento e a qualidade.

- DSDM só precisa que cada passo do desenvolvimento seja terminado com tempo suficiente para começar o seguinte passo. Assim, cada etapa pode começar sem ter que esperar a etapa anterior.
- Tanto as técnicas de desenvolvimento quanto as de gestão de projetos estão inclusas no DSDM.
- A avaliação de riscos deve focar-se na entrega do produto e não no processo de construção.
- A realização de uma tarefa não gera uma recompensa, e sim a entrega do produto completo.
- A estimativa deve se basear na funcionalidade do sistema e não em linhas de código.

2.4.3 Práticas do DSDM

O DSDM fornece diversas práticas que podem ser utilizadas em diferentes projetos:

- **Timeboxing (Caixa de tempo):** O projeto é dividido em partes menores que são colocadas em *timeboxes*, que podem ser executados em paralelo e podem ter tempos diferentes, que variam de uma a duas semanas. Além disso, os *timeboxes* podem ter um orçamento predefinido para desenvolver os requisitos, entre os de alta e baixa prioridade, que estiverem contidos neles. A data definida como limite não pode ser ultrapassada, se isso for ocorrer os requisitos com menor prioridade são omitidos.
- **MoSCoW (Must, Should, Could, Won't):** Essa prática prioriza os requisitos, os quais devem ser classificados em uma das categorias, e por sua vez, cada categoria deve manejar uma prioridade interna dependendo do nível do requisito para saber a ordem de execução dele. As categorias dos requisitos são: Os que o projeto precisa ter (“*Must have*”), pois são importantes para o sucesso do projeto; os que deveria ter se for possível (“*Should have*”), pois são importantes, mas o projeto não depende deles; os que poderia ter (“*Could have*”), pois podem ser deixados de lado sem ocasionar estrago ao projeto; e os que o projeto não terá no momento, mas seria bom ter no futuro (“*Won't have*”).

- **Prototipagem:** São feitos protótipos simples do sistema desde que o projeto é iniciado. Eles servem para descobrir erros no sistema e ajudam no envolvimento dele com o usuário. Existem quatro categorias de protótipos recomendadas pelo DSDM: negócio, usabilidade, desempenho e capacidade, e capacidade técnica.
- **Workshop:** É a prática principal no DSDM. São discussões, que ocorrem durante o projeto entre os membros da equipe, sobre as funcionalidades e o desenvolvimento do projeto em si. Nelas devem ser tomadas as decisões certas em um tempo curto.
- **Testes:** São feitos testes, em sua maioria por usuários que não são técnicos, em todas as iterações durante o desenvolvimento do projeto.
- **Modelagem:** “O DSDM considera que a modelagem ajuda a equipe a adquirir um bom conhecimento sobre o domínio do negócio. Não determina qual o conjunto de modelos a ser usado, apenas guia para quando, onde e como modelar.” [9]
- **Gerenciamento de configurações:** “O DSDM considera imprescindível a gerência de configuração do software e da documentação, visto que um dos seus princípios diz que todas as mudanças durante o projeto podem ser revertidas, o que dá uma maior liberdade para o desenvolvimento.” [9]

2.4.4 Equipe do DSDM

A equipe do DSDM (Figura 6) é formada pelos seguintes papéis:

- Desenvolvedores
- Desenvolvedores *sênior*s
- Coordenador técnico
- Usuário embaixador
- Usuário consultor
- Visionário
- Executivo responsável
- Especialistas do domínio

- Gerente do domínio
- Testador
- Redator
- Facilitador

Alguns destes papéis podem ser executados por vários membros da equipe e cada membro da equipe pode executar um ou vários papéis. “Alguns papéis são ocupados por pessoas com dedicação exclusiva ao projeto e outros por pessoas com dedicação parcial. Os projetos que usam DSDM geralmente possuem de uma a duas equipes, pois se considera preferível trabalhar com várias equipes do que com equipes maiores que seis pessoas.” [9]

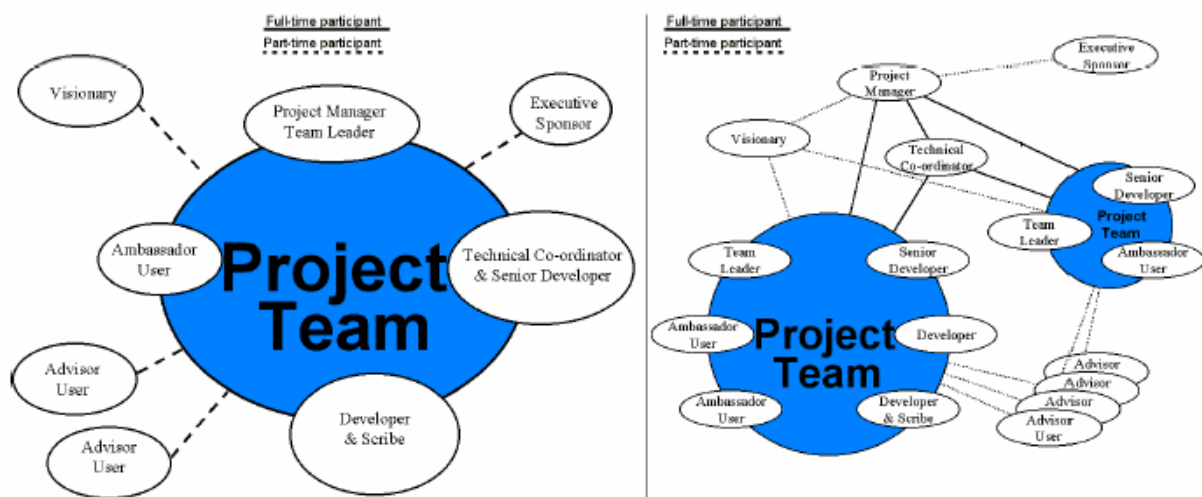


Figura 6 – A equipe de projeto do DSDM[9]

3. DIAGRAMA DE INTERAÇÃO DE USUÁRIO – UID

Nesse capítulo serão explicados os conceitos do Diagrama de Interação do Usuário ou UID (do termo inglês *User Interaction Diagram*).

3.1 Definição

“Um Diagrama de Interação do Usuário ou UID (*User Interaction Diagram*) representa a interação entre o usuário e uma aplicação com intensa troca de informações e suporte à navegação. Este diagrama descreve somente a troca de informações entre o usuário e o a aplicação, sem considerar aspectos específicos da interface com o usuário nem da navegação.” [21]

Um UID é composto por estados e transições. Os estados representam as informações trocadas entre o usuário e a aplicação, e as transições são responsáveis pela troca do foco da interação de um estado para outro. As transições, que conectam os estados, são disparadas geralmente, pela entrada ou seleção de informações pelo usuário.

3.2 Características

Os UIDs focam nas informações necessárias para a interação e não na navegação e interface com o usuário. Também não são usados para descrever diagramaticamente os requisitos não funcionais, que podem ser representados como uma nota anexada aos UIDs, nem o tratamento de exceções [21]. O tratamento de exceções pode ser feito através de outro UID exclusivo para tratar a exceção.

Outra característica importante dos UIDs é que são compreensíveis para usuários quando guiados pelos projetistas. Foram realizados testes junto aos usuários, e eles se mostraram bastante compreensíveis, apesar de não ter tido testes de usabilidade. A simplicidade na notação é importante para que o usuário a compreenda melhor.

3.3 Notação

A notação dos UIDs é apresentada a seguir.

3.3.1 Item de dado: Geralmente escrito em letras minúsculas, representa uma informação única (simples) que aparece durante a interação, pode estar acompanhado do seu domínio, sendo, neste caso, seguido por dois pontos e o nome do domínio. Os domínios são definidos pelo projetista. Se um domínio enumerado for definido, os seus valores devem ser especificados entre chaves e separados por vírgula. Se o domínio do item de dado não for especificado, é assumido o domínio *Texto*. O nome do item de dado pode ser suprimido, nesse caso seria representado por “:” seguido do domínio.

<item de dado>: <domínio>

: <domínio>

3.3.2 Estrutura: Geralmente escrita com a primeira letra em maiúscula, representa uma coleção de informações (itens de dados, estruturas, conjunto de itens de dados ou conjunto de estruturas), relacionadas de alguma maneira. A coleção de informações de uma estrutura é especificada entre parênteses após o nome da estrutura, que é obrigatório. Pode-se utilizar o mesmo nome para duas estruturas diferentes.

<Estrutura> (<item de dado1>, <item de dado2>, ..., <item de dadoN>)

<Estrutura> ()

3.3.3 Conjunto: Representa um conjunto de itens de dados ou estruturas. Multiplicidade representada por min...max frente ao item de dado ou estrutura. Multiplicidade default é 1..N, representada por reticências (...). Para facilitar o entendimento, o rótulo *Conjunto* pode ser incluído após as reticências.

...<item de dado>

...<Estrutura> (<item de dado1>, <item de dado2>, ..., <item de dadoN>)

1..5 <item de dado>

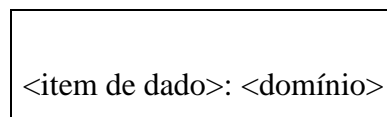
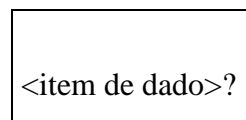
3.3.4 Dado opcional: Representa um item de dado, estrutura ou texto opcional. Representado pelo símbolo “?”. Ou por um conjunto com multiplicidade 0..1. No caso de uma entrada do usuário, um dado opcional pode ser representado por um retângulo com linhas pontilhadas.



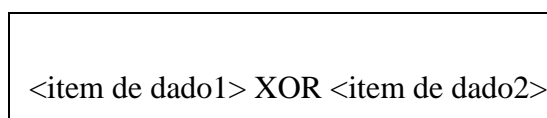
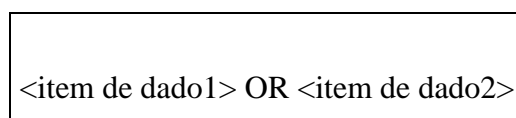
...<item de dado>?

<Estrutura> (<item de dado1>, <item de dado2>,..., <item de dadoN>)?

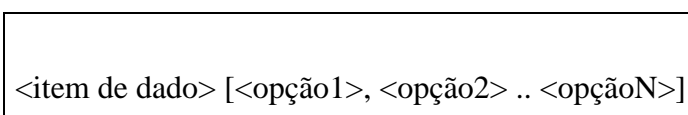
3.3.5 Entrada do usuário: Representa um item de dado ou estrutura fornecida pelo usuário. Toda informação entrada pelo usuário é sempre colocada dentro de um retângulo.



Também são usados os símbolos “OR” e “XOR” entre dois itens de dados na entrada.



3.3.6 Entrada do usuário enumerada: Representa uma entrada do usuário que deve ser selecionada a partir de opções fornecidas pelo sistema. Essas opções aparecem entre colchetes e separadas por vírgula. Se mais de uma opção pode ser selecionada, a quantidade é indicada frente ao nome do item de dado.



<min>..< ré> <item de dado> [<opção1>, <opção2> .. <opçãoN>]

3.3.7 Saída do sistema: Representa um item de dado ou estrutura retornado pelo sistema. Toda saída é colocada no estado de interação, fora dos retângulos.

<item de dado>: <domínio>

<item de dado >?

<Estrutura> (<item de dado>)

3.3.8 Texto: Texto pré-definido, de caráter explicativo, apresentado pelo sistema durante a interação.

“<texto>“

3.3.9 Estado da interação: Elipse que representa um estado da interação entre o usuário e o sistema. Nunca pode estar vazio. Informações fornecidas pelo usuário e retornadas pelo sistema são colocadas dentro da elipse.

O usuário pode interromper sua interação com o sistema a partir de qualquer estado de um UID. Para não permitir isso, em determinado estado da interação, o projetista deve incluir essa informação como uma nota textual anexada ao UID.

3.3.10 Estado inicial da interação: Estado inicial da interação entre o usuário e o sistema. Representado por um uma pequena transição sem origem.

3.3.11 Estados alternativos da interação: Representação usada quando existe mais de uma saída alternativa, a partir de um estado da interação. O estado que será foco da interação depende da opção selecionada, ou da informação fornecida, pelo usuário.

3.3.12 Sub-estados de um estado da interação: Representação usada quando partes de um estado da interação são excludentes. O usuário tem que optar pelo sub-estado que ele vai seguir durante a sua interação com o sistema.

3.3.13 Chamada de outro UID: Foco da interação transferido para outro UID. Após a execução do outro UID, se não existir transição de retorno para um novo estado de interação, o foco retorna ao estado a partir de onde a chamada do UID foi feita.

3.3.14 Chamada a partir de outro UID: Foco da interação transferido para outro UID. Após a execução do UID corrente, o foco da interação retorna ao estado de interação do outro UID, a partir de onde o UID corrente foi chamado.

3.3.15 Transição do estado da interação: Representa que o estado destino torna-se o novo foco da interação. Uma transição sempre está associada, pelo menos, à seleção de um elemento, de uma opção ou ao fornecimento de uma informação. Uma transição pode ter mais de uma origem, que pode ser um estado, um sub-estado, ou um item de dado ou uma estrutura, retornados pelo sistema. O destino de uma transição é sempre um estado.

Quando o usuário seleciona alguma opção que não muda o foco da interação, a origem e o destino de uma transição podem ser o mesmo estado de interação.

3.3.16 Pré e pós-condições: Similares às pré e pós-condições dos casos de uso. Um UID com pré-condição só pode ser executado se essa pré-condição for satisfeita. É importante salientar que a pré-condição de um UID é diferente da condição de uma transição. A primeira está associada ao UID como um todo, enquanto a segunda está associada a um único estado da interação. Pré-condições são representadas dentro de um retângulo e expressas em linguagem natural.

Pré-condições: <Y>

A definição de pós-condições, nos UIDs, descreve as condições que precisam ser satisfeitas após a execução da interação descrita nos UIDs. São representadas igual às pré-condições.

Pós-condições: <Y>

3.3.17 Notas textuais anexadas aos UIDs: Servem para especificar alguma informação importante, que não pode ser representada no UID. Também usadas para descrever requisitos não funcionais.

3.4 Relacionamentos entre UIDs

3.4.1 Relacionamento de inclusão (inclui): Um UID faz parte de outro UID, e sua seqüência de interação é incluída no outro UID.

3.4.2 Relacionamento de extensão (estende): Um UID pode ser estendido por outro UID. Relacionamento usado quando um UID apresenta comportamento alternativo ou opcional. Se, durante a interação do UID que está sendo estendido, houver uma chamada ao outro UID, então a seqüência de interação definida no outro UID será incluída nele.

3.4.3 Relacionamento de precedência (precede): Um UID só pode ser executado se outro tiver sido executado com sucesso anteriormente. A execução de um UID depende da execução do outro UID.

A representação desses relacionamentos é semelhante à representação usada na UML. Eles são representados graficamente por uma dependência estereotipada com o respectivo rótulo («*inclui*», «*estende*» e «*precede*»).

- **Representação do UID inicial:** Às vezes é necessário representar o UID inicial da aplicação. Ele representa as tarefas que podem ser chamadas diretamente quando a aplicação é iniciada. Essas tarefas são representadas através de UIDs e a chamada delas é representada através do relacionamento inclui.

3.5 UIDs no processo de *software*

Os UIDs podem ser incorporados no processo de desenvolvimento de software, sendo usados, normalmente, em conjunto com uma técnica existente para levantar os requisitos ou mesmo substituindo-a. Os UIDs são definidos durante o levantamento de

requisitos e podem ser utilizados durante o projeto conceitual, projeto de navegação e projeto da interface com o usuário.

4. INCLUSÃO DE UIDs

A proposta desse trabalho é usar os UIDs junto com os métodos ágeis e depois analisar se essa inclusão beneficia o desenvolvimento do projeto, tanto em relação ao tempo de entrega do programa quanto ao entendimento do cliente. O fato de se ter um software funcionando, entregue em pouco tempo, mostra que o desempenho da empresa é bom. E o fato do cliente entender melhor o que vai ser feito no programa que ele está solicitando aumenta a satisfação dele pelo produto e pela empresa, além de ter uma comunicação maior com os desenvolvedores, o que o faz sentir ser parte ativa do projeto.

A inclusão dos UIDs em todos os métodos será feita no início de cada iteração. Cada requisito será analisado para verificar se ele tem troca de informação ou não. Esta inclusão não será feita no início do desenvolvimento porque em cada iteração podem ser adicionados novos requisitos, que precisariam dos UIDs para ser mostrados ao cliente, antes da implementação.

4.1 Inclusão dos UIDs no XP

No caso do XP, a inclusão dos UIDs pode ser realizada antes de iniciar cada iteração, após a fase de planejamento, na fase iterações para entrega, conforme mostrado na figura 7, onde são escolhidas as histórias que vão ser usadas na primeira iteração. Para cada história de usuário que apresenta grande troca de informações entre o usuário e o sistema, será definido também um UID. Isso ajudará na hora da comunicação com o cliente para que o cliente compreenda melhor a interação entre o usuário e o sistema, especificamente as informações trocadas na iteração.

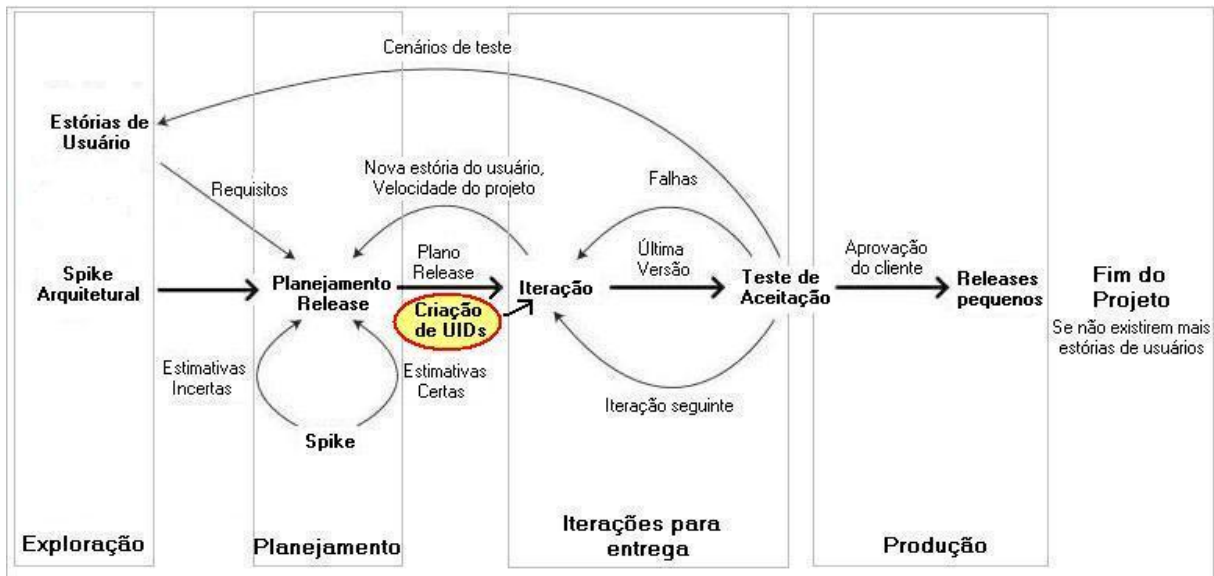


Figura 7. Processo do XP (Adaptado de [5]), com inclusão de UIDs

A modelagem dos UIDs no XP pode ser realizada pelo testador, que é quem faz, junto com o cliente, os testes de aceitação.

Essa modelagem pode ser vista como uma nova prática, que seria chamada modelagem da interação entre usuário e sistema, e pode ser aplicada em conjunto com a prática de jogo de planejamento, onde se decide o que vai ser feito na seguinte iteração.

4.2 Inclusão dos UIDs no Scrum

A inclusão dos UIDs no *Scrum* pode ser realizada na fase de desenvolvimento, chamada também *GamePhase*, conforme mostrado na figura 8. Após a criação do *Product Backlog*, que é a lista das funcionalidades que serão desenvolvidas no projeto, feita pelo cliente, e antes da criação do *Sprint Backlog*, que é a lista das funcionalidades a serem desenvolvidas no próximo *Sprint*, ou próxima iteração. Para cada funcionalidade que está associada com uma grande troca de informações entre o usuário e o sistema, será definido um UID. Assim, o cliente poderá ver como vai funcionar a interação entre o usuário e o sistema segundo os requisitos que ele pediu.

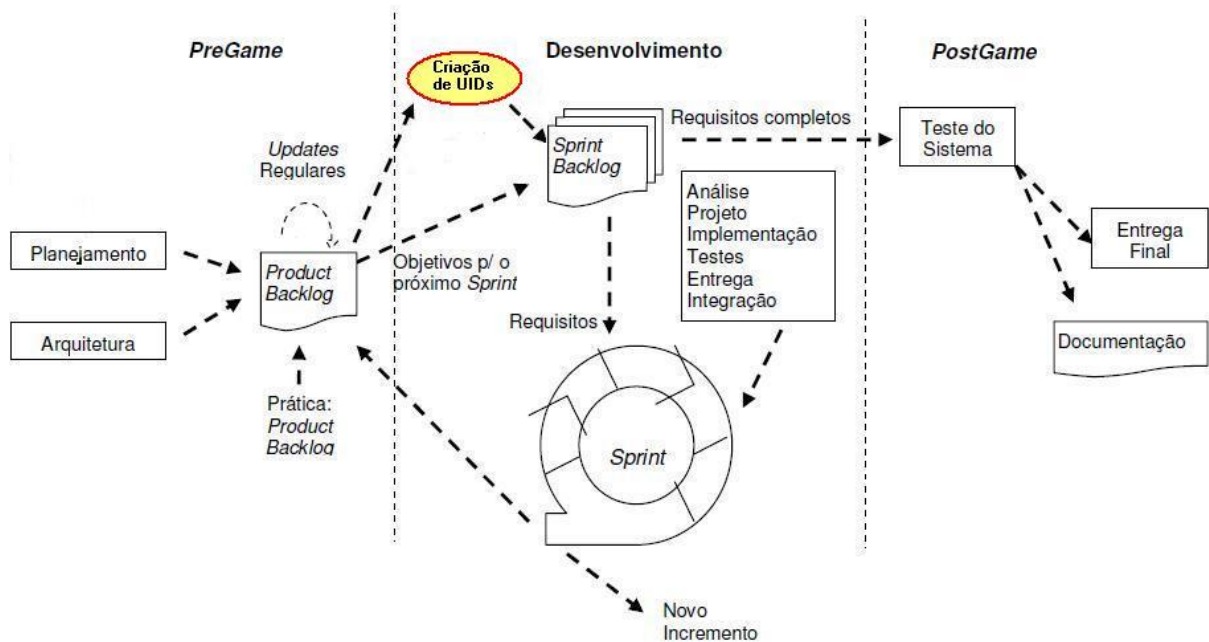


Figura 8. Fases do *Scrum*, com inclusão de UIDs (Adaptado de [6])

A modelagem dos UIDs nesse método pode ser realizada por algum representante da equipe *Scrum*, de preferência por quem define o *Sprint Backlog*.

Essa modelagem, assim como no XP, pode também ser vista como uma nova prática no *Scrum*, chamada modelagem da interação entre usuário e sistema, e pode ser aplicada em conjunto com a prática *Sprint Backlog*, onde são escolhidas as funcionalidades a serem implementadas no *Sprint* seguinte.

4.3 Inclusão dos UIDs no FDD

No caso do FDD, que é um método baseado em funcionalidades, a proposta é incluir os UIDs dentro da fase de construção, conforme mostrado na figura 9, que é quando começa a parte iterativa do método. Antes da fase ‘projetar cada funcionalidade’. É definido um UID para cada funcionalidade que está associada com uma grande troca de informações entre o usuário e o sistema. Assim, poderá ter-se uma visão mais clara e específica do comportamento de cada requisito.

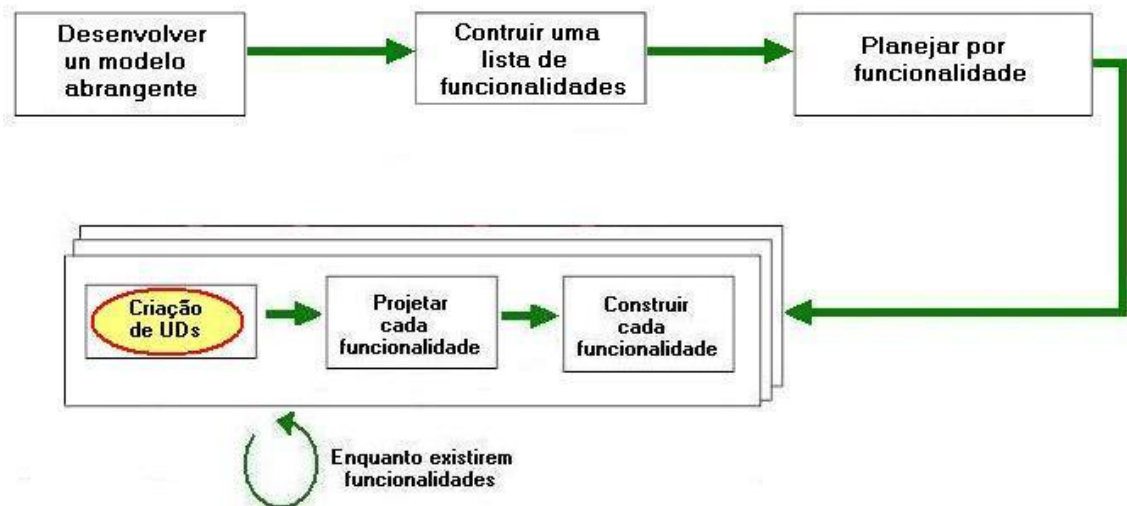


Figura 9. Fases do Processo do FDD (Adaptado de [6]), com inclusão de UIDs

A modelagem dos UIDs no FDD pode ser realizada pelo programador chefe, que participa na análise dos requisitos, ou pelo proprietário de classe, subordinado do programador chefe, que tem como tarefas projetar, codificar, testar e documentar.

Para esse método a prática chamada modelagem de interação entre usuário e sistema seria criada e adicionada ao conjunto de práticas existentes. Aqui seriam modeladas as funcionalidades com grande troca de informações, a serem usadas na seguinte iteração.

4.4 Inclusão de UIDs no DSDM

No DSDM a inclusão dos UIDs pode ser feita dentro da fase do projeto, na etapa de modelagem funcional, quando são desenvolvidos os protótipos que serão mostrados ao cliente, conforme mostrado na figura 10. Sempre que os requisitos selecionados na iteração tratem de interações entre o usuário e o sistema, os UIDs deverão substituir os protótipos.

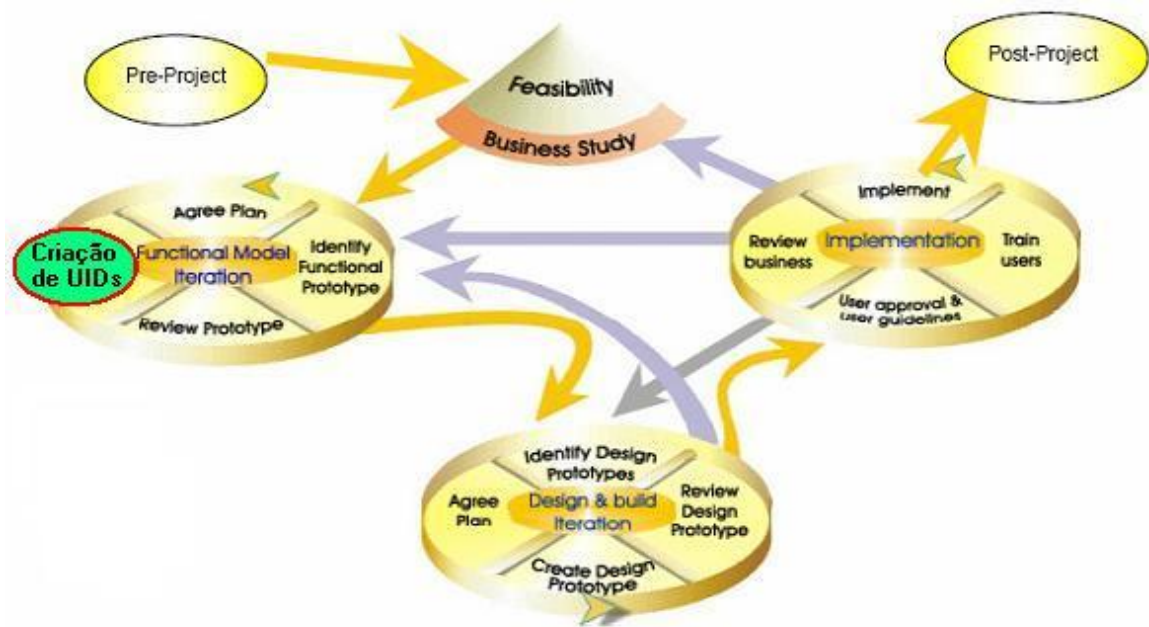


Figura 10. Processo de desenvolvimento do DSDM, com inclusão de UIs (Adaptado de [9])

A modelagem dos UIs nesse método pode ser realizada pelo gerente do domínio.

Nesse caso, a prática de prototipagem seria substituída pela modelagem da interação entre usuário e sistema.

5. ESTUDO DE CASO

Foram desenvolvidas duas aplicações para analisar a modelagem da interação entre o usuário e o sistema, através da inclusão dos UIDs nos Métodos Ágeis. A primeira é um sistema de vendas de uma fábrica de café. A segunda é um sistema complementar ao primeiro, mas direcionado à parte contável e administrativa, levando em consideração as receitas e despesas da empresa, para fazer o balanço do lucro.

A primeira aplicação foi desenvolvida usando os métodos ágeis FDD e DSDM, a segunda usando os métodos XP e Scrum. Cada aplicação foi dividida em quatro iterações, sendo que duas delas eram desenvolvidas usando o método ágil original e as outras duas usando o método ágil estendido com a modelagem da interação entre usuário e sistema através da utilização dos UIDs.

As duas aplicações foram realizadas por um único desenvolvedor, tendo somente um cliente em cada aplicação, que era quem tinha o contato contínuo com o desenvolvedor, e, ocasionalmente, tinha outro usuário pertencente à empresa (cliente) que testava a versão funcional do software depois de cada iteração. Todas as funcionalidades foram requeridas pelo cliente.

No desenvolvimento dessas aplicações não foram aplicadas as práticas que precisavam mais do que uma pessoa na equipe, entre elas podemos citar as reuniões diárias do Scrum, a programação em pares do XP, as equipes por funcionalidade do FDD e o workshop do DSDM. Porém, isso não afetou o objetivo desse trabalho, pois a modelagem da interação é feita na parte de análise e planejamento do projeto, e essas práticas são usadas principalmente na parte de desenvolvimento.

5.1. Aplicação: Sistema de Vendas Fábrica de Café

5.1.1. Visão

É proposto o desenvolvimento de um sistema de controle de vendas de uma fábrica de café, que vai informatizar os pedidos, com as suas respectivas ordens de processamento, venda e pagamento do produto. O objetivo do sistema é organizar e contabilizar com maior precisão as quantidades de café compradas em grão verde e vendidas em grão torrado, assim como manter um cadastro dos clientes.

O sistema deverá ser uma aplicação web. Deverão ser gerados relatórios de vendas por cliente, por nome do café, tamanho do café e apresentação (moído ou grão) do café, principalmente por cada mês, mas com opção para colocar data de início e de fim (podendo ser um intervalo menor ou maior de tempo), assim como relatórios das quantidades em estoque tanto de café verde quanto de café torrado. O sistema deverá calcular automaticamente o valor da venda e ter a opção para, posteriormente, indicar que o pagamento da mesma foi realizado. Não é possível trabalhar com créditos e a impossibilidade de efetuar um pagamento deve deixar o cliente marcado, ou seja, na hora de vender mais quilos de café a venda só poderá ser autorizada pelo administrador.

O sistema, basicamente, funcionará na seguinte seqüência: O funcionário insere o pedido do cliente. É gerada uma ordem de processamento (com a qual a fábrica preparará o pedido), diminuindo o estoque de café verde e aumentando o de café torrado, e é emitida a fatura do produto, que será entregue ao cliente junto com o pedido. Depois disso será feito o pagamento, que será ingressado no sistema.

O programa terá como usuários o administrador (Gerente) e o funcionário, que deverão estar cadastrados no sistema. O cliente não tem acesso ao sistema e deverá ser cadastrado pelo funcionário antes do primeiro pedido. Cada cliente terá um único código de identificação que será seu CPF ou CNPJ.

5.1.2. Funcionalidades a serem implementadas no sistema usando FDD e DSDM

A seguir estão definidas as funcionalidades da primeira aplicação, e as iterações que serão desenvolvidas.

5.1.2.1.FDD

1ª iteração sem UID

- Inclusão e exclusão de cadastro
- Inclusão e exclusão de café verde e de novo estoque de café verde.

2ª iteração com UID

- Inclusão de novo tipo café torrado e de novo estoque
- Exclusão de tipo de café torrado.

5.1.2.2.DSDM

3ª iteração com UID

- Registro de pedido, emissão de ordem de processamento e venda dos diferentes cafés, nas diferentes apresentações e tamanhos.
- Registro do pagamento

4ª iteração sem UID

- Inclusão de funcionário e administrador
- Relatório de estoque e vendas

5.1.3. Desenvolvimento do sistema usando FDD

De acordo com o processo do FDD, antes de começar todas as iterações são executadas as etapas de concepção e planejamento, que inclui as seguintes fases: Desenvolver um modelo abrangente; Construir a lista de funcionalidades e Planejar por funcionalidade. Em seguida, para cada iteração é executada a construção, que inclui as fases: Projetar cada funcionalidade e Construir por funcionalidade.

No desenvolvimento usando o FDD algumas práticas não foram realizadas, pelo fato de se ter só uma pessoa desenvolvendo o método, ou por se tratar de um projeto pequeno.

- Propriedade individual da classe: No caso dessa prática todas as classes pertenceram a uma pessoa só, o desenvolvedor.
- Equipes de funcionalidades: Não teve equipes.
- Administração de configuração: Não se teve um sistema de controle de versões. Porém, se percebeu a necessidade do mesmo, embora tenha sido um projeto pequeno.
- Relatório de resultados: Não foi realizado de maneira formal, mas o cliente foi informado dos resultados do projeto ao final de cada iteração. Não teve relatório mostrado para equipe.

5.1.3.1. Desenvolver um modelo abrangente

A principal tarefa nesta etapa é a modelagem do domínio da aplicação, construção do diagrama de classe UML, diagrama(s) de seqüência UML e uma lista informal das funcionalidades (Figuras 11 e 12).

5.1.3.1.1. Casos de Uso

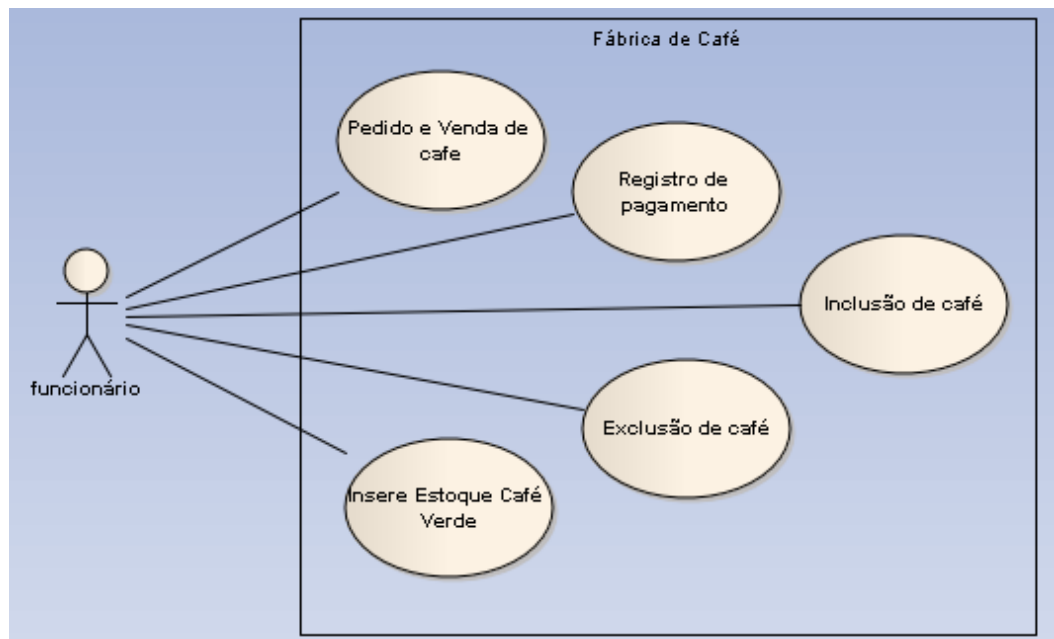


Figura 11. Casos de Uso – Sistema de Vendas Fábrica Café

5.1.3.1.2. Diagrama de classes Inicial

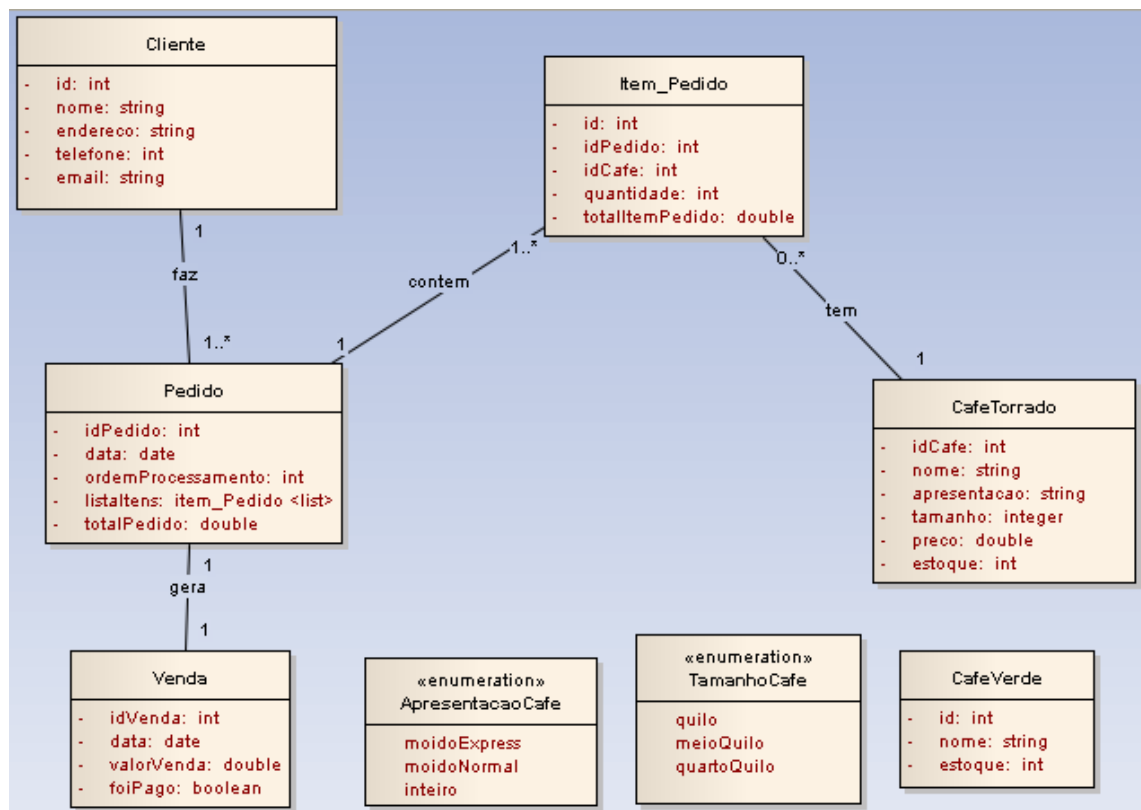


Figura 12. Diagrama de classes Inicial

5.1.3.1.3. Diagramas de Sequência

A seguir o diagrama de seqüência do Cadastro de Cliente.

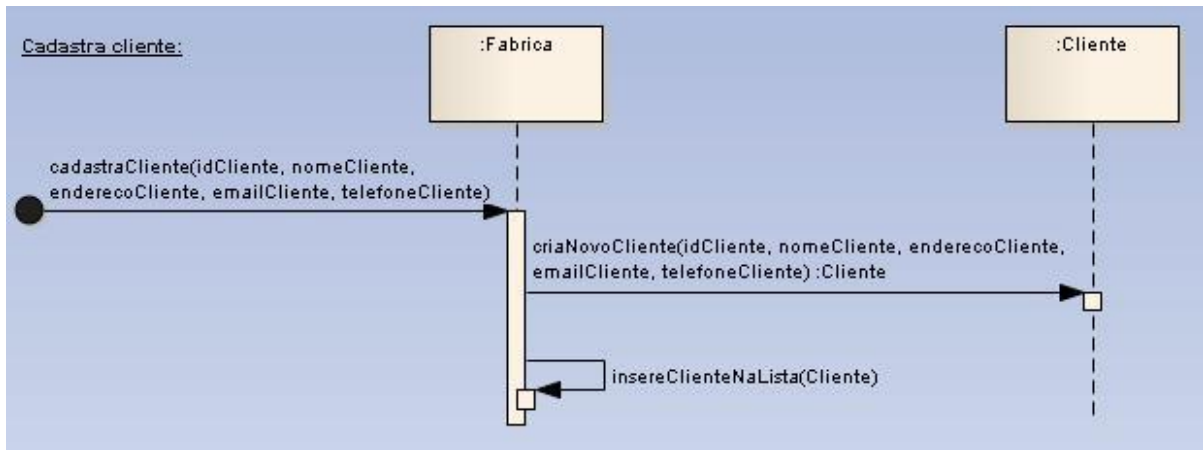


Figura 13. Diagrama de Seqüência – Cadastro de cliente

Outros diagramas de seqüência podem ser encontrados no ANEXO 1 (Figuras 29, 30, 31, 32, 33, 34).

5.1.3.2. Construir a lista de funcionalidades

Nesta fase, a lista principal de funcionalidades é dividida em listas menores que são agrupadas. O diagrama de classes inicial é completado com os métodos e atributos que faltarem.

5.1.3.2.1. Lista de funcionalidades

- Inclusão e exclusão de cadastro
- Inclusão e exclusão de café verde e de novo estoque de café verde.
- Inclusão de novo tipo café torrado e de novo estoque
- Exclusão de tipo de café torrado.

5.1.3.2. Diagrama de classes melhorado

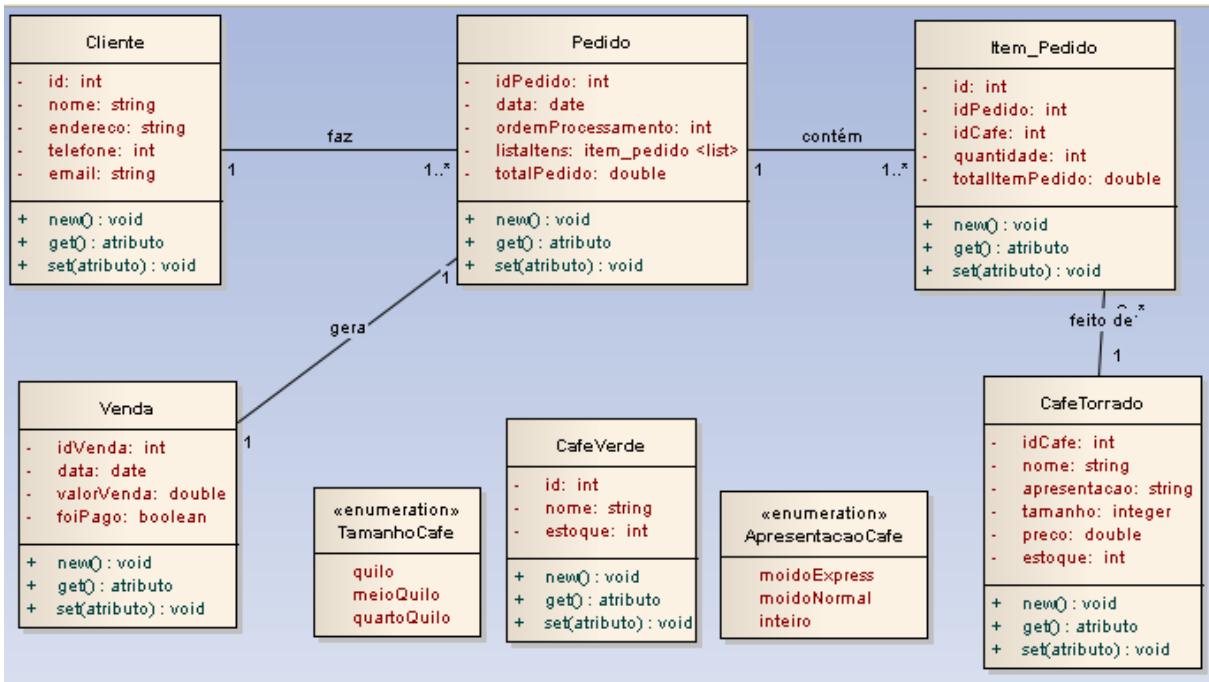


Figura 14. Diagrama de classes com os métodos.

5.1.3.3. Planejar por funcionalidade

Por ser um sistema simples, com poucas funcionalidades, e desenvolvido por somente uma pessoa, não precisou fazer um planejamento para cada funcionalidade.

5.1.4. 1ª Iteração (FDD sem UID)

Nessa iteração foram desenvolvidas as duas primeiras funcionalidades do grupo, as outras ficaram para a próxima iteração:

- Inclusão e exclusão de cadastro de cliente
- Inclusão de café verde e de novo estoque de café verde.

5.1.4.1. Detalhar por funcionalidade

Nessa fase, é estudada a documentação existente, desenvolvido o diagrama de seqüência para as funcionalidades, é refinado o modelo de objetos, são reescritas as classes e os métodos e é feita a inspeção do projeto.

Novamente, por ser um projeto pequeno, não precisou fazer todos os passos. Foi estudada a documentação, foram usados os diagramas de seqüência feitos em uma das fases anteriores, o modelo de objetos foi mantido e foram adicionadas ao projeto as classes de acesso ao banco de dados, comumente chamadas de mapeadores.

5.1.4.2. Construir por funcionalidade

Essa é a etapa de implementação, as classes e métodos são implementados, são realizados alguns testes de unidade e é feita a entrega dessa parte do sistema para o cliente.

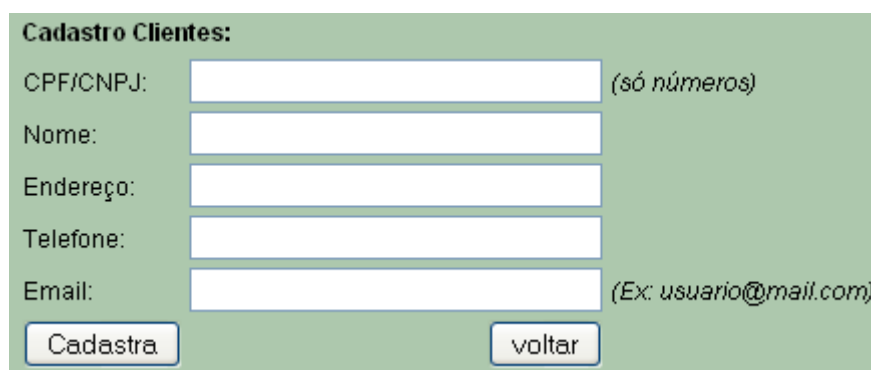
5.1.4.2.1. Implementação

As funcionalidades implementadas nessa iteração foram a de inclusão e exclusão de clientes e a de inclusão de novo tipo de café verde e novo estoque de café verde.

O código foi feito basicamente em linguagem PHP, com a inclusão de alguns tags HTML. O banco de dados escolhido para o programa foi o MySQL, pois possui os recursos necessários para desenvolver esse tipo de aplicação, além disso, o fato de existir bastante documentação sobre suas características e usos ajuda no desenvolvimento. A linguagem PHP usada foi a versão 5.3.0, e o banco de dados MySQL foi a versão 5.1.36.

As imagens da interface geral do sistema, assim como a de fazer pedido e de diminuir estoque de café torrado podem ser encontradas no ANEXO 1 (Figuras 35, 36 e 37)

A seguir é mostrada a interface do cadastro de cliente.



O formulário de cadastro de clientes possui um fundo verde claro. No topo, o título "Cadastro Clientes:" está em negrito. Abaixo dele, há cinco campos de entrada de texto, cada um com um rótulo à esquerda e uma dica de uso à direita. Os campos são: CPF/CNPJ (com a dica "(só números)"), Nome, Endereço, Telefone e Email (com a dica "(Ex: usuario@mail.com)"). Na base do formulário, há dois botões: "Cadastra" e "voltar".

Figura 15. Cadastro de cliente

Para fazer cadastro só é preciso preencher os espaços em branco com os dados solicitados e clicar no botão correspondente para submeter esses dados.

Cada uma das funcionalidades deve ser executada separadamente. Não pode se cadastrar um cliente ao mesmo tempo em que se cadastra um novo tipo de café ou se insere novo estoque.

Após o clique no botão os dados são adicionados nas tabelas do banco de dados. No ANEXO 1, (Figuras 38 e 39) pode-se ver a inserção de alguns dados no BD.

- **Código**

O código está organizado da seguinte forma: Uma página `index.php`, que instancia a classe `interfaces.class.php` para mostrar os botões que abrem os formulários. Em cada um dos formulários o método POST de passagem de parâmetros envia os dados para a página `.php` que estiver no *action* do tag HTML *form*.

Cada formulário é direcionado a uma página específica, por exemplo:

- Botão Cadastrar Cliente → `cadastroCliente.php`
- Botão Excluir Cliente → `exclusaoCliente.php`
- Botão Cadastrar Café Verde → `cadastroCafeVerde.php`
- Botão Inserir Estoque Café Verde → `estoqueCafeVerde.php`

Em cada uma delas são processados os dados obtidos do *form*, atualizando o banco de dados segundo o que for preciso. Inserindo ou excluindo dados.

- **Classes:**

A classe ‘`Cliente.class.php`’ tem os atributos do cliente (id, nome, endereço, telefone) e os métodos get e set para esses atributos.

A classe ‘`CafeVerde.class.php`’ tem os atributos do café verde (id, nome, estoque) e os métodos get e set para esses atributos.

A classe ‘`Conexao.class.php`’ tem os atributos usuário, servidor, senha e banco de dados, que são necessários para a conexão com o banco de dados, assim como os métodos get e set para esses atributos. Além disso, essa classe tem as funções `conectar()` para fazer a conexão com o banco de dados.

As classes 'ClienteDAO.class.php' e 'CafeVerdeDAO.class.php' são as classes que têm acesso ao banco de dados. Elas inserem ou excluem do banco de dados os dados que são enviados no objeto das classes 'Cliente.class.php' ou 'CafeVerde.class.php', usando os métodos 'salvar(\$objeto)', 'excluir(\$objeto)', 'inserirEstoque(\$objeto)'.

5.1.5. 2ª Iteração (FDD com UID)

Nesta iteração os UIDs foram inseridos no processo do FDD. Como proposto, a inserção dos UIDs no FDD é feita na etapa de 'Construção', antes da fase 'Projetar cada funcionalidade'.

Nessa iteração foram desenvolvidas as seguintes funcionalidades do grupo, que são:

- Inclusão de novo tipo café torrado e diminuição de estoque
- Exclusão de tipo de café torrado.

5.1.5.1. Inserção de UIDs

Para cada funcionalidade, foi definido um UID.

- O UID a seguir é referente à funcionalidade de Cadastro de novo tipo de Café (Figura 16). O usuário insere o id, nome, apresentação e tamanho do café para serem adicionados no banco de dados como um tipo novo. Para que essa funcionalidade seja concluída corretamente o id inserido não deve existir no banco de dados, pois todos os cafés têm id único.

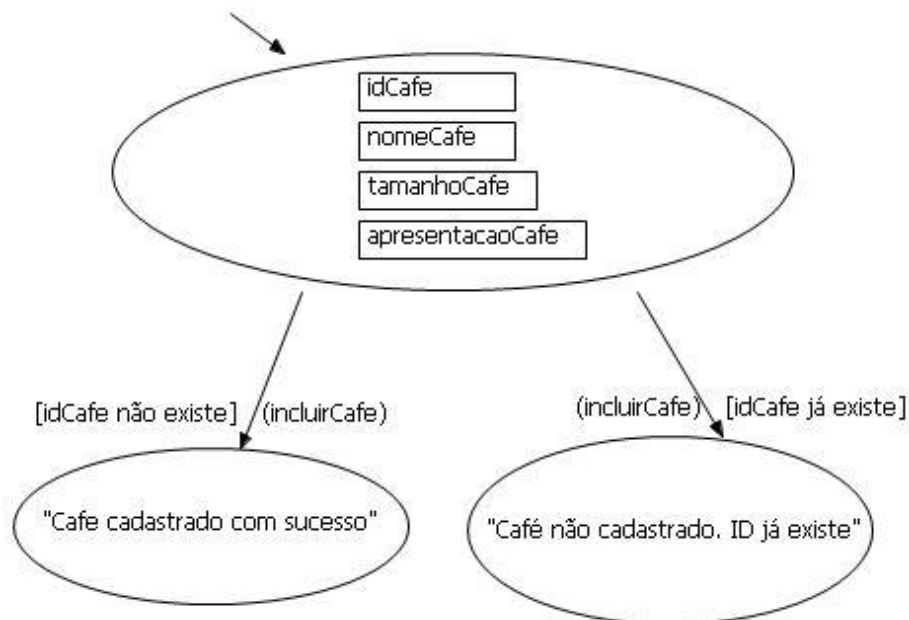


Figura 16. UID Cadastro novo tipo café torrado

- O UID da figura 17 é sobre a funcionalidade de diminuição de estoque de um tipo específico de café. O usuário insere o id e a quantidade a diminuir do estoque, que será subtraída do estoque total, para obter um novo estoque, atualizado. Para que essa funcionalidade seja executada de maneira satisfatória o tipo de café cujo id é inserido deve ter sido cadastrado anteriormente e a quantidade a diminuir deve ser menor ou igual ao estoque total.

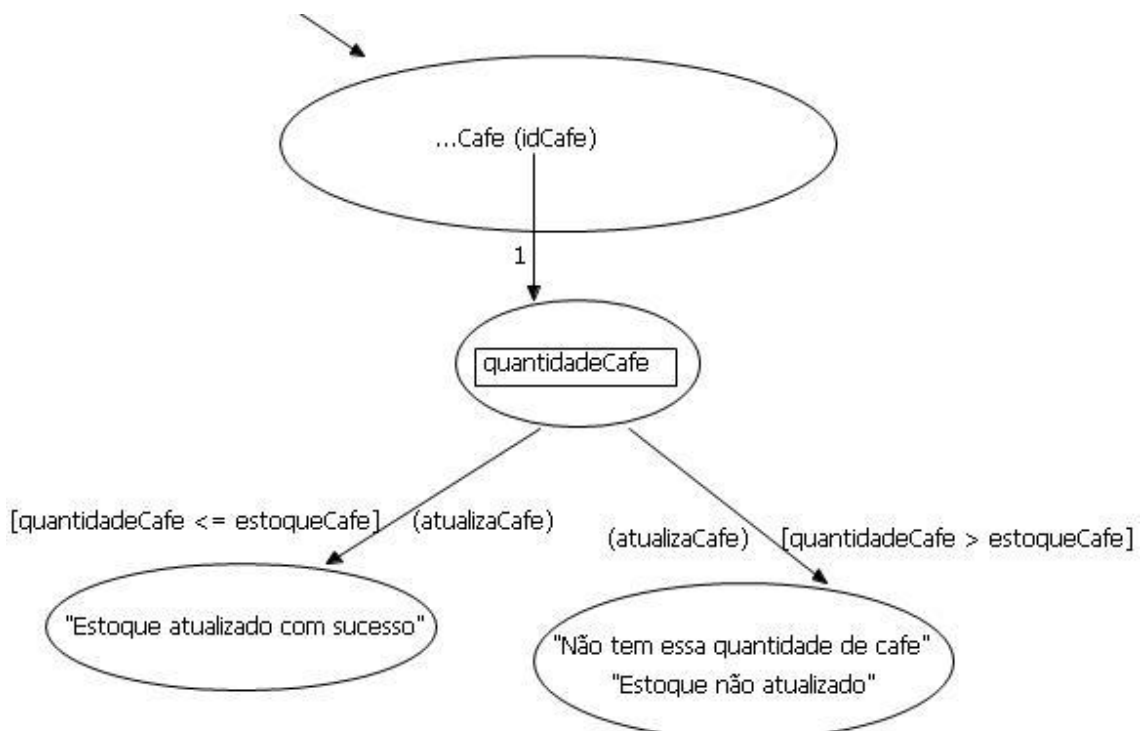


Figura 17. UID Diminuição estoque de café torrado

- O UID da figura 18 se refere à funcionalidade de exclusão de tipo de café torrado. O usuário insere o id do café e, no caso se o estoque desse tipo de café for igual a zero, ele é excluído do banco de dados.

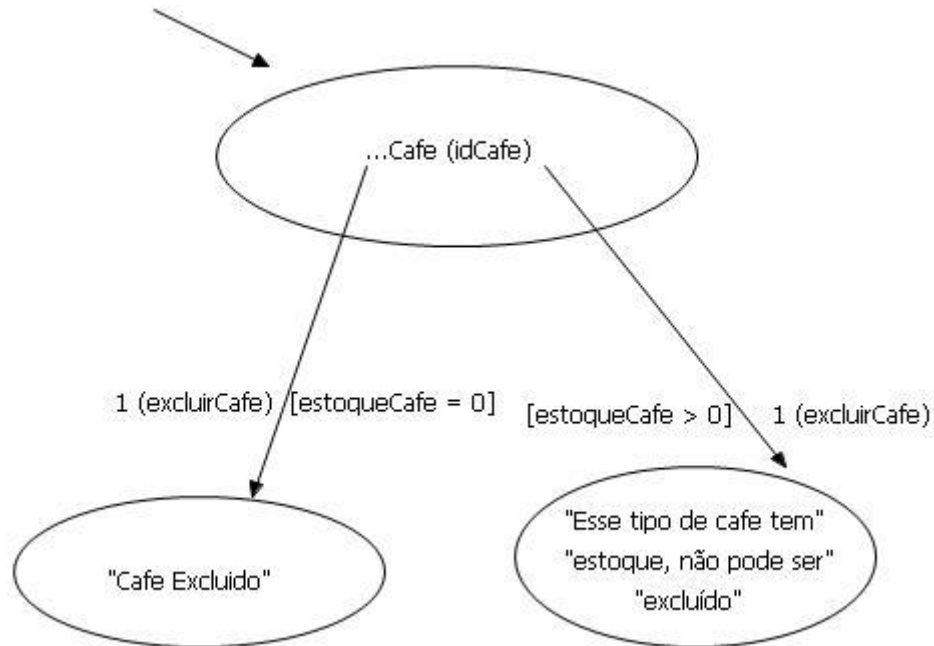


Figura 18. UID Exclusão tipo de café torrado

5.1.5.2. Projetar cada funcionalidade

Nesta iteração, o estudo da documentação, modelo de objetos e diagramas de seqüência mantidos.

5.1.5.3. Construir por funcionalidade

Nessa fase foram implementadas as funcionalidades da segunda iteração (Inclusão de novo tipo café torrado e diminuição de estoque e Exclusão de tipo de café torrado.).

5.1.5.3.1. Implementação

As funcionalidades implementadas nessa iteração foram a de inclusão de novo tipo de café torrado, diminuição de estoque de café torrado, e exclusão de tipo de café torrado.

Para que um tipo de café torrado possa ser excluído o estoque desse café deve ser zero. A diminuição de estoque de café torrado é feita na hora de sair o pedido para ser entregue ao cliente.

As imagens dessa parte do sistema encontram-se no ANEXO 1 (Figuras 40, 41, 42, 43)

- **Código**

O código está separado nas seguintes pastas: ‘adm’, ‘classes’, ‘css’ e ‘dao’. Em ‘adm’ estão as pastas ‘café’ e ‘cliente’, cada uma com os arquivos referentes a elas. Em ‘classes’ estão todas as classes criadas até o momento, em ‘css’ o arquivo de estilos e em ‘dao’ as classes que têm acesso ao banco de dados.

As classes criadas nessa iteração foram: ‘CafeTorrado’ e ‘CafeTorradoDAO’.

5.1.6. Análise da inserção dos UIDs no FDD

Nesse método, os diagramas clássicos, como o de seqüência e o diagrama de classes cobriram todas as necessidades referentes à análise e projeto. Nesse caso os UIDs só foram aproveitados na hora da comunicação com o usuário e não tanto no desenvolvimento do projeto. Porém, pode-se notar a importância da modelagem da interação entre o usuário e o sistema, pois o cliente/usuário conseguiu entender facilmente os diagramas mostrados, o que manteve a comunicação ágil.

Em relação ao tempo não foi feita uma medição exata, mas a inclusão dos UIDs não significou uma diferença notória para o desenvolvedor, cujo papel foi desenvolvido por uma pessoa só, que foi quem assumiu também o papel de responsável pela criação dos UIDs. Não se notou essa diferença, pois os UIDs são diagramas fáceis de desenhar para o desenvolvedor. Além disso, os UIDs são fáceis de entender pelo usuário que terá acesso no momento da comunicação entre ele e o desenvolvedor, o que é uma vantagem, pois a comunicação pode ser mais proveitosa.

5.1.7. Desenvolvimento do sistema usando DSDM

O desenvolvimento de um sistema utilizando o DSDM inclui três fases: pré-projeto, projeto e pós-projeto. E no caso do projeto, serão executadas cinco etapas: estudo de viabilidade, estudo de negócio, modelagem funcional, projeto e construção e implementação. As iterações são executadas dentro das três últimas etapas do projeto. Portanto, as fases de pré-projeto, pós-projeto e as duas primeiras etapas do projeto, apresentadas a seguir, são

executadas somente uma vez, independente de quantas iterações ocorrem durante o desenvolvimento do sistema.

No caso do DSDM, as práticas que não foram realizadas, por ter só um desenvolvedor e ser um projeto pequeno, foram:

- MoSCoW: Sendo uma quantidade pequena de funcionalidades, todas elas foram necessárias (Must have) e implementadas. Por isso não se fez uma priorização usando essa prática.
- Workshop: Não tendo equipe de pelo menos duas pessoas, não foram feitas discussões de workshop.

5.1.7.1. Pré projeto

Na fase de pré-projeto, se faz a identificação do projeto, comprometimento da equipe, plano inicial para estudo de viabilidade e cálculo do orçamento.

A identificação do projeto foi feita na hora de definir a aplicação e as funcionalidades a serem desenvolvidas usando o DSDM. O plano para estudo de viabilidade não foi realizado, pois o sistema era simples e não requeria um estudo de viabilidade. O cálculo do orçamento não foi realizado, pois por ser um projeto sem custo, não era necessário.

5.1.7.2. Projeto ou Ciclo de vida do projeto

Essa fase contém cinco etapas, as duas primeiras se complementam e depois delas começa o desenvolvimento iterativo e incremental, com as outras três etapas:

5.1.7.2.1. Estudo de viabilidade

O estudo de viabilidade não precisou ser feito, pois o sistema era simples e seria desenvolvido de qualquer maneira para testar a inclusão dos UIDs nele.

5.1.7.2.2. Estudo de negócio

- **Requisitos**
 - Registros do pedido, emissão de ordem de processamento e venda.
 - Relatório de estoque e vendas.

- Registro de pagamento.
- Detalhamento do sistema

O sistema deve registrar os pedidos inseridos pelos funcionários, gerar uma ordem de processamento e emitir uma fatura da venda que o pedido gera. Para realizar o pedido o cliente deve estar cadastrado.

O sistema deve ter a opção de emitir relatórios mensais (ou por períodos escolhidos pelo usuário) dos estoques de café verde e torrado, organizados em ordem de ID, e das vendas, organizados em ordem cronológica.

O sistema deve ter a opção de registrar o pagamento de uma venda, só indicando o número da factura (ID da venda).

5.1.8. 3ª Iteração (DSDM com UID)

Nessa iteração, foram usados os UIDs no lugar dos protótipos, que é uma das práticas no processo desse método ágil. O uso dos UIDs permite ver o comportamento do sistema na interação com o usuário, mesmo sem ter que implementar alguma parte dele.

Nessa iteração foram desenvolvidas as seguintes funcionalidades, com a inclusão dos UIDs:

- Registro de pedido, emissão de ordem de processamento e venda dos diferentes cafés, nas diferentes apresentações e tamanhos.
- Registro do pagamento.

5.1.8.1. Modelagem funcional da 3ª. Iteração

Na proposta original do DSDM, na modelagem funcional são feitos protótipos para mostrar os requisitos funcionais. Os protótipos podem ser feitos com implementação ou modelos de interface. Nessa iteração não foi feita implementação, e os protótipos foram substituídos pelos UIDs. Para cada funcionalidade foi feito um UID, descritos a seguir.

- O UID da figura 19 a seguir mostra a interação do usuário com o sistema na hora de fazer o pedido. Inicialmente o usuário insere o ID do cliente, o sistema verifica se o cliente está cadastrado, se não estiver abre a janela de cadastro, se estiver mostra uma

lista com os cafés para escolher os itens. Depois de escolher todos os itens do pedido é gerada a ordem de processamento e emitida a fatura.

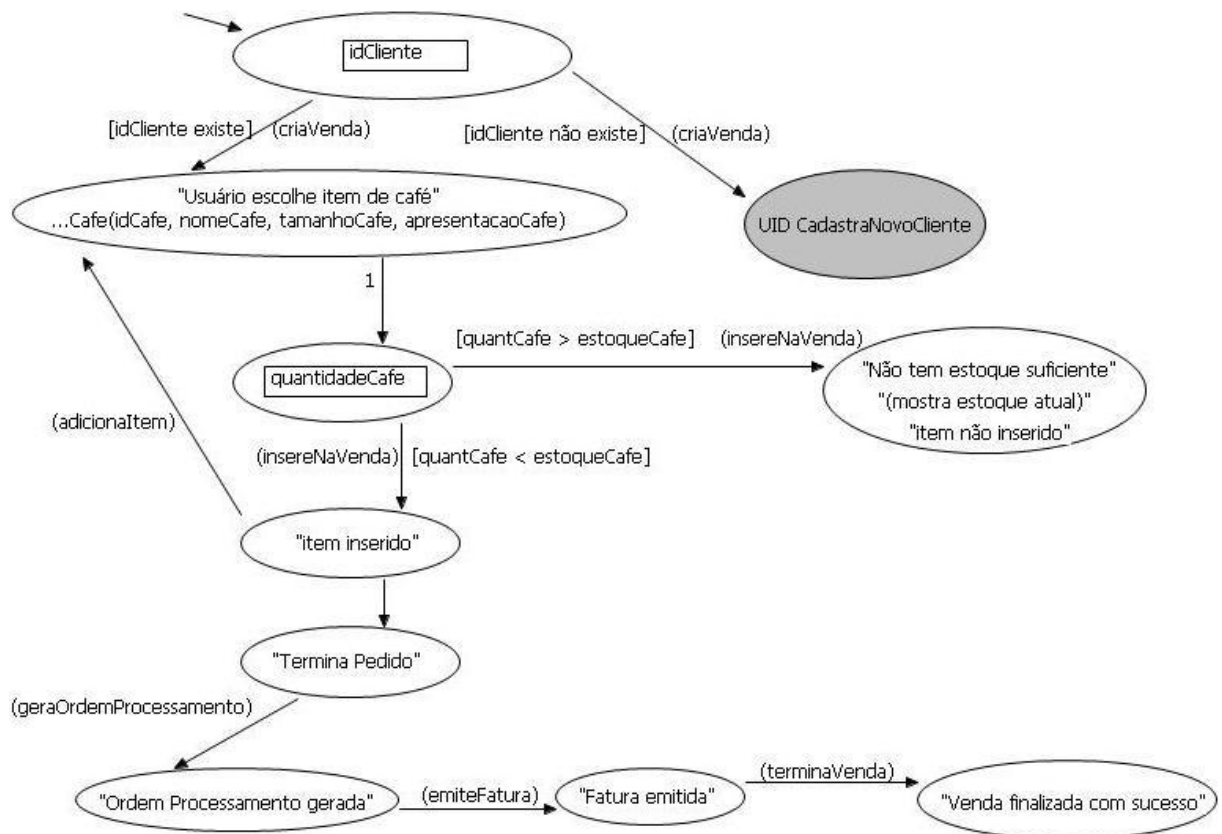


Figura 19. UID Registro de pedido e venda

- O UID da figura 20 mostra a interação entre o usuário e o sistema na hora de registrar o pagamento de uma venda. O usuário tem que inserir o ID da venda (número de fatura) e depois confirmar o pagamento daquela venda.



Figura 20. UID Registrar pagamento.

5.1.8.2. Projeto e construção da 3ª. iteração

Nessa etapa os componentes funcionais da etapa anterior são integrados no sistema. Como não foi feito código na etapa anterior, pois os protótipos foram substituídos pelos UIDs, não foi feita essa integração. A implementação foi feita na seguinte fase.

5.1.8.3.Implementação da 3ª. Iteração

Nessa etapa os protótipos, foram implementados, e entregues aos usuários finais, que são treinados para fazer uso deles. O usuário aprovou o sistema testado.

5.1.8.3.1. Implementação do código

Nessa iteração foram implementadas as funcionalidades registro de pedido, emissão de ordem de processamento e venda, e registro de pagamento.

O registro de pedido gera uma ordem de processamento e uma venda que é colocada em uma fatura em formato PDF. O registro de pagamento é feito para inserir no sistema a confirmação de pagamento de uma dada fatura.

A interface dessa parte do sistema é mostrada no ANEXO 1 (Figuras 44 e 45).

- **Código**

Nessa implementação foi adicionada à pasta ‘adm’ a pasta ‘pedidos’, que inclui a parte dos pedidos, pagamento e criação de fatura.

Foram criadas as classes ‘Pedido’, ‘ItemPedido’ e ‘Venda’, colocadas na pasta ‘classes’ e as classes ‘PedidoDAO’ e ‘VendaDAO’, colocadas na pasta dao, para acesso ao Banco de Dados.

5.1.9. 4ª Iteração (DSDM com UID)

Nessa iteração foram desenvolvidas as seguintes funcionalidades, sem o uso dos UIDs:

- Inclusão de funcionário e administrador
- Relatório de estoque e vendas

5.1.9.1. Modelagem funcional da 4ª. iteração

Nessa iteração não foram usados os UIDs, e sim os protótipos que fazem parte da proposta iteração do DSDM. Como protótipos dessa iteração, para mostrar os requisitos funcionais, foram feitos modelos da interface, que foi implementada na seguinte etapa de desenvolvimento das funcionalidades. Estes protótipos são mostrados no ANEXO 1 (Figuras 46 e 47).

5.1.9.2. Projeto e construção da 4ª. iteração

O código foi implementado nessa fase e entregue ao cliente para realizar os testes e verificar se o que foi feito era o que ele queria.

5.1.9.2.1. Implementação do código

Nessa iteração foram implementadas as funcionalidades ‘Inclusão de funcionário e administrador’ e ‘Relatório de estoque e vendas’.

O cadastro de funcionário e administrador é quase o mesmo que o cadastro de cliente, a única diferença é que tem uma linha a mais onde deve-se colocar a letra A se o funcionário for administrador e F se for só funcionário. Na hora de fazer o protótipo não foi incluída essa última linha.

Os relatórios de estoque foram feitos tanto para o Café Torrado quanto para o Café Verde. As imagens do que foi implementado nessa iteração podem ser vistas no ANEXO 1 (Figuras 48, 49 e 50)

- **Código**

Nessa iteração foi adicionada à pasta ‘adm’ uma pasta de administração (também chamada ‘adm’) na qual foi colocado o arquivo ‘relatorioEstoque.php’.

Foi criada a classe funcionário e adicionada ao banco de dados a tabela do mesmo nome, nessa tabela estarão tanto os funcionários quanto os administradores, que também são funcionários, mas como outras responsabilidades.

5.1.9.3.Implementação da 4ª. iteração

Os protótipos, implementados na etapa anterior, foram melhorados e entregues ao usuário final, que foi treinado para fazer uso deles. O usuário aprovou o sistema testado.

5.1.10. Pós-projeto

Nessa fase, depois de ter terminado as iterações, é realizado o suporte e manutenção do sistema, para que ele continue funcionando.

5.1.11. Análise da inserção dos UIDs no DSDM

Nesse método os UIDs foram mais aproveitados do que no FDD, substituindo os protótipos do sistema, mantendo o processo ágil, pois não foi necessário um protótipo baseado em código. Por se tratar de uma aplicação pequena, o uso de UIDs como protótipos é mais prático. Porém, os UIDs só são indicados para substituir protótipos que mostram a interação entre o usuário e o sistema.

Além disso, os UIDs foram usados na hora da comunicação com o usuário, para mostrar a interação dele com o sistema.

Em relação ao tempo, assim como no método anterior, não se detectou uma demora significativa pelo fato de incluir os UIDs no desenvolvimento, embora não tenha se realizado uma medição exata.

O uso dos UIDs substituindo protótipos de implementação faz bastante diferença no tempo. Em relação a protótipos com interfaces, o uso dos UIDs é mais rápido, mas não muito.

5.2.Aplicação: Sistema administrativo de gastos da empresa

A segunda aplicação desenvolvida para estudar a inclusão dos UIDs nos métodos ágeis foi um Sistema Administrativo para controlar os gastos de uma Empresa. Nesta aplicação foram usados os métodos XP e Scrum, que são os métodos mais conhecidos e usados atualmente. Em algumas empresas eles são usados em conjunto.

As funcionalidades a serem implementadas nessa aplicação foram divididas em quatro iterações, duas para serem desenvolvidas com o método XP (uma iteração utilizando UIDs e outra não) e duas com o método Scrum (uma iteração utilizando UIDs e outra não).

5.2.1. Visão do sistema

Será desenvolvido um sistema para informatizar os gastos de uma empresa, tais como salários dos funcionários, cálculo do seguro de saúde dos funcionários, contas de luz, água, telefone, gastos de insumos, matéria prima, transporte, impostos, perdas na produção, entre outros.

Esses gastos servirão para fazer o cálculo total dos custos de produção da empresa, o que inclui os custos dos produtos a serem vendidos, e da diferença entre uma produção e outra. Deverão ser gerados relatórios dos gastos, agrupados por áreas (gastos na produção, funcionários, gastos externos, etc.) e em alguns casos de forma independente.

Também serão informatizados os valores das vendas da empresa. Com esses valores e os dos gastos se terá uma visão do ganho da empresa em cada mês, da entrada e saída do dinheiro, o que ajudaria no controle futuro do dinheiro investido, tendo uma idéia melhor do que teria que diminuir ou aumentar para que o ganho seja maior.

O sistema será uma aplicação web, com opções de cadastro e inserção de dados. O usuário do programa será o administrador (Gerente) e em alguns casos o funcionário.

5.2.2. Funcionalidades a serem implementadas no sistema usando XP e SCRUM.

A seguir estão definidas as funcionalidades da segunda aplicação, e as iterações que serão desenvolvidas.

5.2.2.1.XP

1ª Iteração

- O usuário deve poder realizar cadastro da folha de pagamento de cada funcionário. Os dados da folha são salário, horas-extras, quantidade de faltas e seguro.
- O usuário deve poder emitir um relatório da folha de pagamento dos funcionários.
- O usuário deve poder atualizar os dados do funcionário e administrador.
- O usuário deve poder excluir os funcionários ou administrador.

2ª Iteração

- O usuário deve poder emitir um relatório das vendas (cada uma aparecendo no relatório), por mês, por ano, ou por tempo definido por ele. Esse relatório deve ter a opção de mostrar uma coluna com o valor referente aos impostos.
- O usuário deve poder cadastrar as despesas de contas fixas, inicialmente luz, água, e telefone.
- O usuário deve poder atualizar os registros das despesas de contas fixas.
- O usuário deve poder emitir o relatório das despesas em contas fixas, por mês, por ano, ou por tempo definido por ele, assim como por tipo de conta.

5.2.2.2.SCRUM

3ª Iteração

- a. Cadastro das despesas com contas variáveis (transporte, impostos).
- b. Cadastro das despesas em produção (matéria prima, perdas em produção, depreciação, insumos).
- c. Emissão do relatório das despesas em contas variáveis, por mês, ano, ou por tempo definido pelo usuário, assim como por tipo de conta.
- d. Emissão do relatório das despesas em produção, por mês, por ano, ou por tempo definido pelo usuário, assim como por tipo de gasto.
- e. Atualização de dados inseridos nos cadastros.

4ª Iteração

- a. Emissão do total de vendas mês a mês, por um tempo determinado pelo usuário.
- b. Emissão do total das despesas mês a mês, de todos os gastos e de maneira independente (fixos, variáveis, produção).
- c. Emissão do total de ingressos menos o total de egressos por um tempo definido pelo usuário.
- d. Emissão do relatório de produção total possível com os recursos que se têm disponíveis no mês. Para conhecer o lucro sem ter gastos adicionais.

5.2.3. Desenvolvimento do sistema usando XP

O processo do XP inclui cinco fases: exploração, planejamento, iterações para entrega, produção e fim do projeto. As duas primeiras são realizadas antes de iniciar as iterações, as seguintes são realizadas dentro de cada iteração.

A parte de implementação no XP foi feita com base nos testes de unidade e aceitação. Uma das características do XP é o refactoring, que é uma limpeza no código para ter uma compreensão melhor do que ele faz, mudando-se a estrutura do código e não a funcionalidade. Nessa aplicação, por ser pequena não foi necessário realizar esta prática. A programação em pares, que é uma prática comum no XP, também não pôde ser realizada no desenvolvimento desse projeto, pois esse trabalho foi feito integralmente por uma única pessoa. No geral, algumas práticas do XP não puderam ser aplicadas pelo mesmo motivo, mas isto não interfere nos resultados do estudo da inclusão de uma modelagem da interação do usuário com o sistema.

Além das práticas já mencionadas estão também:

- Propriedade coletiva: O código foi de propriedade de uma pessoa só, o desenvolvedor.
- Reuniões em pé: Precisa de mais do que uma pessoa para acontecer.

5.2.3.1.Exploração

Nessa fase são escritas as histórias de usuário que serão desenvolvidas nas diferentes iterações.

5.2.3.1.1. Estórias de Usuário

Para esse sistema serão desenvolvidas as seguintes histórias de usuário:

- O usuário deve poder realizar cadastro da folha de pagamento de cada funcionário. Os dados da folha devem ser: salário, horas-extras, quantidade de faltas e seguro.
- O usuário deve poder emitir um relatório da folha de pagamento dos funcionários, com um simples click.
- O usuário deve poder atualizar todos os dados do funcionário e administrador, fazendo a procura pelo CPF, que é o único dado que não pode ser mudado.

- O usuário deve poder excluir os funcionários ou administrador, escrevendo o CPF da pessoa que for excluir.
- O usuário deve poder emitir um relatório das vendas (cada uma aparecendo no relatório), por mês, por ano, ou por tempo definido por ele. Esse relatório deve ter a opção de mostrar uma coluna com o valor referente aos impostos.
- O usuário deve poder cadastrar as despesas de contas fixas, inicialmente luz, água, e telefone.
- O usuário deve poder atualizar os registros das despesas de contas fixas.
- O usuário deve poder emitir o relatório das despesas de contas fixas, por mês, por ano, ou por tempo definido por ele, assim como por tipo de conta.

5.2.3.2.Planejamento

Nessa fase foram definidas as estórias de usuário para a primeira iteração.

As estórias escolhidas são mostradas a seguir:

- O usuário deve poder realizar cadastro da folha de pagamento de cada funcionário. Os dados da folha devem ser: salário, horas-extras, quantidade de faltas e seguro.
- O usuário deve poder emitir um relatório da folha de pagamento dos funcionários, com um simples click.
- O usuário deve poder atualizar todos os dados do funcionário e administrador, fazendo a procura pelo CPF, que é o único dado que não pode ser mudado.
- O usuário deve poder excluir os funcionários ou administrador, escrevendo o CPF da pessoa que for excluir.

5.2.4. 1ª Iteração (XP sem UID)

Nessa iteração foram desenvolvidas as estórias de usuário especificadas na fase anterior de planejamento. E será explicada a fase iterações para entrega, do processo do XP.

São escritos os testes de aceitação e de unidade. No final de cada iteração, feitos todos os testes de aceitação e de unidade, o cliente receberá uma versão nova e funcional do sistema.

5.2.4.1 Testes da 1ª. Iteração

- **Testes de unidade:** No XP são os testes de unidade são feitos pelos desenvolvedores antes da implementação do código do sistema. Os testes de unidade são feitos para testar unidades do programa, que podem ser métodos, funções, ou parte delas.
- **Testes de aceitação:** No XP, são os testes de aceitação realizados sob o olhar dos usuários, para verificar o funcionamento correto da interface do programa, também são feitos antes da implementação. Para a realização desses testes foi utilizado o Plugin do Firefox chamado “Selenium”.

Os testes de unidade e aceitação feitos para essa iteração são apresentados no ANEXO 2.

5.2.4.1.1. Implementação da 1ª. Iteração

Nessa iteração foram implementadas as funcionalidades ‘Cadastro folha de pagamento’, ‘Relatório folha de pagamento’, ‘Atualizar dados de funcionário’ e ‘Excluir funcionário’.

O cadastro de folha de pagamento completa a informação do funcionário no que se refere a salário, seguro, horas extra e faltas ou ausências ao trabalho. No relatório de folha de pagamento aparecem os campos CPF, nome, salário, seguro, horas extras e salário total do funcionário. Não é mostrado o endereço, telefone ou email do funcionário.

Na atualização dos dados é inserido o CPF do funcionário e aparecem para edição todos os dados dele. O CPF não pode ser modificado.

Para exclusão de algum funcionário só é preciso colocar o CPF dele.

Algumas imagens da interface do sistema podem ser encontradas no ANEXO 1 (Figuras 51, 52 e 53)

- **Código**

Foi criada a classe ‘FolhaPagamento’ e os arquivos ‘atualizaFuncionario’, ‘cadastroFolhaPagamento’, ‘exclusaoFuncionario’ e ‘relatorioFolhaPagamento’.

À classe ‘FuncionarioDAO’ foram adicionados os métodos ‘insereFolhaPagamento’, ‘retornaFolhaPagamento’, ‘atualizarFuncionario’, entre outros usados como complemento destes.

- **Resultado dos testes da 1ª. Iteração**

- Teste de unidade: Funcionalidade Cadastro de folha de pagamento – Métodos ‘insereFolha’ e ‘retornaFolha’

O teste foi realizado com sucesso, tanto o salário como as horas extra estavam certos, o valor das faltas era 0, portanto não foi impressa a 3ª mensagem.

Mensagem do sistema: Total Salário do funcionário está certo / Horas extra do funcionário estão certas

- Teste de unidade: Funcionalidade atualização de dados do funcionário - Método ‘retornaFuncionario’

O teste foi realizado com sucesso, encontrando uma diferença de dados no email do funcionário, que propositalmente não coincide com o email inserido ao se fazer o cadastro.

Mensagem do sistema: Nome do funcionário é igual / Endereço do funcionário é igual / Telefone do funcionário é igual / **Email do funcionário não é igual** / Tipo do funcionário é igual.

5.2.5. 2ª Iteração (XP com UID)

Nesta iteração os UIDs são inseridos no processo do XP. Segundo o que foi proposto, essa inserção é feita no início de cada iteração (fase denominada ‘iteração para entrega’), após a fase de planejamento.

Nessa iteração serão desenvolvidas as seguintes histórias de usuário:

- O usuário deve poder emitir um relatório das vendas (cada uma aparecendo no relatório), por mês, por ano, ou por tempo definido por ele. Esse relatório deve ter a opção de mostrar uma coluna com o valor referente aos impostos.

- O usuário deve poder cadastrar as despesas de contas fixas, inicialmente luz, água, e telefone.
- O usuário deve poder atualizar os registros das despesas de contas fixas.
- O usuário deve poder emitir o relatório das despesas em contas fixas, por mês, por ano, ou por tempo definido por ele, assim como por tipo de conta.

5.2.5.1.Inserção de UIDs

- O UID da figura 21 mostra a interação do usuário e o sistema na hora de fazer o cadastro de contas fixas. O usuário seleciona o tipo de conta que quer cadastrar, se não existir a conta na lista ela pode ser adicionada. O fluxo passa para outro UID chamado “Adiciona tipo de conta”. Se existir a conta na lista o usuário escolhe o mês e digita o ano e o valor.

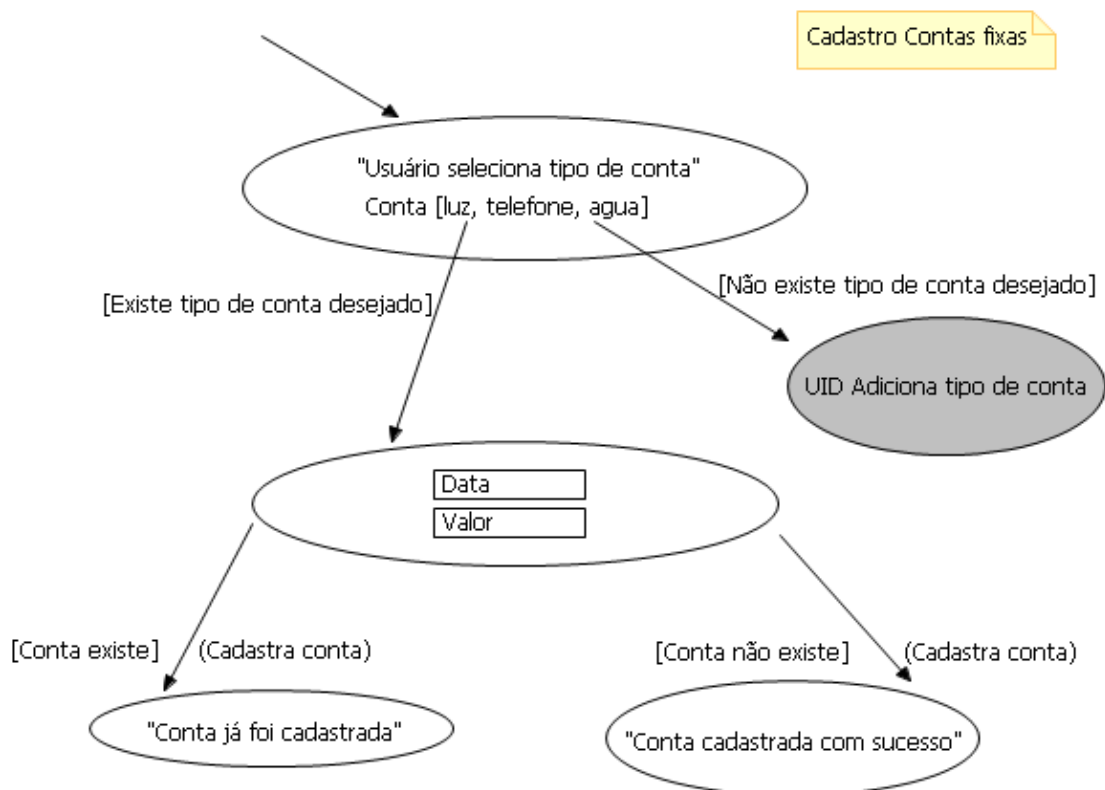


Figura 21. UID Cadastro de contas fixas

- O UID da figura 22 mostra a interação entre o usuário e o sistema para atualizar as contas fixas (luz, telefone, água). O usuário escolhe entre as opções a conta que deseja atualizar, assim como o mês e digita o ano. Se a conta existir, ele digita o valor certo e atualiza.

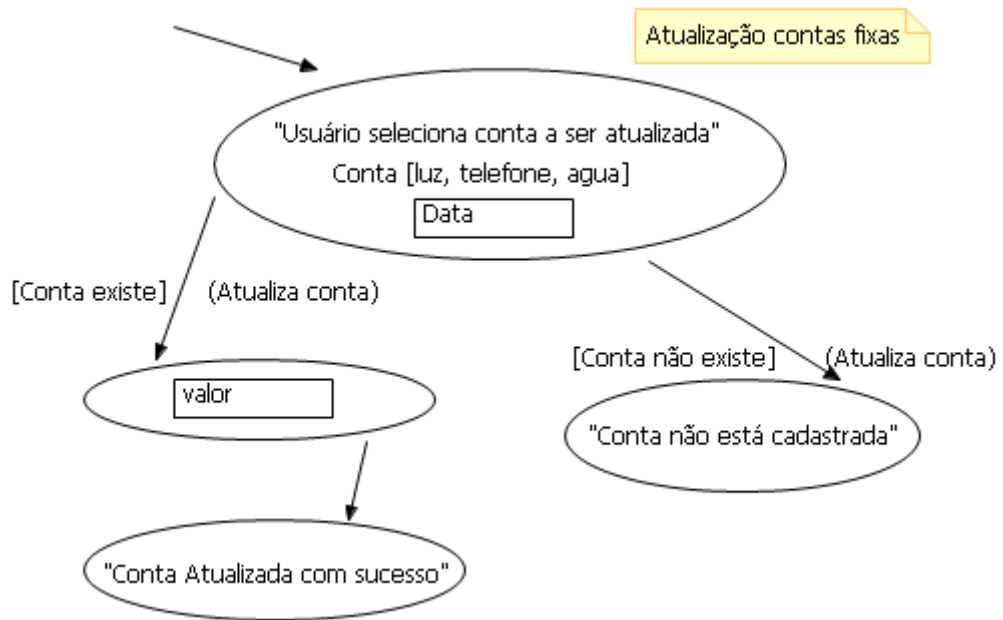


Figura 22. UID Atualização de contas fixas

- O UID da figura 23 mostra a interação entre o usuário e o sistema quando ele quer obter o relatório das contas fixas (luz, água, telefone). O usuário digita as datas de início e fim do relatório e escolhe se quer contas separadas ou não. No caso de escolher contas separadas tem que fazer uma nova escolha, dessa vez da conta que ele desejar. Depois disso o relatório será mostrado.

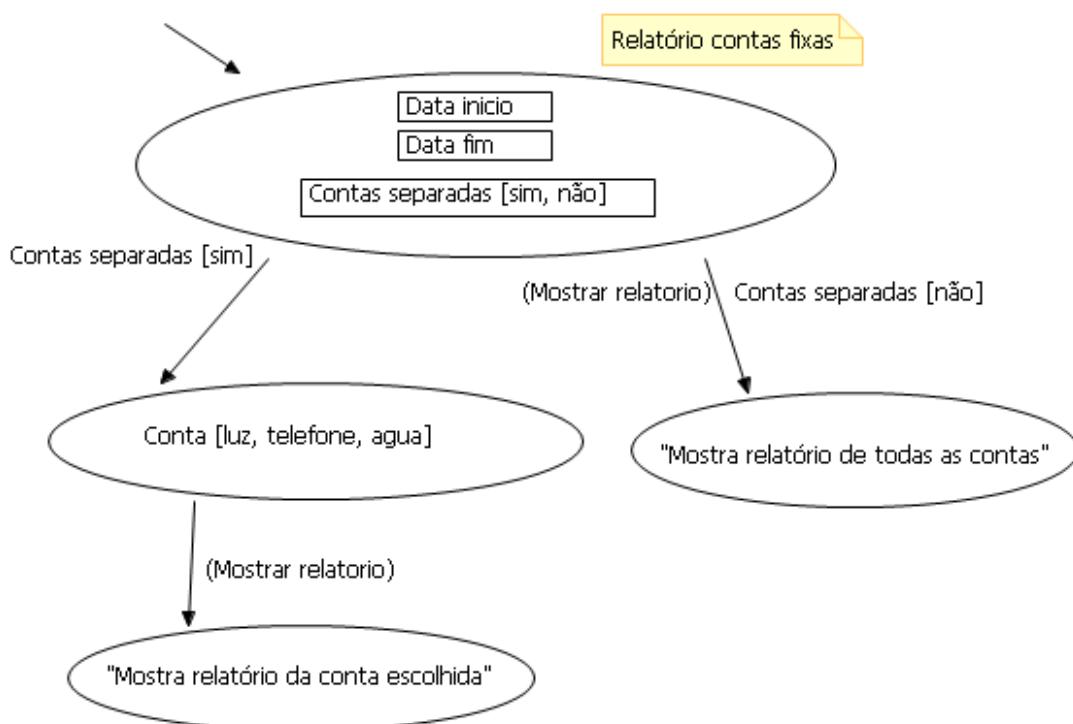


Figura 23. UID Relatório contas fixas

- O UID da figura 24 a seguir mostra a interação do usuário com o sistema na hora de fazer o relatório de vendas. O usuário digita as datas de início e fim do período que desejar e escolhe se quer ou não que apareça a coluna com os impostos das vendas. Feito isso, clica no botão para mostrar o relatório e terá a lista com o que se vendeu no período desejado.

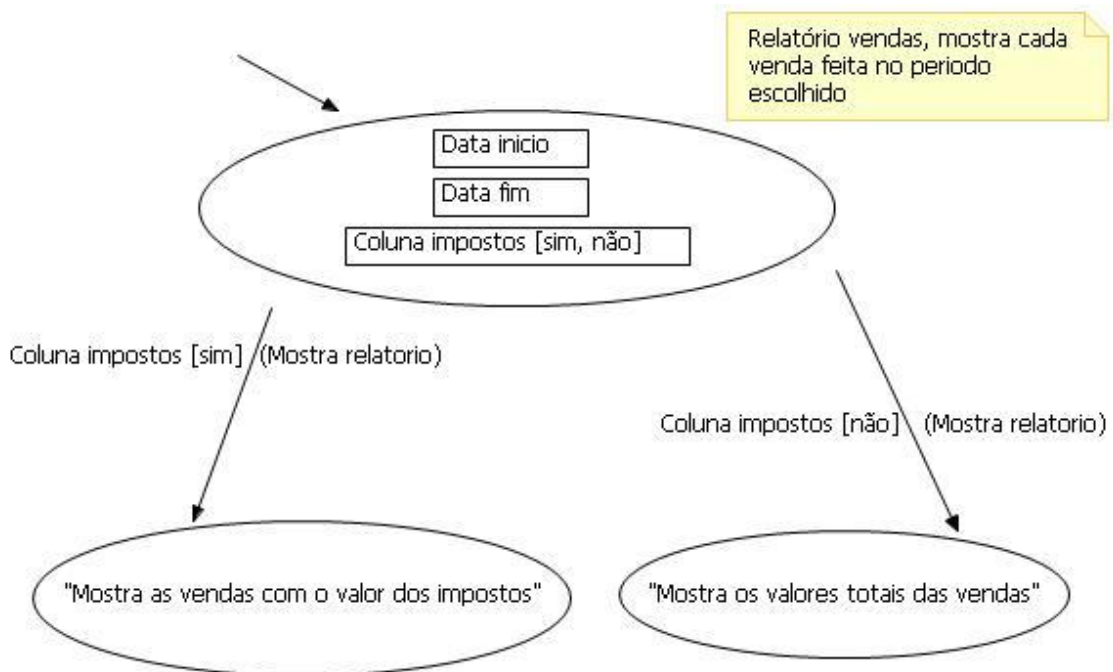


Figura 24. UID Relatório de Vendas

5.2.5.2. Testes da 2ª. Iteração

Os testes de unidade e aceitação escritos para essa iteração podem ser encontrados no ANEXO 2.

5.2.5.3. Implementação da 2ª. Iteração

Nessa iteração foram implementadas as funcionalidades ‘Emitir relatório de vendas com ou sem impostos’, ‘Cadastro de despesas de contas fixas’, ‘Atualização do cadastro de despesas de contas fixas’ e ‘Relatório de despesas de contas fixas’.

No cadastro de despesas o usuário escolhe o tipo de despesa (fixa) e aparece uma relação das contas fixas já cadastradas (telefone, água). No caso de querer adicionar alguma outra despesa, por exemplo, luz, existe a opção de ‘Cadastro de nova despesa’. A atualização

é feita escolhendo, tipo e nome de despesa e digitando a data. Os valores para serem modificados, se precisar, são a data e o valor.

Nos relatórios, tanto o de vendas quanto o de contas fixas são inseridas as datas de início e fim do relatório. No caso das vendas pode-se escolher a opção de mostrar impostos ou não. No caso das contas fixas pode-se escolher entre mostrar o relatório de todas as contas fixas ou mostrar o relatório de cada uma de maneira independente (água, telefone).

Algumas telas da interface podem ser encontradas no ANEXO 1 (Figuras 54, 55 e 56)

- **Código**

Nessa parte da implementação foram adicionados à pasta ‘adm’ interna os arquivos ‘despesa.php’, ‘cadastraDespesa.php’, ‘atualizaDespesa.php’ e ‘relatorioDespesa.php’.

Foram criadas as classes ‘Despesa’ e ‘Tipo’, além da classe ‘DespesaDAO’, que serve para inserir e tirar dados referentes às despesas do Banco de Dados.

- **Resultado dos testes da 2ª. Iteração**

- Teste de unidade: Funcionalidade ‘Atualização de Contas Fixas’: Com erros

O teste foi finalizado com sucesso e não encontrou erros. O valor devolvido pela função é igual ao inserido pelo usuário.

Mensagem do programa: Valor despesa é igual.

- Teste de unidade: Funcionalidade ‘Atualização de Contas Fixas’: Sem erros

O teste foi concluído com sucesso, mostrou a mensagem certa, pois o id da despesa água não é 22, é 20.

Mensagem do programa: O id da despesa está errado.

- Teste de aceitação: Funcionalidade ‘Relatório Contas Fixas’

O teste foi realizado com sucesso e não foram encontrados erros.

Mensagem de aceitação:

- [info] Executing: |type | data | 31/05/2009 |
- [info] Executing: |clickAndWait | procuraDespesa | |
- [info] Executing: |type | valor | 100 |
- [info] Executing: |clickAndWait | atualiza | |
- [info] Executing: |assertTextPresent | **Despesa atualizada com sucesso!** | |

- Teste de aceitação: Funcionalidade ‘Cadastro de despesas’ – Método ‘buscaDespesaGeral’

O teste deu erro, pois a data que foi inserida é inválida.

Mensagem de erro:

- [info] Executing: |type | data | 35/12/2008 |
- [info] Executing: |clickAndWait | procuraDespesa | |
- [info] Executing: |assertTextNotPresent | data inválida | |
- [error] true

5.2.6. Análise sobre a inserção dos UIDs no XP

No XP, que não requer a criação dos diagramas clássicos, e onde são criados testes de unidade e aceitação antes do desenvolvimento do sistema, os UIDs serviram de apoio como entrada para a implementação tanto do código quanto da interface.

Além disso, o XP não possui uma modelagem da interação entre o usuário e o sistema, e por ser um método ágil, onde a comunicação com o usuário é contínua, o seu processo pode ser melhorado com a inclusão de algum tipo de diagrama, que nesse caso foi o UID.

5.2.7. Desenvolvimento do sistema usando Scrum

O processo do Scrum possui três fases que são: PreGame, GamePhase e PostGame. As fases de PreGame e PostGame acontecem, respectivamente, antes e depois das iterações.

Nesse método a prática que não pôde ser realizada por ter somente um desenvolvedor foi a de reuniões diárias do Scrum, pela necessidade de ter mais do que uma pessoa no desenvolvimento.

5.2.7.1.PreGame

Os requisitos foram descritos e priorizados no *Product Backlog* e foi feita uma estimativa de esforço para eles. O *Product Backlog* pode ser constantemente atualizado com novos requisitos.

- **Product Backlog**

- a. Cadastro de despesas com contas variáveis (transporte, impostos) – 3h.

- b. Cadastro de despesas em produção (matéria prima, perdas em produção, depreciação, insumos) – 3h.
- c. Emissão do relatório das despesas em contas variáveis, por mês, ano, ou por tempo definido pelo usuário, assim como por tipo de conta 2h.
- d. Emissão do relatório das despesas em produção, por mês, por ano, ou por tempo definido pelo usuário, assim pelo tipo de gasto 2h.
- e. Atualização de dados inseridos nos cadastros 3h.
- f. Emissão do total de vendas mês a mês, por um tempo determinado pelo usuário.
- g. Emissão do total das despesas mês a mês, de todos os gastos e de maneira independente (fixos, variáveis, produção).
- h. Emissão do total de ingressos menos o total de egressos por um tempo definido pelo usuário.
- i. Emissão do relatório de produção total possível com os recursos que se têm disponíveis no mês. Para conhecer o lucro sem ter gastos adicionais.

5.2.7.2. GamePhase

Nessa fase são realizadas as iterações para o desenvolvimento do sistema.

5.2.8. 3ª Iteração (Scrum sem UID)

Nessa iteração foram selecionadas as funcionalidades a serem desenvolvidas no primeiro *sprint*. Essas funcionalidades estão agrupadas no *Sprint Backlog*, que para essa iteração foi composto por:

- a. Cadastro de despesas com contas variáveis (transporte, impostos) – 3h.
- b. Cadastro de despesas em produção (matéria prima, perdas em produção, depreciação, insumos) – 3h.
- c. Emissão do relatório das despesas em contas variáveis, por mês, ano, ou por tempo definido pelo usuário, assim como por tipo de conta 2h.
- d. Emissão do relatório das despesas em produção, por mês, por ano, ou por tempo definido pelo usuário, assim pelo tipo de gasto 2h.
- e. Atualização de dados inseridos nos cadastros 3h.

O *Scrum* não especifica um método para o desenvolvimento dentro de um Sprint. Neste estudo de caso, para uma melhor compreensão do sistema e suas classes foi feito um diagrama de classes geral (Figura 25), que serviu de guia para a implementação.

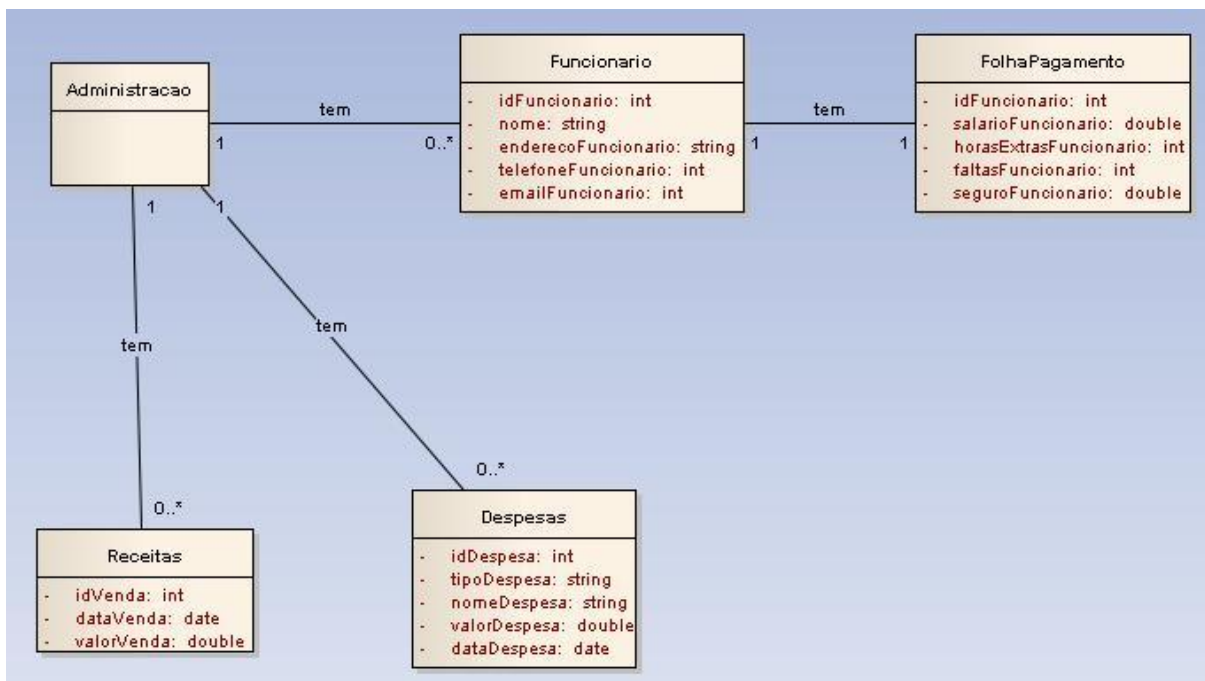


Figura 25. Diagrama de classes – Sistema Administrativo

5.2.8.1.1. Implementação da 3ª. Iteração

A definição do diagrama de classes geral criado e o desenvolvimento das duas primeiras iterações, do mesmo sistema, com o XP, ajudaram na implementação nessa iteração, pois isso permitiu completar algumas classes já criadas. Isto porque as funcionalidades dessa iteração usavam as mesmas classes ou eram parecidas às anteriores.

Nessa iteração foram implementadas as funcionalidades especificadas no *Sprint Backlog*. No cadastro das despesas é escolhido o tipo (variável ou produção) e depois a despesa desejada (luz, telefone, matéria prima), que aparece dependendo o tipo escolhido. Se não tiver a despesa na lista de seleção pode se fazer o cadastro dela, antes de fazer o ingresso da conta no sistema.

Os relatórios são emitidos por tipo de despesa e por tipo de conta independentemente, também pode ser gerado o relatório de todas as contas de um mesmo tipo. Em todos os casos o cliente escolhe a data de início e fim do relatório.

Algumas telas da interface dessa iteração podem ser encontradas no ANEXO 1 (Figuras 57, 58 e 59).

- **Código**

Nessa iteração os arquivos criados anteriormente ‘despesa.php’, ‘cadastraDespesa.php’, ‘atualizaDespesa.php’ e ‘relatorioDespesa.php’, foram completados e modificados com as opções de despesas variáveis e de produção.

Da mesma forma as classes ‘Despesa’, ‘Tipo’ e ‘DespesaDAO’.

5.2.9. 4ª Iteração (*Scrum* com UID)

A inserção dos UIDs no Scrum foi feita no *GamePhase*, depois de ter o *Product Backlog* criado e atualizado, e antes de especificar o *Sprint Backlog*, que é a lista de funcionalidades escolhidas para serem implementadas no *Sprint* seguinte (ou iteração). Os UIDs criados em cada iteração devem coincidir com as funcionalidades definidas no *Sprint Backlog* para essa iteração.

5.2.9.1. Inserção de UIDs da 4ª. Iteração

- O UID da figura 26 mostra a interação entre o usuário e o sistema para obter o relatório dos totais mensais de vendas. O usuário insere a data de início e fim e obtém o relatório das vendas separadas por mês.

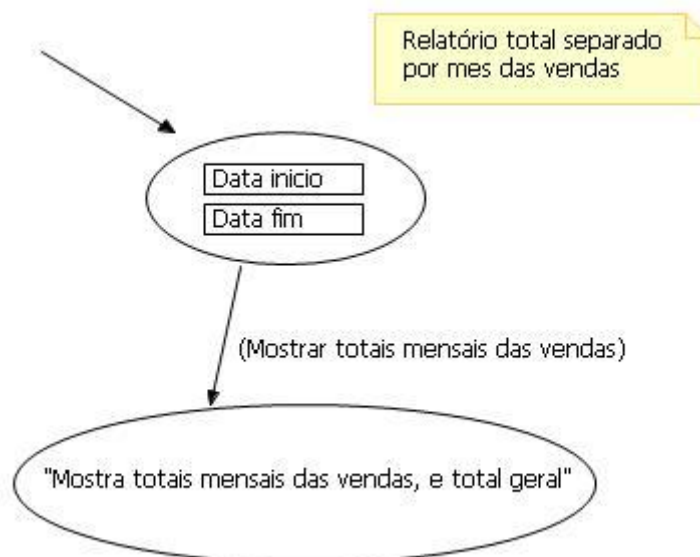


Figura 26. UID Relatório vendas mês a mês

- O UID da figura 27 mostra a interação entre o sistema e o usuário para obter o relatório das despesas de cada mês. Esse relatório pode ser do total de todos os tipos de despesas ou

de cada um deles por separado. O usuário insere a data de início e fim do relatório, escolhe se quer as os tipos de despesas por separado ou não e no caso de ter escolhido para serem separadas tem que fazer uma nova escolha, dessa vez do tipo de despesa que quer (gastos fixos, variáveis ou de produção). O relatório é então mostrado.

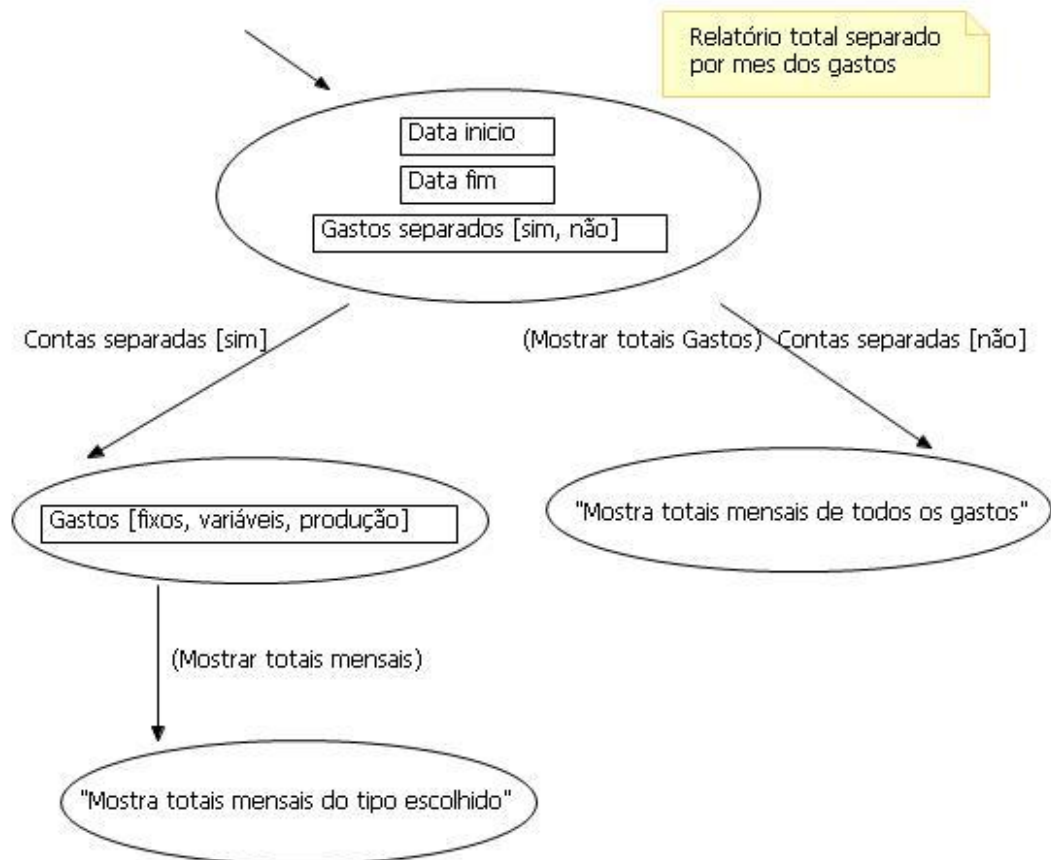


Figura 27. UID Relatório total separado por mês das despesas

- O UID da figura 28 mostra a interação entre o usuário e o sistema para ver o saldo total ou lucro obtido subtraindo o total dos gastos ao total de vendas, por um tempo determinado. O usuário insere a data de início e fim do período que desejar e obtém o valor do saldo.



Figura 28. UID Saldo vendas

As funcionalidades a serem desenvolvidas nessa iteração estão agrupadas no *Sprint Backlog* que está composto por:

- a. Emissão do total de vendas mês a mês, por um tempo determinado pelo usuário.
- b. Emissão do total das despesas mês a mês, de todos os gastos e de maneira independente (fixos, variáveis, produção).
- c. Emissão do total de ingressos menos o total de egressos por um tempo definido pelo usuário.
- d. Emissão do relatório de produção total possível com os recursos que se têm disponíveis no mês. Para conhecer o lucro sem ter gastos adicionais.

5.2.9.2.Implementação da 4ª. Iteração

Nessa iteração a implementação foi menos demorada, pois os UIs facilitaram em parte o entendimento de alguns procedimentos a serem implementados, além do fato de já ter implementado a maior parte do sistema nas três iterações anteriores.

Nessa iteração foram implementadas as funcionalidades ‘Emissão do total de vendas mês a mês, por um tempo determinado pelo usuário’, ‘Emissão do total das despesas mês a mês, de todos os gastos e de maneira independente (fixos, variáveis, produção)’ e ‘Emissão do total de ingressos menos o total de egressos por um tempo definido pelo usuário’.

Nos três casos as variáveis \$dataInicio e \$dataFim definem o período que vai ser usado para calcular os dados. Além disso, na segunda funcionalidade é preciso indicar o tipo de despesa (fixa, variável, produção) que vai ser usada para os cálculos.

Parte da interface dessas funcionalidades pode ser encontrada no ANEXO 1 (Figuras 60, 61 e 62)

- **Código**

Nessa iteração foram criados os arquivos ‘relatorioMensalVendas.php’, ‘relatorioMensalDespesas.php’ e ‘lucro.php’, todos eles na pasta ‘adm’ interna.

Foram usadas as classes ‘DespesaDAO’ e ‘VendaDAO’, criando alguns métodos nelas.

5.2.10. PostGame

O sistema é entregue. Foram realizados os testes e parte da documentação.

5.2.11. Análise da inserção dos UIDs no Scrum

A inserção dos UIDs no Scrum, que também não requer a criação de qualquer diagrama clássico, serviu de apoio para a implementação do código, pois os UIDs simulam a parte da interface.

No caso do Scrum, a inclusão dos UIDs dependerá do processo usado nas iterações. Se for usado um processo que já modela a interação entre o usuário e o sistema, os UIDs não serão necessários.

5.3. Dificuldades na realização do estudo de caso

Entre as dificuldades que se apresentaram no desenvolvimento das duas aplicações estão o pouco conhecimento da linguagem PHP, principalmente no início do desenvolvimento, mas essa dificuldade foi superada no transcorrer do trabalho, mas dificultou a medição de cada iteração. Além disso, a biblioteca PHPUnit, escolhida para a realização dos testes, após várias tentativas de instalação que não deram certo, e de finalmente ter conseguido fazer a instalação, não pôde ser utilizada pois a execução não foi possível. Como a tentativa de usar essa biblioteca tomou muito tempo, se preferiu outra solução.

6. CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho foi feita uma análise da necessidade da modelagem da interação entre o usuário e o sistema nos métodos ágeis, tomando como exemplo os quatro métodos ágeis estudados, que são o FDD, o DSDM, o XP e o Scrum. Para isto foi realizado o estudo de caso com duas aplicações que foram desenvolvidas utilizando estes quatro métodos ágeis juntamente com a utilização dos UIDs. Para esse trabalho os UIDs foram usados como técnica de modelagem, entretanto é importante salientar que outras técnicas poderiam ter sido usadas.

Os métodos ágeis que foram usados são métodos bem diferentes entre si. O FDD é um método um pouco mais parecido com os métodos tradicionais; o DSDM utiliza a prototipação no desenvolvimento do software, prática que foi substituída pela técnica de modelagem da interação; o XP que é mais direcionado para a programação; e o Scrum é direcionado para o gerenciamento e não especifica um processo a ser seguido em cada iteração.

Em todos estes métodos, sentiu-se a falta de uma modelagem que permitisse mostrar ao usuário o comportamento do sistema no momento da interação com ele. E como os métodos ágeis têm entre suas características a comunicação constante e a opção de adicionar, durante o processo, algum requisito que o usuário possa ter esquecido, pode-se notar a importância de uma modelagem para que essa comunicação seja melhorada.

Depois de ter realizado o estudo de caso com as duas aplicações mencionadas, observou-se que os usuários responderam satisfatoriamente quando os UIDs foram apresentados, embora com algumas dúvidas iniciais, mas direcionadas basicamente ao que queriam que fizesse o programa. Eles entenderam o fluxo do diagrama, os estados e transições, e deram opiniões para fazer alguma mudança no projeto, baseados no que estavam vendo no diagrama.

A modelagem da interação, através dos UIDs foi feita para cada funcionalidade ou história de usuário apresentada nas duas aplicações, nas iterações escolhidas, e pode-se notar que no caso de funcionalidades maiores, ou com mais troca de informações, os UIDs tiveram uma participação melhor, já que em pequenas funcionalidades não faz diferença. Em relação ao tempo, o tempo dispendido na criação dos UIDs não significou muito em comparação ao que eles oferecem na comunicação com o usuário.

Outra constatação importante foi que a modelagem da interação foi mais aproveitada nos processos de alguns métodos do que em outros. No FDD, não fez muita diferença a inclusão dos UIDs no processo.

No caso do DSDM, o uso de UIDs foi mais bem aproveitado do que no anterior, pois foi usado no lugar dos protótipos, mantendo o processo ágil, não precisando de um protótipo baseado em código.

No XP, que não requer a criação dos diagramas clássicos, e onde são criados testes de unidade e aceitação antes do desenvolvimento do sistema, os UIDs serviram de apoio como entrada para a implementação tanto do código quanto da interface.

No Scrum, que também não requer a criação de qualquer diagrama clássico, a inclusão dos UIDs facilitou um pouco a implementação do código, pois os UIDs simulam a parte da interface com o usuário. O fato de ter implementado parte do sistema com o XP também ajudou na implementação usando o *Scrum*, pois as iterações feitas com o *Scrum* foram uma continuação do que tinha sido feito com o XP.

Mas independente do método utilizado, no geral, o que os usuários querem é que o programa que eles pediram fique pronto, sem se preocupar muito em como é feito, mas com todas as funcionalidades que pediram funcionando. O uso dos UIDs para mostrar o comportamento do programa é uma maneira simples e direta de se comunicar com o usuário, que em seguida comentará sobre o UID mostrado. Isto se faz de uma maneira rápida, pois o usuário consegue entender o fluxo de dados sem muito problema.

Nos métodos ágeis, onde a relação com o usuário é constante, a inclusão de UIDs melhora a comunicação entre ambas as partes sem aumentar o trabalho significativamente. A criação dos diagramas é simples e pode ser feita na mão, embora exista uma aplicação para criá-los digitalmente, usada para fazer os diagramas deste trabalho.

Portanto, a inclusão dos UIDs pode ser vista como uma prática opcional que pode ser incluída em qualquer método ágil sempre que uma funcionalidade que apresenta grande interação entre o usuário e o sistema estiver sendo desenvolvida. Já em aplicações que tem pouca interação não seriam muito úteis.

Também pode ser comentado que a falta de prática em projetos utilizando um processo ágil possa ter influenciado na análise da modelagem da interação em métodos ágeis. Além disso, o fato de não ter uma equipe para desenvolver os sistemas propostos para o estudo de caso, embora tenham sido pequenos, não permitiu realizar todas as práticas dos métodos, por exemplo, a programação em par e as reuniões diárias. Porém, isso não afetou diretamente o objetivo desse trabalho, portanto, não invalida a análise feita nele, pois os UIDs são usados na parte de análise do projeto, o que não inclui a programação em si, e as práticas ágeis relacionadas com a modelagem da interação foram executadas.

Outro ponto importante a ser comentado é que, por falta de tempo, não foi desenvolvida uma aplicação diferente com cada método ágil original e uma aplicação diferente com cada método ágil estendido com os UIDs. Por isso cada iteração foi desenvolvida com um método diferente. Isto pode ter trazido prejuízos para a análise, pois a partir da segunda iteração, já se tinha um conhecimento maior da aplicação e, por isso, o método ágil utilizado nestas iterações já era utilizado com um conhecimento maior da aplicação. Da mesma forma, as iterações usando o segundo método ágil eram menos complicadas.

Entretanto, depois de ter experimentado o desenvolvimento dos projetos usando métodos ágeis, pode-se confirmar que as vantagens dessa metodologia são muitas, pois a comunicação constante com o usuário permite chegar mais perto do que ele quer, e de uma maneira mais rápida e eficaz, e que a inserção de uma modelagem da interação entre usuário e sistema facilita bastante essa comunicação com o usuário, que compreende melhor o que o sistema vai fazer.

6.1 Trabalhos Futuros

Como trabalhos futuros, podem-se citar a utilização dos UIDs em aplicações com uma equipe de desenvolvimento maior, que tenha algumas experiências anteriores usando os diferentes métodos ágeis no desenvolvimento de projetos, que permita a medição do tempo e a agilidade do processo incluindo os UIDs. E também o uso das extensões no desenvolvimento de aplicações maiores.

7. REFERÊNCIAS

- [1] ADM, Advanced Development Methods. *Scrum It's About Common Sense*. Disponível em: <http://www.controlchaos.com/about/> - Último acesso em 24/05/09.
- [2] AGILE approach. **DSDM – Dynamic Systems Development Methodology**. Disponível em: <http://www.agileapproach.co.uk/html/dsdm.html> - Último acesso em: 27/05/09.
- [3] CUKIER, Daniel. **Metodologias Ágeis de Desenvolvimento**. Disponível em: <http://agilblog.locaweb.com.br/2008/06/20/programacao-extrema-extreme-programming-ou-simplesmente-xp/> - Último acesso em: 17/05/09.
- [4] DOS SANTOS S, Michel. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. Unipac - Universidade Presidente Antônio Carlos Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete.
- [5] EMERY, Patrick. **A term paper prepared for SWE625 software project management and SWE699 software engineering management**. 2002. Disponível em: <http://members.cox.net/cobbler/XPDangers.htm> - Último acesso em: 14/06/09.
- [6] FAGUNDES, Priscila Basto. **Framework Para Comparação e Análise de Métodos Ágeis**. Dissertação de Mestrado, Depto Informática e Estatística, UFSC, 2005.
- [7] GARCIA D.C, Edes. PENTEADO, Rosângela. COUTINHO A.S, Junia. VACCARE B, Rosana T. **Padrões e Métodos Ágeis: agilidade no processo de desenvolvimento de software**. Universidade Federal de São Carlos – Departamento de Computação. Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação.
- [8] HEPTAGON - **FDD - Feature Driven Development**. Disponível em: <http://www.heptagon.com.br/fdd> – Último acesso em 18/03/2009.

- [9] MACHADO, Thiago Leão. **Uma Ferramenta de Suporte ao *Framework* para comparação e Análise de métodos Ágeis**. Trabalho de Conclusão do Curso Sistemas de Informação, Depto Informática e Estatística, UFSC, 2005.
- [10] MANHÃES T, Vinícius. ***Extreme Programming***. Disponível em: <http://improveit.com.br/xp/> - Último acesso em: 17/05/09.
- [11] MANHÃES T, Vinícius. **Scrum**. Disponível em: <http://improveit.com.br/scrum> - Último acesso em: 24/05/09.
- [12] MARCHENKO, Artem. ***AgileSoftwareDevelopment.com***. Disponível em: <http://translate.google.com.br/translate?hl=pt-BR&langpair=en|pt&u=http://agilesoftwaredevelopment.com/xp/> - Último acesso em: 17/05/09.
- [13] MENDONÇA F, Rafael. RIBEIRO B, Marcos J. LEITE B, Vilnei. ARAUJO Z. B, Leonardo. VIANA P, Gabriel. **Desenvolvimento de aplicações WEB utilizando XP, Scrum e Ruby on Rails**. Universidade Federal de Sergipe, Departamento de Computação, Metodologias de Desenvolvimento de Software. Disponível em: <http://www.slideshare.net/MarcosMaozinha/metodologias-ageis-presentation-859575> - Último acesso em: 27/05/09.
- [14] PEDROSO, Ana Elisa da Silva; UESONO, Gabriel; PEGHINI, Lívia; LOPES, Priscila de Abreu. **Métodos Ágeis**. Universidade Federal de São Carlos. 2006.
- [15] PREECE, J.; ROGERS, Y.; SHARP, H.; BENYON, D.; HOLLAND, S.; CAREY, T. **Human-Computer Interaction**. Addison-Wesley, 1994. 775p.
- [16] ROSLINDO K, Giovane. PAMPLONA, Victor F. **Apresentando XP. Encante seus clientes com *Extreme Programming***. Disponível em: <http://www.javafree.org/artigo/871447/#planeja>. – Último acesso em: 17/05/09.

- [17] SANCHEZ, Ivan. **Coding Dojo Floripa Desenvolvimento Ágil**. Disponível em: <http://dojofloripa.wordpress.com/2007/02/07/scrum-em-2-minutos/> - Último acesso em: 24/05/09.
- [18] SAPUTRA, Doddy Ch. **Extreme Programming for Agile Development**. Disponível em: <http://doddychsaputra.thecoderblogs.com/2008/07/19/extreme-programming-for-agile-development/> - Último acesso em: 14/06/09
- [19] SCHÜRHAUS, Sabrina. **Manifesto Para o Desenvolvimento Ágil de Software**. Utah Estados Unidos, 2001. Disponível em: <http://xp.edugraf.ufsc.br/pub/XP/OutrosProcessosAgeis/Metodologiasgeis-Sabrina.ppt> - Último acesso em: 27/05/09.
- [20] TEIXEIRA, Daniel D. AFONSO P, Fernando J. GAIOLAS D. S. P, José P. Gonçalves P. S, Tiago A. **DSDM – Dynamic Systems Development Methodology**. Faculdade de Engenharia da Universidade do Porto. Disponível em: http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_14.pdf - Último acesso em: 27/05/09.
- [21] VILAIN, Patrícia. **Modelagem da interação com o usuário em aplicações hipermídia**. Tese de Doutorado, PUC-RIO, Rio de Janeiro, 2002

ANEXO 1 – Figuras

1. A seguir os diagramas de seqüência do método FDD.

- Diagrama de Seqüência da Exclusão de Cadastro de Cliente.

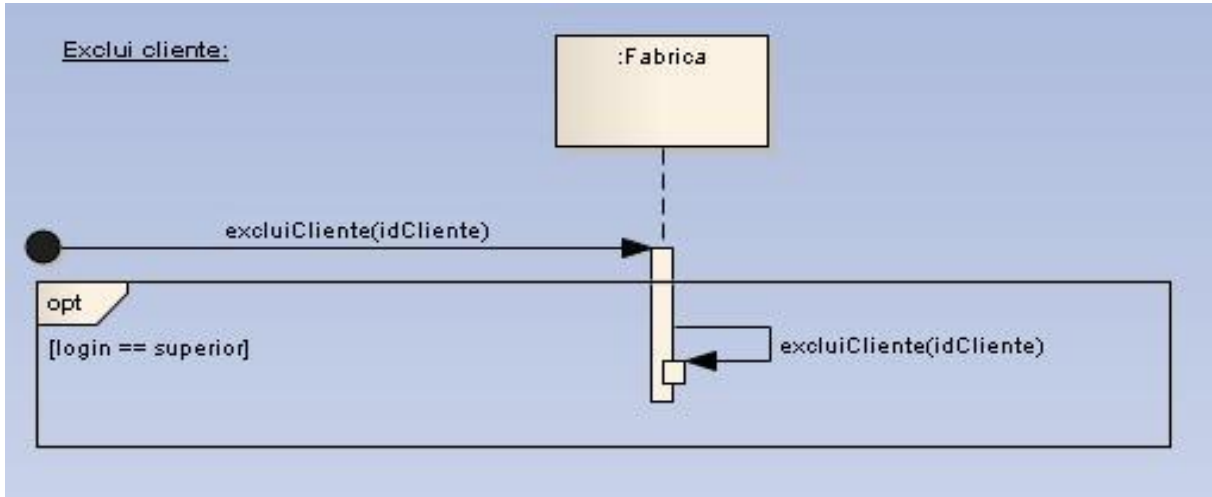


Figura 29. Diagrama de Seqüência – Exclusão de cadastro de cliente

- Diagrama de Seqüência do Cadastro de Café Verde

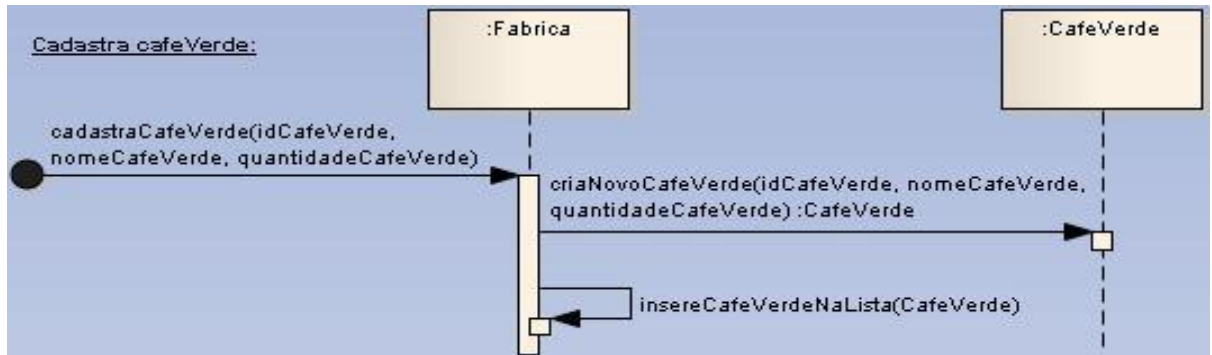


Figura 30. Diagrama de Seqüência – Cadastro de café verde

- Diagrama de Seqüência Insere Estoque Café Verde

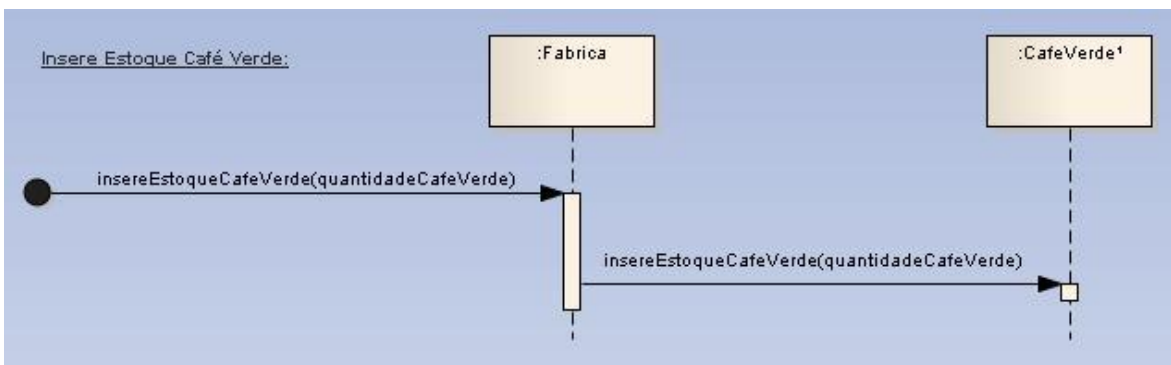


Figura 31. Diagrama de Seqüência – Insere estoque café verde

- Diagrama de Seqüência do Cadastro de Novo Café torrado.

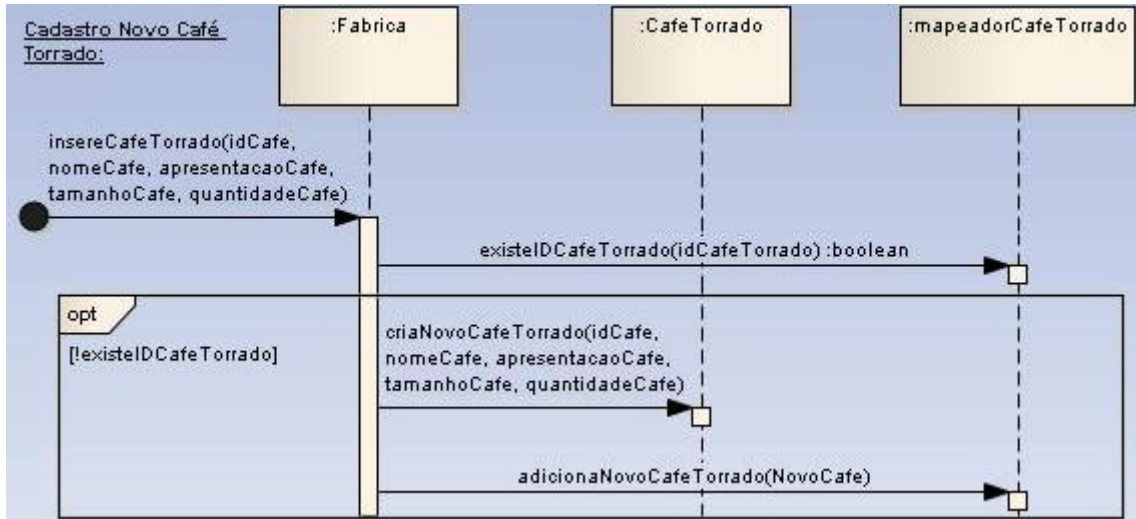


Figura 32. Diagrama de Seqüência – Cadastro novo café torrado

- Diagrama de Seqüência Diminui Estoque Café Torrado

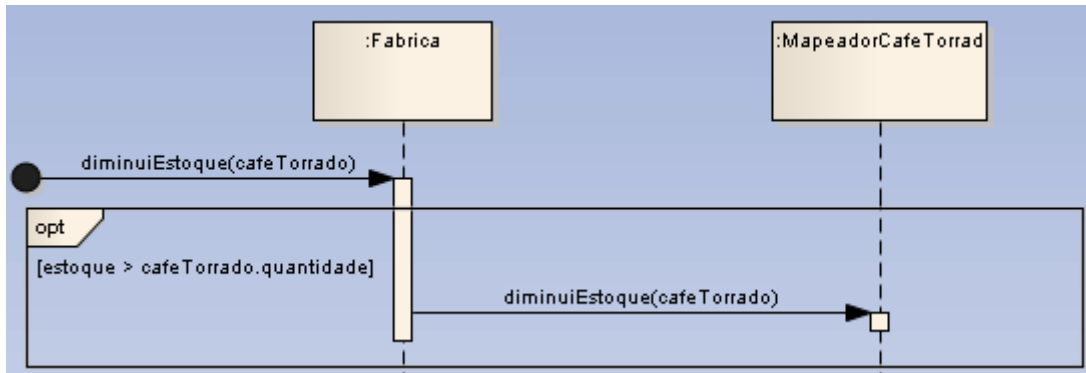


Figura 33. Diagrama de Seqüência – Diminui estoque café torrado

- Diagrama de Seqüência da Exclusão de Café Torrado.

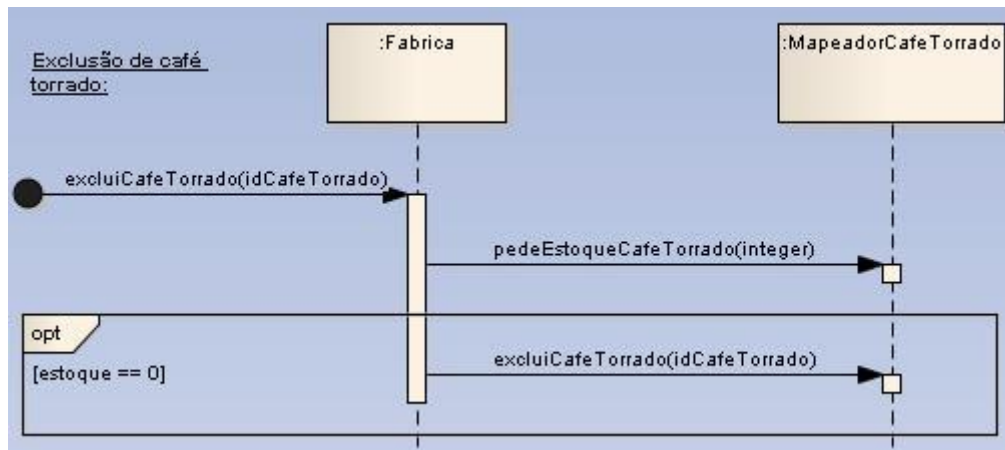


Figura 34. Diagrama de Seqüência – Exclusão de café torrado

2. A seguir as telas da implementação da primeira iteração do FDD, sem UID, assim como as figuras da inserção de dados no BD.

- Interface inicial do sistema

The screenshot shows the main menu of the 'Sistema de Vendas Fábrica de Café'. The title is centered at the top. Below it, there are two columns of buttons. The left column contains: 'Fazer Pedido', 'Confirmar Pagamento', 'Cadastrar Cafe Verde', 'Inserir Estoque Cafe Verde', and 'Excluir Cafe Verde'. The right column contains: 'Cadastrar Cliente', 'Excluir Cliente', 'Cadastrar Cafe Torrado', 'Diminui Estoque Cafe Torrado', and 'Excluir Codigo de Cafe Torrado'.

Figura 35. Interface inicial do sistema

- Fazer pedido

The screenshot shows the 'Item Pedido:' form. It has the following fields and values: 'Cliente:' with the value '123456789'; 'Selecione ID café:' with a dropdown menu showing '-'; 'Nome:' with the value 'Café Chanchamayo'; 'Apresentação:' with the value '1000'; 'Tamanho:' with the value 'Pasar'; and 'Quantidade (em pacotes):' which is an empty text input field. At the bottom, there are three buttons: 'Adiciona item' on the left, 'Termina Venda' in the center, and 'voltar' on the right.

Figura 36. Fazer pedido

- Diminui estoque de café torrado

Diminui Estoque de Café Torrado:

Selecione ID:

Nome:

Apresentação:

Tamanho:

Quantidade (em pacotes):

Figura 37. Diminui estoque de café torrado

- Inserção de cliente no banco de dados

			idCliente	nomeCliente	enderecoCliente	telefoneCliente	emailCliente
<input type="checkbox"/>			1103999941	Augusto Giuffra	Reducto 1525 Miraflores	4451470	agiuffra@hotmail.com
<input type="checkbox"/>			1103999940	Cecilia Giuffra	Rua Luiz Oscar de Carvalho 75 Bloco A2 Apto 32 Tri	99941665	minuska@hotmail.com
<input type="checkbox"/>			1103999942	Gianfranco Giuffra	Reducto 1525 Miraflores	4451470	totti2005@hotmail.com

Figura 38. Inserção de cliente no banco de dados

- Inserção de café verde no banco de dados

			idCafeVerde	nomeCafeVerde	quantCafeVerde
<input type="checkbox"/>			1	chanchamayo	40
<input type="checkbox"/>			2	Los Andes	20

Figura 39. Inserção de café verde no banco de dados

3. A seguir as telas da implementação da segunda iteração do FDD, com UID.

- Cadastro de café torrado

Cadastro Novo Cafe Torrado:

ID Cafe Torrado: (só números)

Nome:

Apresentação: (Expresso, Passar, Grão)

Tamanho (pacotes de): (só números)

Preço: (Ex: 2.00)

Figura 40. Cadastro de café torrado

- Escolha do ID do café a ser excluído.

The screenshot shows a web form titled "Diminui Estoque de Café Torrado:". On the left, there are labels for "Selecione ID:", "Nome:", "Apresentação:", "Tamanho:", and "Quantidade (em pacotes):". Below these is a button labeled "Atualiza Estoque". On the right, there is a dropdown menu currently showing a list of numbers: "-", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "14", and "15". The number "15" is highlighted in blue. To the right of the dropdown are several input fields and a button labeled "voltar".

Figura 41. Escolha do ID do café a ser excluído.

- Diminui estoque de café torrado

The screenshot shows the same form as Figure 41, but now the dropdown menu is closed and the fields are populated. The "Selecione ID:" dropdown shows "-". The "Nome:" field contains "Café 15". The "Apresentação:" field contains "500". The "Tamanho:" field contains "Grano". The "Quantidade (em pacotes):" field is empty. The "Atualiza Estoque" and "voltar" buttons are still present.

Figura 42. Diminui estoque de café torrado

- Exclusão de café torrado

The screenshot shows a form titled "Exclusão de Café Torrado:". At the top right, there is a message: "O cafe ainda tem estoque, não pode ser excluídoO estoque atual é: 53.". Below the message, there is a dropdown menu for "Selecione ID:" showing "-". The "ID:" field contains "10". The "Nome:" field contains "Cafe Gianfranco". The "Apresentação:" field contains "250". The "Tamanho:" field contains "Expresso". At the bottom, there are two buttons: "Exclui" and "voltar".

Figura 43. Exclusão de café torrado.

4. A seguir as telas da implementação da terceira iteração usando o DSDM, com UID.
- Verificando cliente para fazer pedido

Pedido:
CPF/CNPJ Cliente: (só números)

Figura 44. Verificando cliente para fazer pedido.

- Ordem de processamento

Cliente: Nome
Identificação do Pedido: 23
Ordem de processamento: 23
Data do Pedido: 2009-10-12

Id Café	Nome Café	Tamanho Café	Apresentação Café	Quantidade	Preço Unitário	Preço Total Item
9	Café Torrado	500	Grano	10	35	350
8	Descafeinado	1000	Expresso	10	45	450
10	Café Gianfranco	250	Expresso	10	20	200

Total Venda: 1000

Figura 45. Ordem de processamento.

5. A seguir as telas da implementação da quarta e última iteração usando o DSDM, sem UID.
- Protótipo cadastro de funcionários

Cadastro Funcionário:
CPF/CNPJ: (só números)
Nome:
Endereço:
Telefone:
Email: (Ex: usuario@mail.com)

Figura 46. Protótipo cadastro de funcionários.

- Protótipo relatório vendas

Relatório vendas:
Período do 20 ao 25 de Setembro de 2009

Data	Nro Fatura	Valor
20/09/2009	100022	153.00
20/09/2009	100023	22.50
23/09/2009	100024	810.00
23/09/2009	100025	535.50
23/09/2009	100026	63.00
24/09/2009	100027	72.50
25/09/2009	100028	101.00
Total		1757.50

Figura 47. Protótipo relatório de vendas.

- Cadastro funcionário ou administrador

Cadastro Funcionário:

CPF/CNPJ: (só números)

Nome:

Endereço:

Telefone:

Email: (Ex: usuario@mail.com)

Tipo: (F ou A)

Figura 48. Cadastro funcionário ou administrador

- Relatório estoque café torrado

Relatório Estoque:

ID	Nome	Apresentacao	Tamanho	Preço	Quantidade
1	Café Chanchamayo	Pasar	250	R\$ 20	275
2	Café Chanchamayo	Pasar	1000	R\$ 40	212
3	Descafeinado	Pasar	250	R\$ 25	145
4	Descafeinado	Pasar	1000	R\$ 45	178
5	Quillabamba-Villarica-Moyobamba	Expresso	250	R\$ 20	334
6	Quillabamba-Villarica-Moyobamba	Expresso	1000	R\$ 40	64
7	Descafeinado	Expresso	250	R\$ 25	190
8	Descafeinado	Expresso	1000	R\$ 45	167
9	Café Torrado	Grano	500	R\$ 35	81
10	Cafe Gianfranco	Expresso	250	R\$ 20	63

Figura 49. Relatório estoque café torrado.

- Relatório vendas

Relatório Vendas:

Data Inicio:

Data Fim:

Periodo de 14/09/09 até 14/10/09

Data	Nro Fatura	Valor	Foi Paga
14/09/2009	13	R\$ 60	Não
14/09/2009	14	R\$ 40	Não
14/09/2009	15	R\$ 80	Sim
14/09/2009	16	R\$ 20	Não
14/09/2009	17	R\$ 60	Não
15/09/2009	18	R\$ 200	Sim
04/10/2009	19	R\$ 400	Não
10/10/2009	20	R\$ 800	Sim
11/10/2009	21	R\$ 400	Não
12/10/2009	22	R\$ 200	Não
12/10/2009	23	R\$ 1000	Não
12/10/2009	24	R\$ 1000	Não
Total		R\$ 4260	

Figura 50. Relatório vendas.

6. A seguir as telas da implementação da primeira iteração da segunda aplicação, usando o XP, sem UID.

- Interface geral do sistema

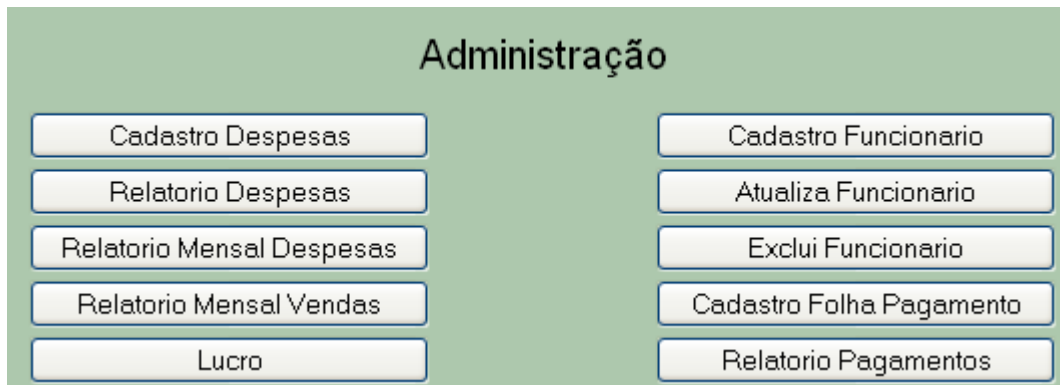


Figura 51. Interface geral do sistema

- Relatório folhas de pagamento dos funcionários

Relatório das folhas de pagamento dos funcionários:

CPF	Nome	Salário	Seguro	Horas Extra	Faltas	Salário Total
01103999940	Cecilia	R\$ 1500	R\$ 195	5	0	R\$ 1770
1234567891	João Perez	R\$ 1200	R\$ 156	0	0	R\$ 1356
1010101010	Maria da Silva	R\$ 2000	R\$ 260	5	0	R\$ 2320

Figura 52. Relatório folhas de pagamento dos funcionários

- Atualiza funcionário

Atualiza Funcionario:

CPF: (só números)

CPF:

Nome:

Endereço:

Telefone:

Email:

Tipo:

Salario:

Horas Extra:

Faltas:

Figura 53. Atualiza funcionário

7. A seguir as telas da implementação da segunda iteração da segunda aplicação, usando o XP, com UID.

- Relatório de vendas com impostos

Relatório Vendas:

Com impostos:

Relatório com imposto:

Data Inicio:

Data Fim:

Periodo de 01/10/2009 até 31/10/2009

Data	Nro Fatura	Valor	Imposto	Valor Total	Foi Paga
04/10/2009	19	R\$ 338.98	R\$ 61.02	R\$ 400	Não
10/10/2009	20	R\$ 677.97	R\$ 122.03	R\$ 800	Sim
11/10/2009	21	R\$ 338.98	R\$ 61.02	R\$ 400	Não
12/10/2009	22	R\$ 169.49	R\$ 30.51	R\$ 200	Não
12/10/2009	23	R\$ 847.46	R\$ 152.54	R\$ 1000	Não
12/10/2009	24	R\$ 847.46	R\$ 152.54	R\$ 1000	Não
18/10/2009	25	R\$ 203.39	R\$ 36.61	R\$ 240	Não
Totais		R\$ 3423.73	R\$ 616.27	R\$ 4040	

Figura 54. Relatório de vendas com impostos

- Cadastro de despesas de contas fixas

Cadastro de despesas:

Selecione o tipo de despesa:

Tipo de Despesa:

Selecione o nome da despesa:

No caso de não ter o nome da despesa desejada, cadastrá-la clicando no botão abaixo:

Figura 55. Cadastro de despesas de contas fixas.

- Relatório das despesas de contas fixas (Todas)

Relatório Despesas:

Selecione o tipo de despesa: -

Tipo de Despesa:

Selecione o tipo de despesa: -

Data Inicio: (dd/mm/aa)

Data Fim: (dd/mm/aa)

Relatório de Despesas no período de 01/10/2009 até 31/10/2009

Código	Data	Nome	Preço
24	10/10/2009	Telefone	15
24	02/10/2009	Telefone	100
24	01/10/2009	Telefone	30
20	15/10/2009	agua	125
20	12/10/2009	agua	23.54
Total			293.54

Figura 56. Relatório das despesas de contas fixas (Todas)

8. A seguir as telas da implementação da terceira iteração da segunda aplicação, usando o Scrum, sem UID.

- Insere despesa variável ao sistema

Cadastro de despesas:

Selecione o tipo de despesa: -

Selecione o nome da despesa: -

Despesa:

Data despesa: (dd/mm/aa)

Valor despesa:

No caso de não ter o nome da despesa desejada, cadastrá-la clicando no botão abaixo:

Figura 57. Insere despesa variável ao sistema.

- Cadastra nova despesa variável

Cadastra nova despesa:

Tipo de despesa: -

Nome da despesa:

Figura 58. Cadastra nova despesa variável.

- Relatório das contas variáveis de transporte

Relatório Despesas:

Selecione o tipo de despesa: -

Tipo de Despesa:

Data Inicio: (dd/mm/aa)

Data Fim: (dd/mm/aa)

Relatório de Despesas no período de 01/10/2009 até 31/10/2009

Código	Data	Nome	Preço
21	01/10/2009	transporte	25
Total			25

Figura 59. Relatório das contas variáveis de transporte

9. A seguir as telas da implementação da quarta e última iteração da segunda aplicação, usando o Scrum, com UID.

- Relatório Vendas mês a mês

Relatório Vendas:

Data Inicio:

Data Fim:

Período de 01/09/2009 até 31/10/2009

Mes	Total
Setembro	R\$ 775
Outubro	R\$ 4040
Total	R\$ 4815

Figura 60. Relatório vendas mês a mês.

- Relatório despesas mês a mês

Selecione o tipo de despesa:

Tipo de Despesa:

Data Inicio: (dd/mm/aa)

Data Fim: (dd/mm/aa)

Relatório de Despesas no período de 01/05/2009 até 31/10/2009

Mes	Total
Outubro	R\$ 45
Agosto	R\$ 143.54
Total	188.54

Figura 61. Relatório despesas mês a mês.

- Cálculo do lucro

Lucro:

Data Inicio:

Data Fim:

Lucro do período de 01/09/2009 até 31/10/2009

Total Vendas	Total Despesas	Lucro
R\$ 4815	R\$ 93.54	R\$ 4721.46

Figura 62. Cálculo do lucro.

ANEXO 2 – Testes de aceitação e unidade

1. Teste de Unidade 1ª iteração XP

- Funcionalidade Cadastro de folha de pagamento – Métodos ‘insereFolha’ e ‘retornaFolha’

```
$id = "1010101010";
$salario = 2000;
$horasExtra = 5;
$faltas = 0;
$seguro = 2000*13/100;
$totalSalario = $salario + (15*$horasExtra) - (15-$faltas) + $seguro;
$folha ->setId($id);
$folha ->setSalario($salario);
$folha ->setHorasExtra($horasExtra);
$folha ->setFaltas($faltas);
$folha ->setSeguro($seguro);
$folha ->setTotalSalario($totalSalario);
$funcionarioDAO->insereFolhaPagamento($folha);
$folha1 = $funcionarioDAO->retornaFolhaPagamento($id);
if ($folha1->getTotalSalario() == $folha->getTotalSalario()){
    echo "<p>Total Salario do funcionario está certo</p>";
}
if ($folha1->getHorasExtra() == $folha->getHorasExtra()){
    echo "<p>Horas extra do funcionario estão certas</p>";
}
if ($folha1->getFaltas() == 2){
    echo "<p>Número de faltas está errado</p>";
}
```

- Funcionalidade atualização de dados do funcionário - Método ‘retornaFuncionario’

```
$id = "1010101010";
$nome = "Maria da Silva";
$endereço = "Rua Falcão 123 Trindade";
$telefone = "48-32347619";
$email = "mds@hotmail.com";
$tipo = "F";
$funcionario->setId($id);
$funcionario->setNome($nome);
```

```

$funcionario->setEndereco($endereco);
$funcionario->setTelefone($telefone);
$funcionario->setEmail($email);
$funcionario->setTipo($tipo);
$funcionarioDAO->salvar($funcionario);
$funcionarioDAO->inicializaFolhaPagamento($funcionario);
$funcionario1 = $funcionarioDAO->retornaFuncionario($id);
if ($funcionario->getNome() == $funcionario1->getNome()){
    echo "<p>Nome do funcionario é igual</p>";
}
if ($funcionario->getEndereco() == $funcionario1->getEndereco()){
    echo "<p>Endereco do funcionario é igual</p>";
}
if ($funcionario->getTelefone() == $funcionario1->getTelefone()){
    echo "<p>Telefone do funcionario é igual</p>";
}
if ($funcionario->getEmail() == "mds1@hotmail.com"){ ← DADO ERRADO
    echo "<p>Email do funcionario é igual</p>";
}else{
    echo "<p>Email do funcionario não é igual</p>";
}
if ($funcionario->getTipo() == $funcionario1->getTipo()){
    echo "<p>Tipo do funcionario é igual</p>";
}
}

```

2. Testes de Aceitação 1ª Iteração XP

- Funcionalidade Cadastro da folha de pagamento

```

class Example extends PHPUnit_Extensions_SeleniumTestCase{
function testMyTestCase() {
$this->open("~/giuffra/fabricaCafe/index.php");
$this->click("//input[@name='enviar' and @value='Cadastro Folha Pagamento']");
$this->waitForPageToLoad("30000");
$this->type("idFuncionario", "1020304050");
$this->type("salario", "2500");
$this->type("horasExtra", "3");
$this->type("faltas", "1");
$this->click("enviaCadastroFolha");
$this->waitForPageToLoad("30000");
$this->assertTrue($this->isTextPresent("Folha cadastrada com sucesso")); }}

```


- Funcionalidade Atualização dados funcionário ou administrador

```

class Example extends PHPUnit_Extensions_SeleniumTestCase{
function testMyTestCase() {
$this->open("~/giuffra/fabricaCafe/index.php");
$this->click("//input[@name='enviar' and @value='Cadastro Despesas']");
$this->waitForPageToLoad("30000");
$this->click("atualizaDespesa");
$this->waitForPageToLoad("30000");
$this->type("id", "1020304050");
$this->click("procuraFuncionario");
$this->waitForPageToLoad("30000");
$this->type("salario", "3500");
$this->click("atualiza");
$this->waitForPageToLoad("30000");
$this->assertTrue($this->isTextPresent("Funcionário atualizado com sucesso"));
}}

```

3. Testes de Unidade 2ª Iteração XP

- Funcionalidade ‘Cadastro de despesas’ – Método ‘buscaDespesaGeral’.

```

$conta = agua;
$data = "15/10/2009";
$valor = "125.00";
$idDespesa = $despesaDAO->buscarIdDespesa($conta);
$despesa->setIdDespesa($idDespesa->getIdDespesa());
$despesa->setDataDespesa($data);
$despesa->setValorDespesa($valor);
$despesaDAO->insereDespesaGeral($despesa);
$despesa = $despesaDAO->buscaDespesaGeral($conta, $data);
if ($despesa->getValorDespesa() == "125.00"){
    echo "Valor despesa é igual";
} else {
    echo "Valor despesa não é igual";
}

```

- Funcionalidade ‘Relatório Contas Fixas’:

```

$conta = agua;
$dataInicio = "01/14/2009";

```

```

$dataFim = "31/10/2009";
$idDespesa = $despesaDAO->buscarIdDespesa($conta);
$listasDespesasGerais[] = $despesaDAO->buscaDespesasGerais(
    $idDespesa->getIdDespesa(), $dataInicio, $dataFim);
if ($idDespesa->getIdDespesa() == "22"){
    echo "O id da despesa está certo";
} else {
    echo "O id da despesa está errado";
}

```

4. Testes de Aceitação 2ª Iteração XP

- Funcionalidade ‘Atualização de Contas Fixas’: Sem erros

```

class Example extends PHPUnit_Extensions_SeleniumTestCase
{
    ...
    function testMyTestCase()
    {
        $this->open("~/giuffra/fabricaCafe/index.php");
        $this->click("//input[@name='enviar' and @value='Cadastro Despesas']");
        $this->waitForPageToLoad("30000");
        $this->click("atualizaDespesa");
        $this->waitForPageToLoad("30000");
        $this->select("selection", "label=Fixa");
        $this->waitForPageToLoad("30000");
        $this->select("//form[@id='relatDesp']/table/tbody/tr[3]/td[2]/
            select", "label=agua");
        $this->waitForPageToLoad("30000");
        $this->type("data", "31/05/2009");
        $this->click("procuraDespesa");
        $this->waitForPageToLoad("30000");
        $this->type("valor", "100");
        $this->click("atualiza");
        $this->waitForPageToLoad("30000");
        $this->assertTrue($this->isTextPresent("Despesa atualizada com sucesso!"));
    }
}

```

- Funcionalidade ‘Atualização de Contas Fixas’: Com erros

```

class Example extends PHPUnit_Extensions_SeleniumTestCase
{
    ...
    function testMyTestCase()

```

```
{
  $this->open("~/giuffra/fabricaCafe/index.php");
  $this->click("//input[@name='enviar' and @value='Cadastro Despesas']");
  $this->waitForPageToLoad("30000");
  $this->click("atualizaDespesa");
  $this->waitForPageToLoad("30000");
  $this->select("selection", "label=Fixa");
  $this->waitForPageToLoad("30000");
  $this->select("//form[@id='relatDesp']/table/tbody/tr[3]/td[2]/select", "label=Telefone");
  $this->waitForPageToLoad("30000");
  $this->type("data", "35/12/2008"); ← ERRO
  $this->click("procuraDespesa");
  $this->waitForPageToLoad("30000");
}
}
```

Modelagem da Interação do Usuário no Desenvolvimento Ágil

Cecilia Giuffra, Patrícia Vilain

Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)

giuffra@inf.ufsc.br, vilain@inf.ufsc.br

Resumo. Dentre os processos de desenvolvimento de software, os métodos ágeis vêm sendo uma opção bastante explorada. Princípios ágeis como o comportamento iterativo e incremental e a entrega de uma versão funcional no final de cada iteração, além da comunicação constante com o usuário facilitam a detecção de erros e a execução de mudanças durante todo o desenvolvimento do software. É importante que essa resposta do usuário, incluindo a descrição de sua interação com o sistema, seja rápida e aproveitada da melhor forma. Para tanto, fazer a modelagem da interação do usuário com o sistema e incluí-la como uma das práticas dos métodos ágeis, pode trazer benefícios na hora da comunicação, e posterior desenvolvimento. Esse artigo faz uma análise de como a modelagem da interação entre o usuário e o sistema é realizada no desenvolvimento ágil atualmente.

Palavras-chave: Métodos Ágeis, Modelagem da Interação, Diagrama de Interação de Usuário, XP, Scrum, FDD, DSDM.

1. Introdução

Os métodos ágeis são processos de software bastante utilizados atualmente. Eles surgiram no final dos anos 90, como uma alternativa aos métodos tradicionais, e propõem agilizar o processo de desenvolvimento de software. Esses métodos possuem em comum o fato de serem aplicados em projetos de baixa complexidade, utilizando ciclos iterativos curtos, planejamento guiado por funcionalidades, retro alimentação constante, tolerância a mudanças, proximidade da equipe, intimidade com o cliente e um foco no ambiente geral de trabalho da equipe [MACHADO 2005]. Entre os métodos ágeis podemos citar: XP (Extreme Programming), Scrum, Crystal, FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method), entre outros.

Esses métodos são caracterizados por repetidas iterações durante o desenvolvimento do software. Cada iteração, que dura entre duas e oito semanas, apresenta a entrega de uma versão funcional para o cliente, onde é informado o que foi feito até o momento, com o intuito de intercambiar opiniões e idéias, tanto entre os programadores e projetistas como entre projetistas e clientes.

Os métodos ágeis oferecem às organizações uma grande variedade de práticas e enfoques para o desenvolvimento de software, propiciando que as organizações escolham o método que melhor supra suas necessidades. Entretanto, apesar destes métodos serem guiados por funcionalidades, onde um conjunto de funcionalidades é desenvolvido a cada iteração, eles especificam estas funcionalidades de diferentes maneiras. Além disso, por não fazer parte dos princípios ágeis, a modelagem da interação entre o usuário e o sistema nem sempre é realizada nos diferentes métodos. Entretanto, para aplicações interativas, que podem ser vistas como aquelas que apresentam interação entre o usuário e o sistema [PREECE 94], a modelagem desta interação poderia facilitar a comunicação entre o usuário e o desenvolvedor.

E apesar dos métodos ágeis serem muito utilizados em aplicações do tipo sistemas de informação que apresentam bastante interação com o usuário, eles não dão ênfase nesta modelagem.

Este artigo apresenta uma análise de como é feita a modelagem da interação entre o usuário e o sistema nestes métodos, e se, para o desenvolvimento de aplicações interativas, existe vantagens na inclusão desta modelagem nos processos destes métodos ágeis.

O artigo está organizado da seguinte maneira. A seção 2 apresenta o processo dos métodos ágeis e a forma como eles fazem a modelagem do usuário com o sistema. A seção 3 apresenta a extensão dos métodos ágeis estudados (FDD, DSDM, XP e Scrum) com a modelagem da interação. E, por fim, na seção 4 são apresentadas as conclusões.

2. Métodos Ágeis e a Interação do Usuário

No geral, os processos dos métodos ágeis estão divididos basicamente em uma parte de planejamento e uma parte iterativa, onde se realiza o desenvolvimento da aplicação em si.

Na parte de planejamento são feitos, geralmente, os estudos do sistema, a documentação necessária, o cálculo dos custos do projeto, a criação das histórias de usuário ou lista de funcionalidades do sistema, entre outros. Na parte iterativa é desenvolvido o projeto, baseando-se na lista de funcionalidades ou histórias de usuário feitas na etapa de planejamento, e em alguns casos, como o XP, são realizados testes de aceitação e unidade antes da implementação.

Todos os métodos tem um conjunto de práticas parecidas, em alguns casos em maior número que em outros, mas no geral todos os métodos ágeis incluem a comunicação constante com o usuário e a entrega de versões funcionais no final de cada iteração. Porém, a interação do usuário com o sistema não é uma prática adotada por esses métodos.

3. Extensão dos Métodos Ágeis

A modelagem da interação do usuário foi inserida no início da parte iterativa de cada um dos processos. A escolha desta posição dentro do processo foi feita porque em cada iteração podem ser adicionados novos requisitos, os quais podem requerer a modelagem da interação para ser mostrada ao cliente antes da implementação. Assim, cada requisito será analisado para verificar se ele tem troca de informação ou não.

A seguir é mostrado como esta nova prática pode ser utilizada no desenvolvimento ágil através da sua inclusão em quatro métodos ágeis distintos.

3.1. Processo FDD

O FDD (*Feature Driven Development*) é um método ágil para gerenciamento e desenvolvimento de software, criado em 1997 por Peter Coad e Jeff De Luca, num grande projeto em Java para o United Overseas Bank, em Singapura. Ele “combina as melhores práticas do gerenciamento ágil de projetos com uma abordagem completa para Engenharia de Software orientada por objetos, conquistando os três principais públicos de um projeto de software: clientes, gerentes e desenvolvedores.” [HEPTAGON]

O processo do FDD está dividido em cinco etapas que são: desenvolver um modelo abrangente, construir a lista de funcionalidades, planejar por funcionalidade, projetar cada funcionalidade e construir por funcionalidade. Dessas etapas as três primeiras pertencem à parte de planejamento do processo e as duas últimas à parte iterativa. Nesse método a prática de modelagem será inserida dentro da parte iterativa, antes do projeto de cada funcionalidade,

conforme mostrado na figura 1. É definido um UID para cada funcionalidade associada com uma grande troca de informações entre o usuário e o sistema. Assim, poderá ter-se uma visão mais clara e específica do comportamento de cada requisito. A modelagem nesse método pode ser realizada pelo programador chefe. Essa prática é adicionada às existentes.

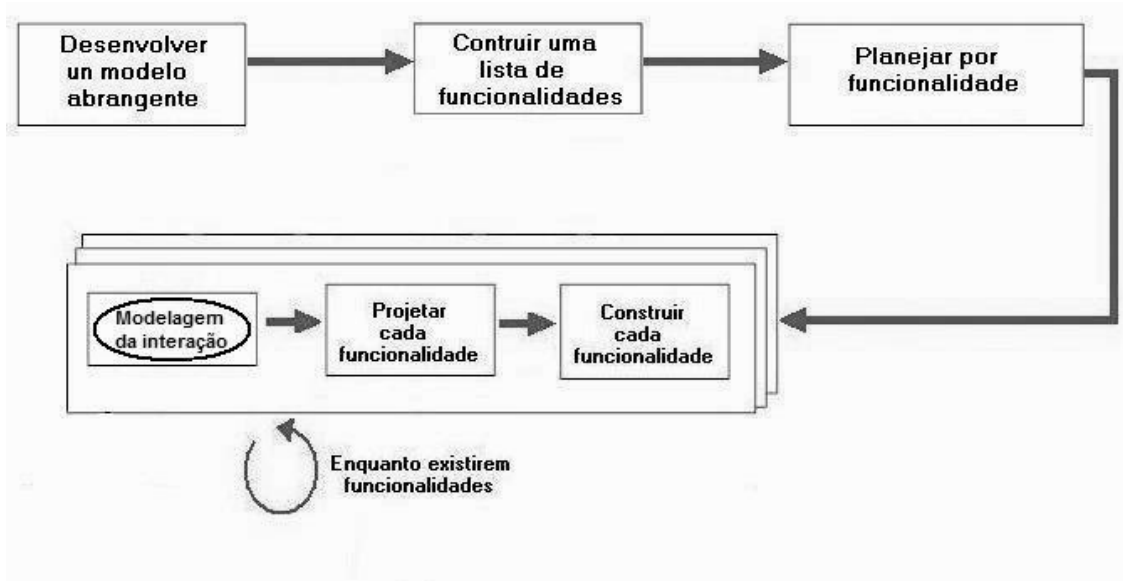


Figura 1. Modelagem da Interação no FDD

3.2. Processo DSDM

O DSDM (*Dynamic System Development Method*), desenvolvido na década dos 90 na Inglaterra, é um framework para o desenvolvimento ágil de software, portanto uma de suas características é a aceitação de mudanças. Ele é uma extensão do RAD (*Rapid Application Development*) e tem como objetivo principal gerenciar as ações dentro dos limites desejados a respeito do tempo e do orçamento [TEIXEIRA].

O processo do DSDM apresenta três etapas: Pré-projeto, Projeto e Pós-projeto. O projeto, por sua vez, está dividido em cinco fases: estudo de viabilidade, estudo de negócio, modelagem funcional, projeto e construção, e implementação. A parte iterativa nesse método se encontra dentro do projeto, durante a execução das três últimas fases dessa etapa. Nesse método a prática de modelagem da interação substituirá a criação dos protótipos e poderá ser feita dentro da fase do projeto, na etapa de modelagem funcional, quando são desenvolvidos os protótipos que serão mostrados ao cliente, conforme mostrado na figura 2. Sempre que os requisitos selecionados na iteração corrente tratarem de interações entre o usuário e o sistema, os UIDs deverão substituir os protótipos. A modelagem da interação no DSDM pode ser realizada pelo gerente do domínio. A prática de prototipagem é substituída pela modelagem da interação.

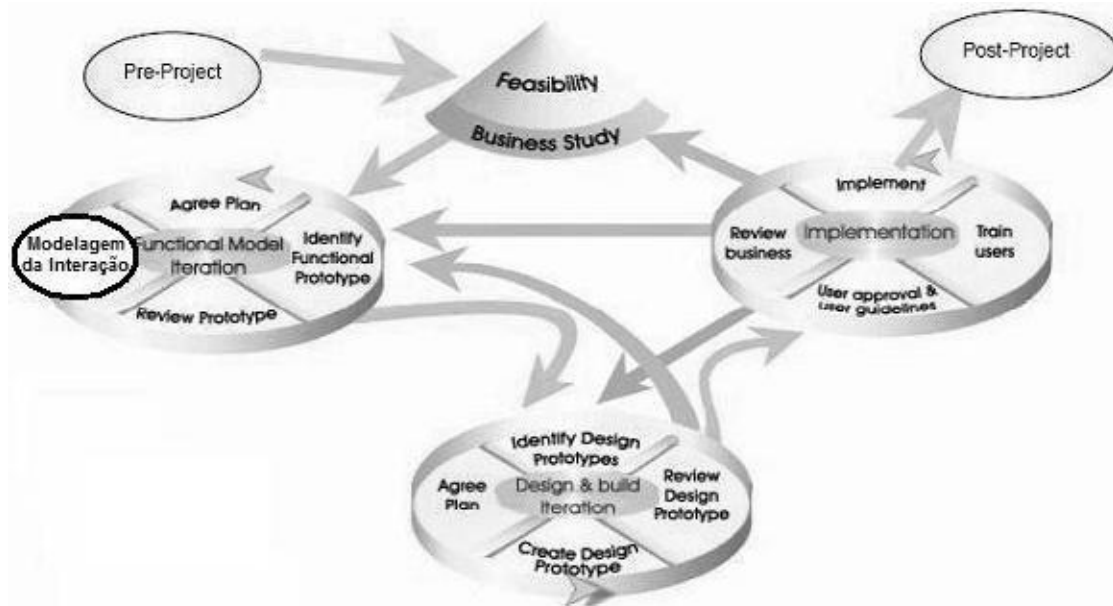


Figura 2. Modelagem da Interação no DSDM

3.3. Processo XP

O XP (*Extreme Programming*), criado por Kent Beck no final da década de 90, é o método ágil mais conhecido atualmente. Assim como outros métodos ágeis, entre suas características podemos citar: grupos pequenos de desenvolvedores; softwares desenvolvidos em um tempo menor do que os desenvolvidos utilizando métodos tradicionais; requisitos dinâmicos que não estão totalmente definidos e que mudam constantemente [FAGUNDES]. “No XP o cliente define as funcionalidades do sistema que será desenvolvido por meio das chamadas estórias do usuário, e as prioriza em seguida” [GARCIA ET AL].

O processo do XP tem cinco etapas, sendo que somente a terceira corresponde à parte iterativa do método, embora as etapas de planejamento e produção possam ser executadas mais do que uma vez. Nesse método a prática de modelagem será adicionada antes de iniciar cada iteração, após a fase de planejamento, onde são escolhidas as estórias que vão ser usadas na primeira iteração, conforme mostrado na figura 3. Para cada estória de usuário que apresenta grande troca de informações entre o usuário e o sistema, também será modelada a interação do usuário com o sistema. Isso ajudará na hora da comunicação com o cliente para que o cliente compreenda melhor a interação entre o usuário e o sistema, especificamente as informações trocadas na iteração. A modelagem da interação do usuário no XP pode ser realizada pelo testador. É proposta a inclusão da prática de modelagem da interação às práticas já existentes no método, e ela pode ser aplicada de forma conjunta com a prática jogo de planejamento, que é onde se decide o que vai ser feito na seguinte iteração.

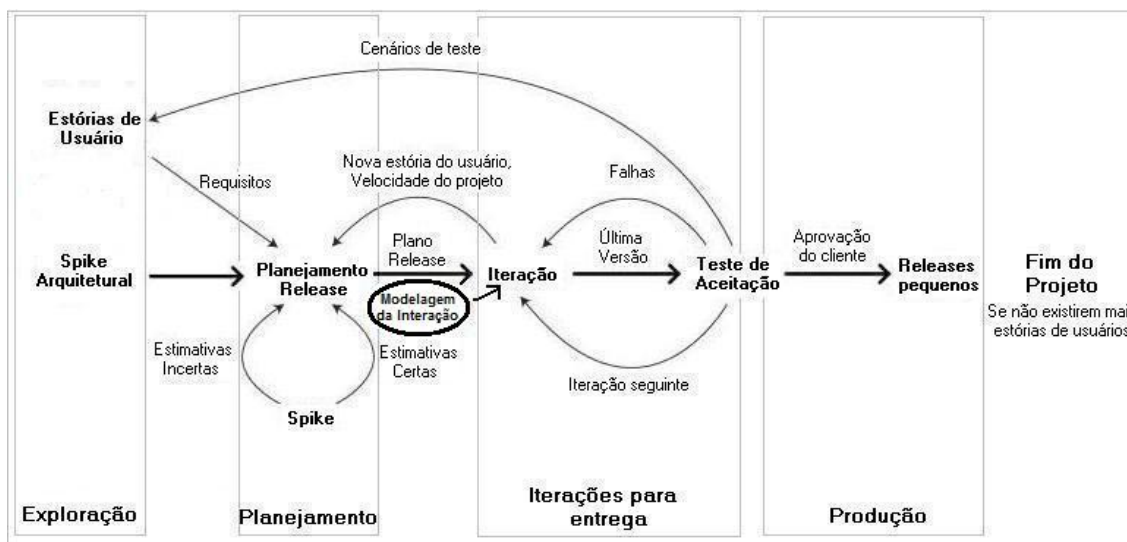


Figura 3. Modelagem da Interação no XP

3.4. Processo SCRUM

Scrum, cuja definição foi formalizada por Ken Schwaber nos anos 90, é um método ágil direcionado para a gerência de projetos. O objetivo desse método é a entrega, dentro de iterações, de um software de qualidade que seja útil para o cliente. Essas iterações são formadas por *Sprints*, que são ciclos de, no máximo, 30 dias, onde se trabalha para alcançar objetivos específicos. Estes objetivos são representados por uma lista de funcionalidades, chamada de *Product Backlog*, definida pelo Dono do Produto (*Product Owner*) e constantemente atualizada e priorizada, [SANCHEZ].

O processo do *Scrum* está dividido em três etapas: *PreGame*, *Game* ou Desenvolvimento e *PostGame*. A parte iterativa é a de desenvolvimento. Nesse método a modelagem da interação do usuário será realizada na fase de desenvolvimento, após a criação do *Product Backlog* e antes da criação do *Sprint Backlog*, que é a lista das funcionalidades a serem desenvolvidas no próximo *Sprint*, ou próxima iteração, conforme mostrado na figura 4. Para cada funcionalidade associada com uma grande troca de informações entre o usuário e o sistema, será modelada a interação do usuário com o sistema. Assim, o cliente poderá ver como vai funcionar a interação entre o usuário e o sistema segundo os requisitos que ele pediu. A modelagem da interação no Scrum pode ser realizada por quem define o *Sprint Backlog*. Essa modelagem, assim como no XP, também pode ser vista como uma nova prática no *Scrum*, e pode ser aplicada em conjunto com a prática *Sprint Backlog*.

4. Estudo de Caso

Para analisar a inclusão da nova prática de modelagem da interação no desenvolvimento ágil, os quatro métodos ágeis citados anteriormente foram utilizados no desenvolvimento de duas aplicações. Cada aplicação foi desenvolvida usando dois métodos ágeis, e cada método ágil foi utilizado em duas iterações, uma incluindo a prática de modelagem da interação do usuário e outra não.

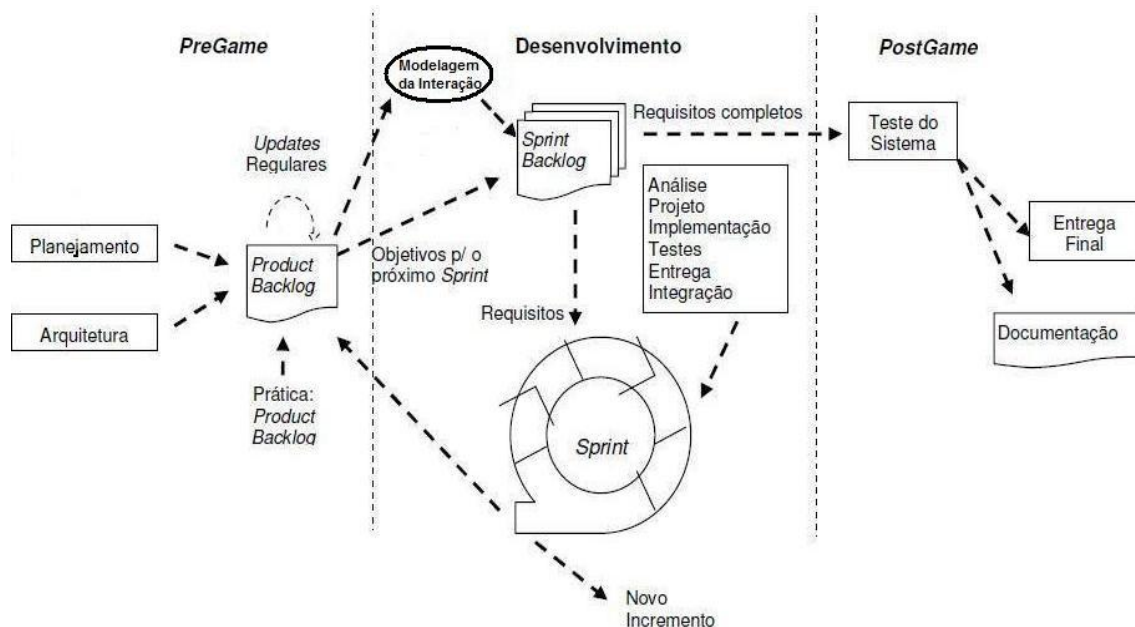


Figura 4. Modelagem da Interação no Scrum

As duas aplicações foram realizadas por um desenvolvedor, tendo somente um cliente em cada aplicação, que era quem tinha o contato contínuo com o desenvolvedor, e, ocasionalmente, tinha outro usuário pertencente à empresa (cliente) que testava a versão funcional do software depois de cada iteração.

No desenvolvimento dessas aplicações não foram aplicadas as práticas que precisavam mais do que uma pessoa na equipe, entre elas podemos citar as reuniões diárias do Scrum, a programação em pares do XP, as equipes por funcionalidade do FDD e o workshop do DSDM.

A técnica utilizada na aplicação da prática de modelagem da interação nos quatro métodos ágeis foram os UIDs (User Interaction Diagramas), que representam a interação do usuário com o sistema.

A seguir são apresentados os UIDs e o resultado do desenvolvimento de cada aplicação.

4.1. UIDs

Os UIDs representam a interação entre o usuário e o sistema, necessária para realizar as tarefas desejadas pelo usuário de uma aplicação [VILAIN]. Os UIDs podem ser usados em conjunto com os casos de uso e, portanto, podem ser usados em qualquer processo de software que os utilize. Para cada caso de uso pode ser definido um UID composto por um conjunto de estados de interação, que incluem as informações trocadas durante a interação usuário-sistema, conectados através de transições.

Na figura 5, apresentada a seguir, é mostrado um UID com seus principais elementos: entrada do usuário (informação fornecida pelo usuário), estado (contém a informação trocada entre o usuário e o sistema), transição (troca de foco da interação), e saída do sistema (informação do sistema que é mostrada para o usuário).

Esse UID corresponde à funcionalidade de cadastro de novo tipo de café, da primeira aplicação. Nele o usuário insere o id, nome, apresentação e tamanho do café para serem adicionados como um tipo novo. Para que essa funcionalidade seja concluída corretamente o id inserido não deve existir, pois todos os cafés têm id único.

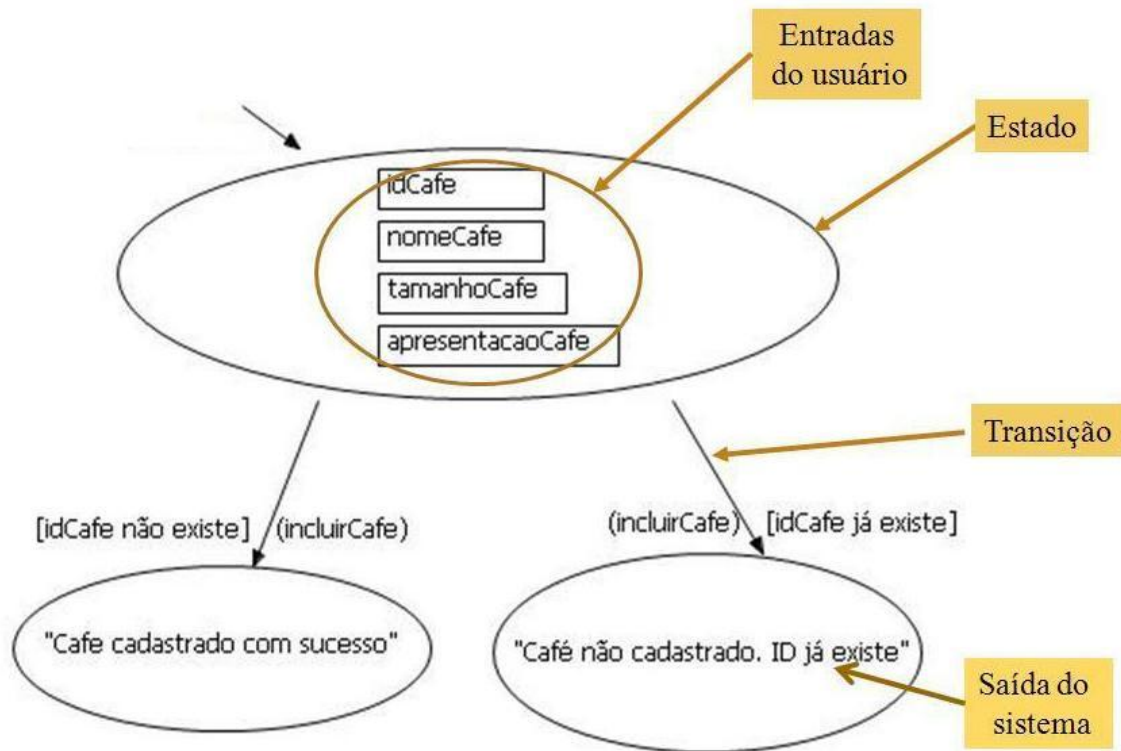


Figura 5. Exemplo de UID

4.2. Primeira aplicação: Sistema de Vendas Fábrica de Café

A primeira aplicação desenvolvida apresentou as seguintes funcionalidades: venda dos cafés, nas diferentes apresentações e tamanhos; relatório de estoque e vendas; registro do pedido, emissão de ordem de processamento; registro do pagamento; cadastramento e exclusão de cadastro de cliente; inclusão de novo café e novo estoque; exclusão de café. Nesta aplicação foram usados os métodos FDD, nas duas primeiras iterações, e DSDM, nas duas seguintes. A figura 6 mostra a interface inicial desta aplicação.

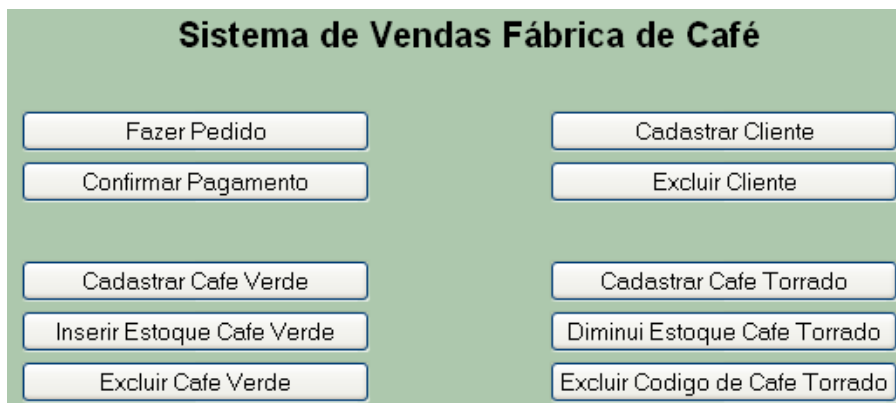


Figura 6. Interface inicial do sistema

4.2.1. Análise do FDD com Modelagem da Interação

Na primeira iteração o processo do FDD foi desenvolvido normalmente, sem modificações, já na segunda iteração foram incluídos os UIDs no início da iteração para modelar a interação do usuário com o sistema. No FDD, os diagramas clássicos, como o de seqüência e o diagrama

de classes cobriram todas as necessidades referentes à análise e projeto. Nesse caso os UIDs só foram aproveitados na hora da comunicação com o usuário e não tanto no desenvolvimento do projeto. Porém, pode-se notar a importância da modelagem da interação entre o usuário e o sistema, pois o cliente/usuário conseguiu entender facilmente os diagramas mostrados, o que manteve a comunicação ágil. Em relação ao tempo não foi feita uma medição exata, mas a inclusão dos UIDs não significou uma diferença notória para o desenvolvedor, cujo papel foi desenvolvido por uma pessoa só, que foi quem assumiu também o papel de responsável pela criação dos UIDs. Não se notou essa diferença, pois os UIDs são diagramas fáceis de desenhar. Além disso, os UIDs também são fáceis de entender pelo usuário que terá acesso no momento da comunicação entre ele e o desenvolvedor, o que é uma vantagem, pois a comunicação pode ser mais proveitosa.

4.2.2. Análise do DSDM com Modelagem da Interação

Na terceira iteração, com o DSDM, os protótipos foram realizados usando os UIDs, e na quarta e última iteração foi executado o processo normal do método sem os UIDs. No DSDM os UIDs foram mais aproveitados do que no FDD, substituindo os protótipos do sistema, mantendo o processo ágil, pois não foi necessário um protótipo baseado em código. Por se tratar de uma aplicação pequena, o uso de UIDs como protótipos é mais prático. Porém, os UIDs só são indicados para substituir protótipos que mostram a interação entre o usuário e o sistema. Em relação ao tempo, assim como no método anterior, não se detectou uma demora significativa na utilização dos UIDs no desenvolvimento, embora não tenha se realizado uma medição exata. O uso dos UIDs substituindo protótipos de implementação faz bastante diferença no tempo. Em relação a protótipos com interfaces, o uso dos UIDs é mais rápido, mas não muito.

4.3. Segunda aplicação: Sistema Administrativo de gastos da empresa.

As funcionalidades da segunda aplicação incluíram: cadastro de folha de pagamento de funcionário; cadastro, atualização e relatório de despesas; relatório de vendas, relatório de receitas; cálculo do saldo mensal. Nesta aplicação foram usados os métodos XP, nas duas primeiras iterações, e Scrum, nas duas seguintes. A figura 7 mostra a interface de uma das funcionalidades do sistema.

4.3.1. Análise do XP com Modelagem da Interação

Na primeira iteração o processo do XP foi desenvolvido sem a inclusão de UIDs, sendo que esta foi realizada no início da segunda iteração. No XP, que não requer a criação dos diagramas clássicos, e onde são criados testes de unidade e aceitação antes do desenvolvimento do sistema, os UIDs serviram de apoio como entrada para a implementação tanto do código quanto da interface. Além disso, o XP não possui uma modelagem da interação entre o usuário e o sistema, e por ser um método onde a comunicação com o usuário é contínua, a inclusão dos UIDs no XP melhorou o entendimento do usuário. Em relação ao tempo, apesar de não ter sido feita uma medição exata, a iteração com UID não apresentou uma demora notória para o desenvolvimento.

Atualiza Funcionario:

CPF: (só números)

CPF:

Nome:

Endereço:

Telefone:

Email:

Tipo:

Salario:

Horas Extra:

Faltas:

Figura 7. Interface atualiza funcionário

4.3.2. Análise do Scrum com Modelagem da Interação

Na terceira iteração, com o *Scrum*, o processo desse método foi realizado normalmente, sem UIDs. Na quarta e última iteração, foram inseridos os UIDs no início da parte iterativa, na etapa de desenvolvimento ou GamePhase. O caso do *Scrum* é um pouco diferente dos outros métodos, porque ele não define um procedimento para o desenvolvimento, pois é mais voltado à gerência do projeto. A inserção dos UIDs no Scrum, que também não requer a criação de qualquer diagrama clássico, serviu de apoio para a implementação do código, pois os UIDs simulam a parte da interface. Entretanto, a inclusão dos UIDs dependerá do processo usado nas iterações. Se for usado um processo que já modela a interação entre o usuário e o sistema, os UIDs não serão necessários.

5. Conclusões

Nesse trabalho foi feita uma análise da necessidade da modelagem da interação entre o usuário e o sistema no desenvolvimento ágil, tomando como exemplo os quatro métodos ágeis estudados, que são o FDD, o DSDM, o XP e o Scrum. Para isto foi realizado o estudo de caso com duas aplicações que foram desenvolvidas utilizando estes quatro métodos ágeis juntamente com a utilização dos UIDs. Para esse trabalho os UIDs foram usados como técnica de modelagem, entretanto é importante salientar que outras técnicas poderiam ter sido usadas.

Os métodos ágeis que foram usados são métodos bem diferentes entre si. O FDD é um método um pouco mais parecido com os métodos tradicionais; o DSDM utiliza a prototipação no desenvolvimento do software, prática que foi substituída pela técnica de modelagem da interação; o XP que é mais direcionado para a programação; e o Scrum é direcionado para o gerenciamento e não especifica um processo a ser seguido em cada iteração.

Em todos os quatro métodos, sentiu-se a falta de uma modelagem que permitisse mostrar ao usuário o comportamento do sistema no momento da interação com ele. E como os métodos ágeis têm entre suas características a comunicação constante e a opção de adicionar, durante o processo, algum requisito que o usuário possa ter esquecido, pode-se notar a importância de uma modelagem para que essa comunicação seja melhorada.

A modelagem da interação, através dos UIDs, foi feita para cada funcionalidade ou estória de usuário apresentada nas duas aplicações, nas iterações escolhidas, e pode-se notar que no caso de funcionalidades maiores, ou com mais troca de informações, os UIDs tiveram uma participação melhor, já que em sistemas com funcionalidades com pouca troca de informação, a modelagem não faz diferença. Em relação ao tempo, o tempo dispendido na criação dos UIDs não significou muito em comparação ao que eles oferecem na comunicação com o usuário.

Outra constatação importante foi que a modelagem da interação foi mais aproveitada nos processos de alguns métodos do que em outros. No FDD, não fez muita diferença a inclusão desta modelagem no processo. No caso do DSDM, esta modelagem foi mais bem aproveitada do que no anterior, pois foi usado no lugar dos protótipos, mantendo o processo ágil, não precisando de um protótipo baseado em código. No XP, que não requer a criação dos diagramas clássicos, e onde são criados testes de unidade e aceitação antes do desenvolvimento do sistema, a modelagem da interação serviu de apoio como entrada para a implementação tanto do código quanto da interface. No Scrum, que também não requer a criação de qualquer diagrama clássico, a inclusão da modelagem da interação facilitou um pouco a implementação do código, pois os UIDs simularam a parte da interface com o usuário.

Mas independente do método utilizado, no geral, o que os usuários querem é que o programa que eles pediram fique pronto, sem se preocupar muito em como ele foi desenvolvido, mas com todas as funcionalidades que pediram funcionando. O uso dos UIDs para modelar a interação do usuário com o sistema é uma maneira simples e direta de se comunicar com o usuário. Isto se faz de uma maneira rápida, pois o usuário consegue entender o fluxo de dados sem muito problema.

Portanto, a inclusão da modelagem da interação pode ser vista como uma prática opcional que pode ser incluída em qualquer método ágil sempre que uma funcionalidade que apresenta grande interação entre o usuário e o sistema estiver sendo desenvolvida. Já em aplicações que tem pouca interação não seriam muito úteis.

Apesar do estudo de caso ter sido desenvolvido com uma equipe de apenas um desenvolvedor, isto não invalida a análise feita neste trabalho porque as práticas ágeis relacionadas com a modelagem da interação foram executadas.

6. Bibliografia

- FAGUNDES, Priscila Basto. **Framework Para Comparação e Análise de Métodos Ágeis**. Dissertação de Mestrado, Depto Informática e Estatística, UFSC, 2005.
- GARCIA D.C, Edes. PENTEADO, Rosângela. COUTINHO A.S, Junia. VACCARE B, Rosana T. **Padrões e Métodos Ágeis: agilidade no processo de desenvolvimento de software**. Universidade Federal de São Carlos – Departamento de Computação. Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação.
- HEPTAGON - **FDD - Feature Driven Development**. Disponível em: <http://www.heptagon.com.br/fdd> – Último acesso em 18/03/2009.
- MACHADO, Thiago Leão. **Uma Ferramenta de Suporte ao Framework para comparação e Análise de métodos Ágeis**. Trabalho de Conclusão do Curso Sistemas de Informação, Depto Informática e Estatística, UFSC, 2005.
- PREECE, J.; ROGERS, Y.; SHARP, H.; BENYON, D.; HOLLAND, S.; CAREY, T. **Human-Computer Interaction**. Addison-Wesley, 1994. 775p.

SANCHEZ, Ivan. **Coding Dojo Floripa Desenvolvimento Ágil**. Disponível em: <http://dojofloripa.wordpress.com/2007/02/07/scrums-em-2-minutos/> - Último acesso em: 24/05/09.

TEIXEIRA, Daniel D. AFONSO P, Fernando J. GAIOLAS D. S. P, José P. Gonçalves P. S, Tiago A. **DSDM – Dynamic Systems Development Methodology**. Faculdade de Engenharia da Universidade do Porto. Disponível em: http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_14.pdf - Último acesso em: 27/05/09.

VILAIN, Patrícia. **Modelagem da interação com o usuário em aplicações hipermídia**. Tese de Doutorado, PUC-RIO, Rio de Janeiro, 2002.

ANEXO 4 – Código Fonte

- **Index.php**

```
<?php
require_once (./Interfaces.class.php);
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Sistema de Vendas Fábrica de Café</title>
<link href="css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="header">
    <h1>Sistema de Vendas Fábrica de Café</h1>
</div><!-- ends header -->
<div id="container" class="clearfix">
<div id="content">
    <?php $interface = new Interfaces();
        $interface-> mostraInicio();?>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>
```

- **Interfaces.php**

```
<?php
class Interfaces{
public function Interfaces(){

}

public function mostraInicio(){
echo "
<html>
<table class='inicio'>
<tr class='um'>
<td class='pedido'>
    <form class='fazer' action='adm/pedido/pedido.php'> <input type='submit' value='Fazer Pedido'
name='enviar' id='enviar'></form>
    <form action='adm/pedido/pagamento.php'> <input type='submit' value='Confirmar Pagamento'
name='enviar' id='enviar'></form></td>
<td class='cliente'>
    <form class='fazer' action='adm/cliente/cadastroCliente.php'><input type='submit' value='Cadastrar
Cliente' name='enviar' id='enviar'></form>
    <form action='adm/cliente/exclusaoCliente.php'><input type='submit' value='Excluir Cliente'
name='enviar' id='enviar' ></form></td></tr>
<tr class='dois'>
<td class='verde'>
    <form class='fazer' action='adm/cafes/cadastroCafeVerde.php'><input type='submit' value='Cadastrar
Cafe Verde' name='enviar' id='enviar'></form>
    <form class='fazer' action='adm/cafes/estoqueCafeVerde.php'><input type='submit' value='Inserir
Estoque Cafe Verde' name='enviar' id='enviar'></form>
```

```

        <form action='adm/cafe/exclusaoCafeVerde.php'><input type='submit' value='Excluir Cafe Verde'
        name='enviar' id='enviar'></form></td>
<td class='torrado'>
    <form class='fazer' action='adm/cafe/cadastroCafeTorrado.php'><input type='submit' value='Cadastrar
    Cafe Torrado' name='enviar' id='enviar'></form>
    <form class='fazer' action='adm/cafe/diminuiEstoqueCafeTorrado.php'> <input type='submit'
    value='Diminui Estoque Cafe Torrado' name='enviar' id='enviar' ></form>
    <form action='adm/cafe/exclusaoCafeTorrado.php'><input type='submit' value='Excluir Codigo de
    Cafe Torrado' name='enviar' id='enviar'></form></td></tr>
<tr class='tres'>
<td class='esq'>
    <form action='adm/adm/relatorioEstoque.php'><input type='submit' value='Relatório Estoque'
    name='enviar' id='enviar'></form></td>
<td class='dir'>
    <form action='adm/adm/relatorioVendas.php'><input type='submit' value='Relatório Vendas'
    name='enviar' id='enviar'></form></td></tr>
</table>
-----
<p class='tituloAdm'>Administração</p>
<table class='inicio1'>
<tr>
<td class='esq'>
    <form action='adm/adm/despesas.php'> <input type='submit' value='Cadastro Despesas' name='enviar'
    id='enviar'></form></td>
<td class='dir'>
    <form action='adm/cliente/cadastroFuncionario.php'> <input type='submit' value='Cadastro
    Funcionario' name='enviar' id='enviar'></form></td></tr>
<tr>
<td class='esq'>
    <form action='adm/adm/relatorioDespesas.php'> <input type='submit' value='Relatorio Despesas'
    name='enviar' id='enviar'></form></td>
<td class='dir'>
    <form action='adm/cliente/atualizaFuncionario.php'> <input type='submit' value='Atualiza Funcionario'
    name='enviar' id='enviar'></form></td></tr>
<tr>
<td class='esq'>
    <form action='adm/adm/relatorioMensalDespesas.php'> <input type='submit' value='Relatorio Mensal
    Despesas' name='enviar' id='enviar'></form></td>
<td class='dir'>
    <form action='adm/cliente/exclusaoFuncionario.php'> <input type='submit' value='Exclui Funcionario'
    name='enviar' id='enviar'></form></td></tr>
<tr>
<td class='esq'>
    <form action='adm/adm/relatorioMensalVendas.php'> <input type='submit' value='Relatorio Mensal
    Vendas' name='enviar' id='enviar'></form></td>
<td class='dir'>
    <form action='adm/cliente/cadastroFolhaPagamento.php'> <input type='submit' value='Cadastro Folha
    Pagamento' name='enviar' id='enviar'></form></td></tr>
<tr>
<td class='esq'>
    <form action='adm/adm/lucro.php'> <input type='submit' value='Lucro' name='enviar'
    id='enviar'></form></td>
<td class='dir'>
    <form action='adm/cliente/relatorioFolhaPagamento.php'> <input type='submit' value='Relatorio
    Pagamentos' name='enviar' id='enviar'></form></td></tr>
</table>
</html>";
} } ?>

```


Adm/adm

- **atualizaDespesa.php**

```
<?php
    session_start();
?>
<script> //guarda o tipo do select option
function getTipo(what) {
    if (what.selectedIndex != "") {
        var tipo = what.value;
        document.location=('atualizaDespesa.php?tipo=' + tipo);
    }
}
function getNome(what) {
    if (what.selectedIndex != "") {
        var nome = what.value;
        document.location=('atualizaDespesa.php?nome=' + nome);
    }
}
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/DespesaDAO.class.php");
    require_once("../classes/Despesa.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
    $despesaDAO = new DespesaDAO();

    $tipo = $_GET["tipo"]; // $tipo = 1, 2 ou 3      (número dos tipos de despesa: fixa, variável ou
    produção)
    $nome = $_GET["nome"]; // $nome = agua, luz, todos, etc (nomes das despesas)
    $idTipo = $despesaDAO->buscaIdTipo($tipo); //procura id do tipo de despesa (1, 2 ou 3)
    if ( isset ( $_POST["data"] )) {
        $data = $_POST["data"];
        if ($nome) {
            $despesa = $despesaDAO->buscaDespesaGeral($nome, $data);
            if (!$despesa) {
                $comDespesa = false;
            } else {
                $comDespesa = true; } }
        if (isset ( $_POST["atualiza"] )) {
            $id = $_POST["idDespesaGeral"];
            $data = $_POST["data"];
            $valor = $_POST["valor"];
            $despesaDAO->atualizaDespesaGeral($id, $data, $valor);
            echo "<p>Despesa atualizada com sucesso!</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage();
    }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Atualização de Despesas</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
```



```

    }?>
    <tr> <td><input type='submit' value='Procura Despesa' name='procuraDespesa'></td></tr>
    <tr> <td><input type='submit' value='Atualiza' name='atualiza'></td>
    <td align="right"><a href=" ../index.php"><input type='submit' value='voltar'
    name='voltar'></a></td></tr>
</table>
    </form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **cadastraDespesa.php**

```

<?php
    session_start();
    ?>
<script> //guarda o id do select option
    function getStates(what) {
        if (what.selectedIndex != "") {
            var id = what.value;
            document.location=('cadastraDespesa.php?id=' + id);}
    }
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/DespesaDAO.class.php");
    require_once("../classes/Despesa.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
    $despesaDAO = new DespesaDAO();
    $despesa = new Despesa();

    //busca os Ids para preencher o select option
    $listaTipos = $despesaDAO->buscarTipos();
    //coloca na variável $id o valor selecionado no select option (script)
    $id = $_GET["id"];
    if ( isset ( $_POST["nomeDespesa"] ) ) {
        $nomeDespesa = $_POST["nomeDespesa"];
        foreach ( $listaTipos as $lista ) {
            if ( $lista->getNomeTipoDespesa() == $id ) {
                $idTipo = $lista->getIdTipoDespesa(); }
            $despesa->setIdTipoDespesa($idTipo);
            $despesa->setNomeDespesa($nomeDespesa);
            $despesaDAO->adicionaDespesa($despesa);
            echo "<p>Despesa adicionada com sucesso</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage();
    }
} ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Despesas</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />

```

```

</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
  <form action="" method='POST' id='despesa'>
  <table>
    <tr>
      <td><strong>Cadastra nova despesa:</strong></td></tr>
    </table>
    <table>
      <tr>
        <td>Tipo de despesa:</td>
        <td><select name='tipoDespesa' onchange='getStates(this);'>
          <option value=""> - </option>
        </td>
      </tr>
      <tr>
        <td><?php
        foreach ($listaTipos as $tipoDespesas){?>
          <option value="<?php echo ($tipoDespesas->getNomeTipoDespesa());?>"><?php echo
          $tipoDespesas->getNomeTipoDespesa();? </option><? }?>
        </td>
      </tr>
      <tr>
        <td><?php if($id) {
        echo "<td></td>
        <td><input type='text' maxlength='50' name='tipoDespesa' readonly='readonly'
        size='20' value='Despesa ".$id."></td>";
        }?></tr>
      </tr>
      <tr>
        <td>Nome da despesa: </td>
        <td><input type='text' maxlength='50' name='nomeDespesa' size='20' value=""></td></tr>
      </tr>
      <td><input type='submit' value='Cadastra Despesa' name='cadastroDespesa'></td>
      <td align="right"><a href="despesas.php">
      <input type='submit' value='voltar' name='voltar'></a></td>
    </tr>
  </table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **despesas.php**

```

<?php
  session_start();?>
<script>
function getStates(what) { //guarda o id do select option
  if (what.selectedIndex != "") {
    var id = what.value;
    document.location=('despesas.php?id=' + id);}
  }
function getTipo(what) { //guarda o tipo do select option
  if (what.selectedIndex != "") {
    var tipo = what.value;
    document.location=('despesas.php?tipo=' + tipo);}
  }
</script>
<?php

```

```

try {
    require_once("../classes/Conexao.class.php");
    require_once("../classes/Despesa.class.php");
    require_once("../dao/DespesaDAO.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
    $despesaDAO = new DespesaDAO();
    $despesa = new Despesa();

    //coloca na variável $id o valor selecionado no select option (script)
    $nome = $_GET["id"];
    $tipo = $_GET["tipo"];
    $idTipo = $despesaDAO->buscaIdTipo($tipo);

    //busca os Ids para preencher o select option
    if ($idTipo){
        $listaIDs = $despesaDAO->buscarNomesDespesa($idTipo);
        if ( count( $listaIDs ) == 0 ){
            echo "<p>Nenhum tipo de despesa cadastrado. Clique em <strong>Cadastrar
            Despesa</strong> para adicionar um.</p>";}}
    if (isset($_POST['insereDespesa'])){ //lê valores inseridos em dataDespesa e valor despesa e os insere
    em despesaGeral
        $dataDespesa = $_POST['dataDespesa'];
        $valorDespesa = $_POST['valorDespesa'];
        $idDespesa = $despesaDAO->buscarIdDespesa($nome);
        $despesa->setIdDespesa($idDespesa->getIdDespesa());
        $despesa->setDataDespesa($dataDespesa);
        $despesa->setValorDespesa($valorDespesa);
        $despesaDAO->insereDespesaGeral($despesa);
        echo "<p>A despesa <strong>".$nome."</strong> foi inserida com sucesso </p>";}

    } catch ( Exception $e ) {
        echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Despesas</title>
<link href="../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
    <form action="" method='POST' id='despesa'>
    <table>
        <tr>
            <td><strong>Cadastro de despesas:</strong></td>
        </tr>
    </table>
    <table class="cadastroDespesas">
        <tr>
            <td>Selecione o tipo de despesa:</td>
            <td><select id="selection" name="tipoDespesa" onChange="getTipo(this);">
                <option value = " "> - </option>
                <option value = "Fixa"> Fixa </option>
                <option value = "Variavel"> Variável </option>
            </td>
        </tr>
    </table>

```

```

                <option value = "Producao"> Produção </option>
            </select></td></tr>
<?php if ($idTipo){
    echo "<tr>
        <td>Tipo de Despesa:</td>
        <td><input type=text maxlength='50' name='tipoDespesa' readonly='readonly' size='15'
            value='".$tipo."></td> </tr>";}?>
    <tr>
        <td>Selecione o nome da despesa:</td>
        <td><select name='idDespesa' onchange='getStates(this);'>
            <option value=" " - </option><?php
foreach ($listaIDs as $despesa){?>
            <option value="<?php echo ($despesa->getNomeDespesa());?>"> <?php echo
                $despesa->getNomeDespesa();?></option> <?php }?>
            </select></td></tr>
</form>
<form action=" method="post" id="insereDespesa">
<?php
    if ($nome){
        echo "<tr>
            <td>Despesa:</td>
            <td><input type=text maxlength='50' name='nomeDespesa' readonly='readonly'
                size='15' value='".$nome."></td>
            </tr>
            <tr>
            <td>Data despesa: (dd/mm/aa)</td>
            <td><input type=text maxlength='8' name='dataDespesa' size='15' value=" "></td></tr>
            <tr>
            <td>Valor despesa:</td>
            <td><input type=text maxlength='50' name='valorDespesa' size='15'
                value=" "></td></tr>";}?>
            <tr class="insereDespesa">
            <td><input type="submit" value="Insere despesa ao sistema"
                name="insereDespesa"></td>
</form>
        <td align="right"><a href=" ../index.php">
            <input type='submit' value='voltar' name='voltar'></a></td>
        </tr>
</table>
<table class='cadastroDespesas'>
    <tr><td class="botaoCadastra"><strong>No caso de não ter o nome da despesa desejada,
        cadastrá-la clicando no botão abaixo:</strong></td></tr>
    <tr>
<form action='cadastraDespesa.php'>
        <td ><input type="submit" name="CadastraDespesa" value="Cadastra Nova Despesa"></td>
</form>
    </tr>
    <tr>
<form action='atualizaDespesa.php'>
        <td ><input type="submit" name="atualizaDespesa" value="Atualiza Despesa"></td>
</form></tr>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **relatorioDespesas.php**

```

<?php
    session_start();
?>
<script> //guarda o tipo do select option
function getTipo(what) {
    if (what.selectedIndex != "") {
        var tipo = what.value;
        document.location=('relatorioDespesas.php?tipo=' + tipo);}}
function getNome(what) {
    if (what.selectedIndex != "") {
        var nome = what.value;
        document.location=('relatorioDespesas.php?nome=' + nome);}}
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/DespesaDAO.class.php");
    require_once("../classes/Despesa.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
    $despesaDAO = new DespesaDAO();
    $tipo = $_GET["tipo"]; //$tipo = fixa, variável ou produção (nomes dos tipos de despesas)
    $nome = $_GET["nome"]; //$nome = agua, luz, todos, etc (nomes das despesas)

    $idTipo = $despesaDAO->buscaIdTipo($tipo); //procura id do tipo de despesa (1, 2 ou 3)
    $despesaDAO->guardaDados($idTipo); //guarda o idTipo no BD

    $mostra = false;
    if ( isset ($_POST["dataInicio"])){
        $dataInicio = $_POST["dataInicio"];
        $dataFim = $_POST["dataFim"];
        $mostra = true;}

    if ($mostra && $nome == "Todos" ){
        $idTipo = $despesaDAO->pegarIdTipo1(); //pega o idTipo do BD (1, 2 ou 3) quando foram
        escolhidas "todas" as despesas
        $listaDespesas = array();
        $listaDespesas = $despesaDAO->buscarDespesas($idTipo); //procura as despesas do tipo
        idTipo no BD
        if ( count( $listaDespesas ) == 0 ){
            echo "<p>Nenhuma despesa desse tipo cadastrada.";}
        foreach ($listaDespesas as $lista){
            $listaDespesasGerais[] = $despesaDAO->buscaDespesasGerais($lista-
            >getIdDespesa(), $dataInicio, $dataFim);}
    }else if ($mostra){
        $idDespesa = $despesaDAO->buscarIdDespesa($nome);
        $listaDespesasGerais[] = $despesaDAO->buscaDespesasGerais($idDespesa->getIdDespesa(),
        $dataInicio, $dataFim);}

    } catch ( Exception $e ) {
        echo $e->getMessage();}
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

```

```

<title>Relatório de Despesas</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method="post" id="relatDesp">
  <table>
    <tr>
      <td><strong>Relatório Despesas:</strong></td>
    </tr>
  </table>
  <table>
    <tr>
      <td>Selecione o tipo de despesa:</td>
      <td><select id="selection" name="tipoDespesa" onChange="getTipo(this);">
        <option value = " "> - </option>
        <option value = "Fixa"> Fixa </option>
        <option value = "Variavel"> Variável </option>
        <option value = "Producao"> Produção </option>
      </select></td>
    </tr>
  </table>
<?php
if ($tipo){
  echo "<tr>
  <td>Tipo de Despesa:</td>
  <td><input type=text maxlength='50' name='tipoDespesa' readonly='readonly' size='15'
  value='". $tipo."></td>  </tr>";
}
if ($nome){
  echo "<tr>
  <td>Tipo de Despesa:</td>
  <td><input type=text maxlength='50' name='tipoDespesa' readonly='readonly' size='15'
  value='". $nome."></td>  </tr>";
}
if ($idTipo){
  $listaDespesas = $despesaDAO->buscarDespesas($idTipo);?>
  <tr>
    <td>Selecione o tipo de despesa:</td>
    <td><select id="selection" name="tipoDespesa" onChange="getNome(this);">
      <option value = " "> - </option>
      <option value = "Todos">Todos</option>
      <option value = " "> - </option>
      <?php foreach ($listaDespesas as $listaDes){ //mostra as despesas do idTipo que foram
      cadastradas
        echo "<option value = ".$listaDes->getNomeDespesa()." ". $listaDes->
        getNomeDespesa() ."</option>"; }
      echo "</select>";
    }
  echo "</td>";
  echo
"</tr>";}?>
  <tr><td>Data Inicio: (dd/mm/aa)</td><td><input type=text maxlength='50' name='dataInicio' size='15'
  value=""></td></tr>
  <tr><td>Data Fim: (dd/mm/aa)</td><td><input type=text maxlength='50' name='dataFim' size='15'
  value=""></td></tr>
  <tr> <td><input type='submit' value='Gera Relatório' name='geraRelatorio'></td>

  <td align="right"><a href="../../index.php"><input type='submit' value='voltar'
  name='voltar'></a></td>
  </tr>
</table>

```



```

</form>
<?php
$total = 0;
if ($mostra){
    echo "<table class='pedido2'>";
    echo "<tr><td>Relatório de Despesas no período de ". $dataInicio. " até ". $dataFim.
    "</td></tr>" ;
    echo "</table>";
    echo "<table class='relatorio'>";
    echo "<tr><td><strong>Código</strong></td> <td> <strong>Data</strong>
    </td><td><strong>Nome</strong></td><td align='right'><strong>Preço</strong></td></tr>";
    foreach($listaDespesasGerais as $gerais1){//mostra despesas
    if (sizeof($gerais1) != 0)
        foreach ($gerais1 as $gerais){
            $nome = $despesaDAO->buscarNomesDespesaComId ($gerais-
            >getIdDespesa());
            echo "<tr>";
            echo "<td>". $gerais->getIdDespesa(). "</td>";
            echo "<td>". $gerais->getDataDespesa(). "</td>";
            echo "<td>". $nome. "</td>";
            echo "<td align='right'>". $gerais->getValorDespesa(). "</td>";

            echo "</tr>";
            $total = $total + $gerais->getValorDespesa();}
        }
    echo "<tr class='total'><td colspan='3'><strong> Total</strong> </td><td
    align='right'>". $total. "</td></tr>";}?"
    </table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **relatorioEstoque.php**

```

<?php
session_start();
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/CafeTorradoDAO.class.php");
    require_once("../dao/CafeVerdeDAO.class.php");

    $conexao = new Conexao();
    $conexao->conectar();

    if (isset ($_POST['estoqueVerde'])){
        $listaCafeVerde = array();
        $cafeVerdeDAO = new CafeVerdeDAO();
        $listaCafeVerde = $cafeVerdeDAO->estoque();}
    if (isset ($_POST['estoqueTorrado'])){
        $listaCafeTorrado = array();
        $cafeTorradoDAO = new CafeTorradoDAO();
        $listaCafeTorrado = $cafeTorradoDAO->estoque(); }
} catch ( Exception $e ) {
    echo $e->getMessage();}?"
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

```

```

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Relatório Estoque</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
    <table>
        <tr>
            <td><strong>Relatório Estoque:</strong></td>
        </tr>
    </table>
    <table>
        <form action="" method="post" id="relatEstoqueVerde">
            <tr><td><input type='submit' value='Estoque Cafe Verde' name='estoqueVerde'></td>
        </form>
        <form action="" method="post" id="relatEstoqueTorrado">
            <tr><td><input type='submit' value='Estoque Cafe Torrado' name='estoqueTorrado'></td>
        </form>
    </tr>
    <tr>
        <td align="left"><a href="../../index.php"><input type='submit' value='voltar'
            name='voltar'></a></td>
    </tr>
</table><?php
if (isset($_POST['estoqueVerde'])) {
    echo "<table border='1' width='30%'>";
    echo "<tr>";
    <td align='center'>ID</td>
    <td align='center'>Nome</td>
    <td align='center'>Quantidade</td>
    </tr>";
    if (sizeof($listaCafeVerde) != 0)
        foreach ($listaCafeVerde as $cafeVerde) {
            echo "<tr>";
            echo "<td>".$cafeVerde->getId()."</td>";
            echo "<td>".$cafeVerde->getNome()."</td>";
            echo "<td align='right'>".$cafeVerde->getEstoque()."</td>";
            echo "</tr>"; } }
if (isset($_POST['estoqueTorrado'])) {
    echo "<table border='1' width='60%'>";
    echo "<tr>";
    <td align='center'>ID</td>
    <td align='center'>Nome</td>
    <td align='center'>Apresentacao</td>
    <td align='center'>Tamanho</td>
    <td align='center'>Preço</td>
    <td align='center'>Quantidade</td>
    </tr>";
    if (sizeof($listaCafeTorrado) != 0)
        foreach ($listaCafeTorrado as $cafeTorrado) {
            echo "<tr>";
            echo "<td>".$cafeTorrado->getId()."</td>";
            echo "<td>".$cafeTorrado->getNome()."</td>";
            echo "<td>".$cafeTorrado->getApresentacao()."</td>";
            echo "<td>".$cafeTorrado->getTamanho()."</td>";
            echo "<td> R$ ".$cafeTorrado->getPreco()."</td>";

```

```

        echo "<td align='right'>".$scafeTorrado->getEstoque()."</td>";

        echo "</tr>";}}?>
    </table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **relatorioMensalDespesas.php**

```

<?php
    session_start();
?>
<script> //guarda o tipo do select option
function getTipo(what) {
    if (what.selectedIndex != "") {
        var tipo = what.value;
        document.location=('relatorioMensalDespesas.php?tipo=' + tipo);
    }
}
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/DespesaDAO.class.php");
    require_once("../dao/VendaDAO.class.php");
    require_once("../classes/Despesa.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
    $despesaDAO = new DespesaDAO();
    $vendaDAO = new VendaDAO();

    $tipo = $_GET["tipo"]; // $tipo = fixa, variável ou produção (nomes dos tipos de despesas)

    $idTipo = $despesaDAO->buscaIdTipo($tipo); //procura id do tipo de despesa (1, 2 ou 3)
    $despesaDAO->guardaDados($idTipo); //guarda o idTipo no BD

    $mostra = false;
    if ( isset ( $_POST["dataInicio"] )){
        $dataInicio = $_POST["dataInicio"];
        $dataFim = $_POST["dataFim"];
        $mostra = true; }

    if ($mostra){
        $idTipo = $despesaDAO->pegaIdTipo1(); //pega o idTipo do BD (1, 2 ou 3) quando foram
        escolhidas "todas" as despesas
        $listaDespesas = array();
        $listaDespesas = $despesaDAO->buscarDespesas($idTipo); //procura as despesas do tipo
        idTipo no BD
        if ( count( $listaDespesas ) == 0 ){
            echo "<p>Nenhuma despesa desse tipo cadastrada."; }
        foreach ( $listaDespesas as $lista ){
            $listaDespesasGerais[] = $despesaDAO->buscaDespesasGerais($lista-
            >getIdDespesa(), $dataInicio, $dataFim); } }
    } catch ( Exception $e ) {
        echo $e->getMessage(); }?>

```

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Relatório Mensal de Despesas</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method="post" id="relatDesp">
  <table>
    <tr>
      <td><strong>Relatório Mensal Despesas:</strong></td>
    </tr>
  </table>
  <table>
    <tr>
      <td>Selecione o tipo de despesa:</td>
      <td><select id="selection" name="tipoDespesa" onChange = "getTipo(this);">
        <option value = " "> - </option>
        <option value = "Fixa"> Fixa </option>
        <option value = "Variavel"> Variável </option>
        <option value = "Producao"> Produção </option>
      </select></td>
    </tr><?php
if ($tipo){
  echo "<tr>
  <td>Tipo de Despesa:</td>
  <td><input type=text maxlength='50' name='tipoDespesa' readonly='readonly' size='15'
  value='".$tipo."'></td>
  </tr>";}
  <tr><td>Data Inicio: (dd/mm/aa)</td><td><input type=text maxlength='50' name='dataInicio'
  size='15' value=""></td></tr>
  <tr><td>Data Fim: (dd/mm/aa)</td><td><input type=text maxlength='50' name='dataFim'
  size='15' value=""></td></tr>
  <tr> <td><input type='submit' value='Gera Relatório' name='geraRelatorio'></td>

  <td align="right"><a href="../../index.php"><input type='submit' value='voltar'
  name='voltar'></a></td>
  </tr>
</table>
</form>
<?php
$total = 0;
if ($mostra){
  echo "<table class='pedido2'>";
  echo "<tr><td>Relatório de Despesas no período de ". $dataInicio. " até ". $dataFim.
  "</td></tr>";
  echo "<table border='1' width='40%'>";
  echo "<tr>
  <td align='center'>Mes</td>
  <td align='center'>Total</td>
  </tr>";
  $tam = sizeof($listaDespesasGerais);
  if ($tam != 0)
    foreach($listaDespesasGerais as $gerais1){

```

```

        if (sizeof($gerais1) != 0)
            foreach ($gerais1 as $gerais){
                $arrayDespesas[] = $gerais; } }
    $totalMes = 0;
    $total = 0;
    $data = $dataInicio;
    $cont = 0;
    $contaDespesas = 0;
    $tam = sizeof($arrayDespesas);
if ($tam != 0)
    foreach ($arrayDespesas as $array){
        $contaDespesas = $contaDespesas + 1;
        $data1 = $array->getDataDespesa();
        $iguais = $vendaDAO->verificaMes($data, $data1);
        if ($iguais == "1"){
            $totalMes = $totalMes + $array->getValorDespesa();
            $cont = $cont + 1;}
        else if($cont > 0) {
            $mes = $vendaDAO->devolveMes($data);
            echo "<tr>";

                echo "<td>".$mes."</td>";
                echo "<td align='right'>R$ ".$totalMes ."</td>";
                echo "</tr>";
                $total = $total + $totalMes;
            $cont = 0;
            $totalMes = 0;
            $totalMes = $totalMes + $array->getValorDespesa();

        }else if ($cont == 0){
            $totalMes = $totalMes + $array->getValorDespesa();
        if ($tam == $contaDespesas){
            $mes = $vendaDAO->devolveMes($data);
            echo "<tr>";
            echo "<td>".$mes."</td>";
            echo "<td align='right'>R$ ".$totalMes ."</td>";
            echo "</tr>";
            $total = $total + $totalMes;
            $cont = 0;
            $totalMes = 0;}
            $data = $data1;
        }
    echo "<tr class='total'><td><strong>Total </strong></td> <td
    align='right'>".$total."</td></tr>";}    ?>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **relatorioMensalVendas.php**

```

<?php
    session_start();
    try {
        require_once("../classes/Conexao.class.php");
        require_once("../dao/VendaDAO.class.php");

        $conexao = new Conexao();

```

```

$conexao->conectar();

if (isset ($_POST['relatorioVenda'])){
    $dataInicio = $_POST['dataInicio'];
    $dataFim = $_POST['dataFim'];
    $listaVenda = array();
    $VendaDAO = new VendaDAO();
    $listaVenda = $VendaDAO->relatorio($dataInicio, $dataFim);}
} catch ( Exception $e ) {
    echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Relatório Vendas</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<table>
<tr>
<td><strong>Relatório Mensal Vendas:</strong></td>
</tr>
</table>
<table style="padding-bottom:10px;">
<form action="" method="post" id="relatVendas">
<tr><td>Data Inicio: </td>
<td><input type='text' maxlength='50' name='dataInicio' size='20' value=""></td></tr>
<tr><td>Data Fim: </td>
<td><input type='text' maxlength='50' name='dataFim' size='20' value=""></td></tr>
<tr>
<td><input type='submit' value='Mostra relatório' name='relatorioVenda'></td>
</form>
</tr>
<tr>
<td align="left"><a href="../../../index.php"><input type='submit' value='voltar'
name='voltar'></a></td>
</tr>
</table><?php
if (isset ($_POST['relatorioVenda'])){
    echo "<table><tr><td><strong>Periodo de ".$dataInicio." até
".$dataFim."</strong></td></tr></table>";
    echo "<table border='1' width='40%'>";
    echo "<tr>
<td align='center'>Mes</td>
<td align='center'>Total</td>
</tr>";
    $totalMes = 0;
    $total = 0;
    $data = $dataInicio;
    $cont = 0;
    $contaVendas = 0;
    $tam = sizeof($listaVenda);
    if ($tam != 0)
        foreach ($listaVenda as $venda){
            $contaVendas = $contaVendas + 1;

```

```

        $data1 = $venda->getDataVenda();
        $iguais = $VendaDAO->verificaMes($data, $data1);
    ($iguais == "1"){
        $totalMes = $totalMes + $venda->getValorVenda();
        $cont = $cont + 1;}
        else if($cont > 0) {
        $mes = $VendaDAO->devolveMes($data);
        echo "<tr>";
        echo "<td>".$mes."</td>";
        echo "<td align='right'>R$ ".$totalMes ."</td>";
        echo "</tr>";
        $total = $total + $totalMes;
        $cont = 0;
        $totalMes = 0;
        $totalMes = $totalMes + $venda->getValorVenda();
        }else if ($cont == 0){
        $totalMes = $totalMes + $venda->getValorVenda(); }
    if ($tam == $contaVendas){
        $mes = $VendaDAO->devolveMes($data);
        echo "<tr>";
        echo "<td>".$mes."</td>";
        echo "<td align='right'>R$ ".$totalMes."</td>";
        echo "</tr>";
        $total = $total + $totalMes;
        $cont = 0;
        $totalMes = 0;}
        $data = $data1;}
        echo "<tr>
        <td>Total</td>
        <td align='right'>R$ ".$total."</td>
        </tr>";} ?>
    </table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **relatorioVendas.php**

```

<?php
    session_start();
    ?>
<script> //guarda o tipo do select option
function getImposto(what) {
    if (what.selectedIndex != "") {
        var imposto = what.value;
        document.location=('relatorioVendas.php?imposto=' + imposto);} }
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/VendaDAO.class.php");

    $conexao = new Conexao();
    $conexao->conectar();

    $imposto = $_GET["imposto"];
    if (isset ($_POST['relatorioVenda'])) {

```

```

        $dataInicio = $_POST['dataInicio'];
        $dataFim = $_POST['dataFim'];
        $listaVenda = array();
        $VendaDAO = new VendaDAO();
        $listaVenda = $VendaDAO->relatorio($dataInicio, $dataFim);}
    } catch ( Exception $e ) {
        echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Relatório Vendas</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<table>
    <tr>
        <td><strong>Relatório Vendas:</strong></td>
    </tr>
</table>
<table style="padding-bottom:10px;">
    <form action="" method="post" id="relatVendas">
        <tr><td>Com impostos:</td>
        <td><select id="selection" name="tipoDespesa" onChange="getImposto(this);">
            <option value = " " > - </option>
            <option value = "Sim"> Sim </option>
            <option value = "Nao"> Não </option>
        </select></td>
        </tr>
        <tr>
            <td><?php
            if ($imposto){
                if ($imposto == "Nao")
                    $imposto1 = "Não";
                else
                    $imposto1 = $imposto;
                echo "<tr>
                <td>Relatório com imposto:</td>
                <td><input type=text maxlength='50' name='imposto' readonly='readonly' size='15'
                value='".$imposto1."'></td>
                </tr>";?>
                <tr><td>Data Inicio: </td>
                <td><input type='text' maxlength='50' name='dataInicio' size='20' value=""></td></tr>
                <tr><td>Data Fim: </td>
                <td><input type='text' maxlength='50' name='dataFim' size='20' value=""></td></tr>
                <tr>
                <td><input type='submit' value='Mostra relatório' name='relatorioVenda'></td>
            </form>
            </tr>
            <tr>
            <td align="left"><a href="../../index.php"><input type='submit' value='voltar' name='voltar'></a></td>
        </tr>
    </table><?php
    if (isset ($_POST['relatorioVenda'])){
        echo "<table><tr><td><strong>Periodo de ".$dataInicio." até
        ".$dataFim."</strong></td></tr></table>";

```



```

if ($imposto == "Nao"){
    echo "<table border='1' width='40%'>";
    echo "<tr>
    <td align='center'>Data</td>
    <td align='center'>Nro Fatura</td>
    <td align='center'>Valor</td>
    <td align='center'>Foi Paga</td>
    </tr>";
}
else{
    echo "<table border='1' width='60%'>";
    echo "<tr>
    <td align='center'>Data</td>
    <td align='center'>Nro Fatura</td>
    <td align='center'>Valor</td>
    <td align='center'>Imposto</td>
    <td align='center'>Valor Total</td>
    <td align='center'>Foi Paga</td>
    </tr>";}
$total = 0;
$totalValorVenda = 0;
$totalValorImposto = 0;
if (sizeof($listaVenda) != 0)
foreach ($listaVenda as $venda){
    if ($imposto == "Nao"){
        echo "<tr>";
        echo "<td>".$venda->getDataVenda()."</td>";
        echo "<td>".$venda->getIdVenda()."</td>";
        echo "<td align='right'>R$ ".$venda->getValorVenda()."</td>";
        $total = $total + $venda->getValorVenda();
        if ($venda->getFoiPaga())
            echo "<td align='right'>Sim</td>";
        else
            echo "<td align='right'>Nao</td>";
        echo "</tr>";
    }
    else{
        $valorVenda = $venda->getValorVenda()/1.18;
        $valorVenda = number_format($valorVenda, 2, '.', '');
        $valorImposto = $venda->getValorVenda() - $valorVenda;
        echo "<tr>";
        echo "<td>".$venda->getDataVenda()."</td>";
        echo "<td>".$venda->getIdVenda()."</td>";
        echo "<td align='right'>R$ ".$valorVenda. "</td>";
        echo "<td align='right'>R$ ".$valorImposto. "</td>";
        echo "<td align='right'>R$ ".$venda->getValorVenda()."</td>";
        $totalValorVenda = $totalValorVenda + $valorVenda;
        $totalValorImposto = $totalValorImposto + $valorImposto;

        $total = $total + $venda->getValorVenda();
        if ($venda->getFoiPaga())
            echo "<td align='right'>Sim</td>";
        else
            echo "<td align='right'>Nao</td>";
        echo "</tr>";} }
if ($imposto == "Nao"){
    echo "<tr>
    <td colspan='2'>Totais</td>
    <td align='right'>R$ ".$total."</td>
    <td>.</td>
    </tr>";
}
else{

```

```

        echo "<tr>
        <td colspan='2'>Totais</td>
        <td align='right'>R$ ".$totalValorVenda."</td>
        <td align='right'>R$ ".$totalValorImposto."</td>
        <td align='right'>R$ ".$total."</td>
        <td></td>
        </tr>";}}?>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **lucro.php**

```

<?php
session_start();
try {
    require_once("../classes/Conexao.class.php");
    require_once("../dao/VendaDAO.class.php");
    require_once("../dao/DespesaDAO.class.php");

    $conexao = new Conexao();
    $conexao->conectar();

    if (isset($_POST['relatorioVenda'])){
        $dataInicio = $_POST['dataInicio'];
        $dataFim = $_POST['dataFim'];
        $despesaDAO = new DespesaDAO();
        $vendaDAO = new VendaDAO();
        $totalVendas = $vendaDAO->calculaTotalVendas($dataInicio, $dataFim);
        $totalDespesas = $despesaDAO->calculaTotalDespesas($dataInicio, $dataFim);

        $lucro = $totalVendas - $totalDespesas;}
    } catch (Exception $e) {
        echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Relatório Vendas</title>
<link href="../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<table>
    <tr>
    <td><strong>Lucro:</strong></td>
    </tr>
</table>
<table style="padding-bottom:10px;">
    <tr><td>Data Inicio: </td>
    <td><input type='text' maxlength='50' name='dataInicio' size= '20' value=' '></td></tr>

```

```

        <tr><td>Data Fim: </td>
        <td><input type='text' maxlength='50' name='dataFim' size='20' value=''></td></tr>
        <tr>
        <td><input type='submit' value='Mostra relatório' name='relatorioVenda'></td>
</form>
</tr>
<tr>
<td align="left"><a href=" ../.. /index.php"><input type='submit' value='voltar' name='voltar'></a></td>
</tr>
</table> <?php
if (isset ($_POST['relatorioVenda'])) {
    echo "<table><tr><td><strong>Lucro do periodo de ".$dataInicio." até
    ".$dataFim."</strong></td></tr></table>";
    echo "<table border='1' width='40%'>";
    echo "<tr>
    <td align='center'>Total Vendas</td>
    <td align='center'>Total Despesas</td>
    <td align='center'>Lucro</td>
    </tr>";
    echo "<tr>";
    echo "<td align='center'>R$ ".$totalVendas."</td>";
    echo "<td align='center'>R$ ".$totalDespesas."</td>";
    echo "<td align='center'>R$ ".$lucro."</td>";
    echo "</tr>"; }?>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

Adm/cafe

- **exclusãoCafeVerde.php**

```

<?php
try {
    session_start();
    require_once("../.. /classes/CafeVerde.class.php");
    require_once("../.. /classes/Conexao.class.php");
    require_once("../.. /dao/CafeVerdeDAO.class.php");

    $conexao = new Conexao();
    $cafeVerde = new CafeVerde();
    $cafeVerdeDAO = new CafeVerdeDAO();
    $conexao->conectar();

    if ( isset($_POST["excluiCafeVerde"]) ) {
        $cafeVerde->setId($_POST['idCafeVerde']);
        $cafeVerdeDAO->excluir( $cafeVerde );
        $cafeVerde = new CafeVerde();
        echo "Cafe Verde excluido com sucesso.<br/>";}
    } catch ( Exception $e ) {
        echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

```

```

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Exclusão de Café Verde</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method="POST" id="exclusaoCafeVerde">
  <table>
    <tr>
      <th>Exclusão de Café Verde:</th>
    </tr>
  </table>
  <table>
    <tr>
      <td>ID Café Verde:</td>
      <td><input type="text" maxlength="15" name='idCafeVerde' size= '30' value='<?php
      $cafeVerde->getId() ?>'></td>
      <td><i>(só números)</i></td>
    </tr>
    <tr>
      <td><input type='submit' value='Exclui' name='excluiCafeVerde' '></td>
      <td align="right"><a href="../../index.php"><input type= 'submit' value='voltar'
      name='voltar'></a></td>
    </tr>
  </table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **exclusaoCafeTorrado.php**

```

<?php
  session_start();
?>
<script> //guarda o id do select option
function getStates(what) {
  if (what.selectedIndex != "") {
    var id = what.value;
    document.location=('exclusaoCafeTorrado.php?id=' + id);}}
</script>
<?php
try {
  require_once("../../classes/CafeTorrado.class.php");
  require_once("../../classes/Conexao.class.php");
  require_once("../../dao/CafeTorradoDAO.class.php");

  //coloca na variável $id o valor selecionado no select option (script)
  $id = $_GET["id"];
  $conexao = new Conexao();
  $cafeTorrado = new CafeTorrado();
  $cafeTorradoDAO = new CafeTorradoDAO();
  $conexao->conectar();
  $listaIDs = $cafeTorradoDAO->buscarIDs();

```

```

$listaVazia = false;
if ( count( $listaIDs ) == 0 ){
    $listaVazia = true;
    echo "<p>Nenhum Cafe Torrado cadastrado</p>";

    if ( isset($_POST["excluiCafeTorrado"]) ){
        $cafeTorrado->setId($_POST['idCafeTorrado']);
        $cafeTorradoDAO->excluir( $cafeTorrado );
        $cafeTorrado = new cafeTorrado();
        echo "<p>Cafe Torrado excluído com sucesso.</p>";
    } catch ( Exception $e ) {
        echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Exclusão de Código de Café Torrado</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="exclusaoCafeTorrado">
<table>
<tr>
<th>Exclusão de Café Torrado:</th>
</tr>
</table>
<table>
<tr>
<td>Selecione ID:</td>
<td><select name='idCafeTorrado' onchange='getStates(this);'><?php
if (!$listaVazia){
    ?><option value=" "> - </option><?php
    foreach ($listaIDs as $cafeTorrado){?>
        <option value="<?php echo ($cafeTorrado->getId());?>">
        <?php echo $cafeTorrado->getId();?>
        </option> <?}
    }else{
        echo "Nenhum ID cadastrado";}?>
</select></td><?php
if($id) {
    $cafe = new CafeTorrado();
    $cafe = $cafeTorradoDAO->buscarCafe($id);
    echo "</tr>";
    echo "<tr>";
    echo "<td>ID:</td>";
    echo "<td><input type='text' maxlength='50' name= 'idCafeTorrado' readonly='readonly'
size='30' value='". $id."'></td>";
    echo "</tr>";
    echo "<tr>";
    echo "<td>Nome:</td>";
    echo "<td><input type='text' maxlength='50' name= 'nomeCafeTorrado' readonly='readonly'
size='30' value='". $cafe->getNome()."'></td>";
    echo "</tr>";
    echo "<tr>";
    echo "<td>Apresentação:</td>";

```

```

        echo "<td><input type='text' maxlength='50' name=' tamanhoCafeTorrado' readonly='readonly'
        size='30' value='". $cafe->getTamanho()." "></td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Tamanho:</td>";
        echo "<td><input type='text' maxlength='20' name= 'apresentacaoCafeTorrado'
        readonly='readonly' size='30' value='". $cafe->getApresentacao()." "></td>";
        echo "</tr>"; }?>
        <tr>
        <td><input type='submit' value='Exclui' name='excluiCafeTorrado' ></td>
        <td align="right"><a href=" ../index.php"><input type= 'submit' value='voltar'
        name='voltar'></a></td>
        </tr>
    </table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **estoqueCafeVerde.php**

```

<?php
try {
    session_start();
    require_once("../classes/CafeVerde.class.php");
    require_once("../classes/Conexao.class.php");
    require_once("../dao/CafeVerdeDAO.class.php");

    $conexao = new Conexao();
    $cafeVerde = new CafeVerde();
    $cafeVerdeDAO = new CafeVerdeDAO();
    $conexao->conectar();

    if ( isset($_POST["insereCafeVerde"]) ){
        $cafeVerde->setId($_POST['idCafeVerde']);
        $cafeVerde->setEstoque($_POST['estoqueCafeVerde']);
        $cafeVerdeDAO->insereEstoque( $cafeVerde );
        $cafeVerde = new CafeVerde();
        echo "Estoque de Cafe Verde adicionado com sucesso.<br/>"; } catch ( Exception $e ) {
        echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Inserção de Estoque de Café Verde</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action=" method='POST' id="insereCafeVerde">
<table>
    <tr>

```

```

        <th>Insere Estoque de Café Verde:</th>
    </tr>
</table>
<table>
    <tr>
        <td>ID Café Verde:</td>
        <td><input type='text' maxlength="15" name='idCafeVerde' size='30' value='<?php $cafeVerde-
        >getId() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
        <td>Quantidade (em Kilos):</td>
        <td><input type='text' maxlength="15" name='estoqueCafeVerde' size='30' value='<?php $cafeVerde-
        >getEstoque() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
        <td><input type='submit' value='Insere Estoque Cafe Verde' name='insereCafeVerde'></td>

        <td align="right"><a href=".../index.php"><input type='submit' value='voltar'
        name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **diminuiEstoqueCafeTorrado.php**

```

<?php
    session_start();
?>
<script> //guarda o id do select option
function getStates(what) {
    if (what.selectedIndex != "") {
        var id = what.value;
        document.location=('diminuiEstoqueCafeTorrado.php?id=' + id);}
</script>
<?php
try {

    require_once("../classes/Conexao.class.php");
    require_once("../classes/CafeTorrado.class.php");
    require_once("../dao/CafeTorradoDAO.class.php");

    //coloca na variável $id o valor selecionado no select option (script)
    $id = $_GET["id"];

    $conexao = new Conexao();
    $cafeTorrado = new CafeTorrado();
    $cafeTorradoDAO = new CafeTorradoDAO();
    $conexao->conectar();
    $listaIDs = $cafeTorradoDAO->buscarIDs();
    $listaVazia = false;
    if ( count( $listaIDs ) == 0 ){
        $listaVazia = true;

```

```

        echo "<p>Nenhum Cafe Torrado cadastrado</p>";}

    if ( isset($_POST["diminuiCafeTorrado"]) ){
        $cafeTorrado->setId($_POST['idCafeTorrado']);
        $cafeTorrado->setEstoque($_POST['estoqueCafeTorrado']);
        $cafeTorradoDAO->diminuirEstoque( $cafeTorrado );
        $cafeTorrado = new cafeTorrado();
        echo "<p>Cafe Torrado atualizado com sucesso.</p>"; }

} catch ( Exception $e ) {
    echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Diminui estoque de Café Torrado</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="diminuiCafeTorrado">
    <table>
        <tr>
            <th>Diminui Estoque de Café Torrado:</th>
        </tr>
    </table>
    <table>
        <tr>
            <td>Selecione ID:</td>
            <td><select name='idCafeTorrado' onchange='getStates(this);'>
                <?php
                if (!$listaVazia){
                    <?><option value=" "> - </option><?php
                    foreach ($listaIDs as $cafeTorrado){?>
                        <option value="<?php echo $cafeTorrado->getId();?>">
                            <?php echo $cafeTorrado->getId();?>
                        </option><?>
                    }else{
                        echo "Nenhum ID cadastrado";}    ?>
                </select></td><?php
                if($id) {
                    $cafe = new CafeTorrado();
                    $cafe = $cafeTorradoDAO->buscarCafe($id);
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>ID:</td>";
                    echo "<td><input type='text' maxlength='50' name= 'idCafeTorrado'
                    readonly='readonly' size='30' value='". $id."'></td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Nome:</td>";
                    echo "<td><input type='text' maxlength='50' name= 'nomeCafeTorrado'
                    readonly='readonly' size='30' value='". $cafe->getNome()."'></td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Apresentação:</td>";

```



```

        echo "<td><input type='text' maxlength='50' name= 'tamanhoCafeTorrado'
        readonly='readonly' size='30' value='". $cafe->getTamanho()."></td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Tamanho:</td>";
        echo "<td><input type='text' maxlength='20' name= 'apresentacaoCafeTorrado'
        readonly='readonly' size='30' value='". $cafe->getApresentacao()."></td>";
        echo "</tr>";}?">
        <tr>
        <td>Quantidade (em pacotes):</td>
        <td><input type='text' maxlength="15" name= 'estoqueCafeTorrado' size='30'
        value='<?php $cafeTorrado->getEstoque() ?>'></td>
        </tr>
        <tr>
        <td><input type='submit' value='Atualiza Estoque' name=
        'diminuiCafeTorrado'></td>
        <td align="right"><a href=" ../index.php"><input type= 'submit' value='voltar'
        name='voltar'></a></td>
        </tr>
    </table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **cadastroCafeVerde.php**

```

<?php
try {
    session_start();
    require_once("../classes/Conexao.class.php");
    require_once("../classes/CafeVerde.class.php");
    require_once("../dao/CafeVerdeDAO.class.php");

    $conexao = new Conexao();
    $cafeVerde = new CafeVerde();
    $cafeVerdeDAO = new CafeVerdeDAO();
    $conexao->conectar();

    if ( isset($_POST["enviaCadastroCafeVerde"]) ){
        $cafeVerde->setId($_POST['idCafeVerde']);
        $cafeVerde->setNome($_POST['nomeCafeVerde']);
        $cafeVerde->setEstoque($_POST['estoqueCafeVerde']);
        $cafeVerdeDAO->salvar( $cafeVerde );
        $cafeVerde = new cafeVerde();
        echo "Cafe Verde cadastrado com sucesso.<br/>";}
    } catch ( Exception $e ) {
        echo $e->getMessage();}?">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Café Verde</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">

```

```

<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="cadastroCafeVerde">
<table>
    <tr>
        <th>Cadastro Novo Cafe Verde:</th>
    </tr>
</table>
<table>
    <tr>
        <td>ID Cafe Verde:</td>
        <td><input type='text' maxlength="15" name='idCafeVerde' size='30' value='<?php $cafeVerde->getId() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
        <td>Nome:</td>
        <td><input type='text' maxlength="200" name='nomeCafeVerde' size='30' value='<?php $cafeVerde->getNome() ?>'></td>
    </tr>
    <tr>
        <td>Quantidade (em Kilos):</td>
        <td><input type='text' maxlength="200" name='estoqueCafeVerde' size='30' value='<?php $cafeVerde->getEstoque() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
        <td><input type='submit' value='Cadastra' name='enviaCadastroCafeVerde' ></td>
        <td align="right"><a href=".../index.php"><input type='submit' value='voltar' name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **cadastroCafeTorrado.php**

```

<?php
try {
    session_start();
    require_once("../classes/Conexao.class.php");
    require_once("../classes/CafeTorrado.class.php");
    require_once("../dao/CafeTorradoDAO.class.php");

    $conexao = new Conexao();
    $cafeTorrado = new CafeTorrado();
    $cafeTorradoDAO = new CafeTorradoDAO();
    $conexao->conectar();

    if ( isset($_POST["enviaCadastroCafeTorrado"]) ){
        $cafeTorrado->setId($_POST['idCafeTorrado']);
        $cafeTorrado->setNome($_POST['nomeCafeTorrado']);
        $cafeTorrado->setApresentacao($_POST[ 'apresentacaoCafeTorrado']);
    }
}

```

```

        $cafeTorrado->setTamanho($_POST['tamanhoCafeTorrado']);
        $cafeTorrado->setPreco($_POST['precoCafeTorrado']);
        $cafeTorrado->setEstoque(0);
        $cafeTorradoDAO->salvar( $cafeTorrado );
        $cafeTorrado = new cafeTorrado();
        echo "<p>Café Torrado cadastrado com sucesso.</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Café Torrado</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="cadastroCafeTorrado">
<table>
    <tr>
    <th>Cadastro Novo Cafe Torrado:</th>
    </tr>
</table>
<table>
    <tr>
    <td>ID Cafe Torrado:</td>
    <td><input type='text' maxlength="15" name='idCafeTorrado' size='30' value='<?php $cafeTorrado->getId() ?>'></td>
    <td><i>(só números)</i></td>
    </tr>
    <tr>
    <td>Nome:</td>
    <td><input type='text' maxlength="50" name='nomeCafeTorrado' size='30' value='<?php $cafeTorrado->getNome() ?>'></td>
    </tr>
    <tr>
    <td>Apresentação:</td>
    <td><input type='text' maxlength="50" name='apresentacaoCafeTorrado' size='30' value='<?php $cafeTorrado->getNome() ?>'></td>
    <td><i>(Expresso, Passar, Grão)</i></td>
    </tr>
    <tr>
    <td>Tamanho (pacotes de):</td>
    <td><input type='text' maxlength="20" name='tamanhoCafeTorrado' size='30' value='<?php $cafeTorrado->getNome() ?>'></td>
    <td><i>(só números)</i></td>
    </tr>
    <tr>
    <td>Preço:</td>
    <td><input type='text' maxlength="20" name='precoCafeTorrado' size='30' value='<?php $cafeTorrado->getPreco() ?>'></td>
    <td><i>(Ex: 2.00)</i></td>
    </tr>
    <tr>
    <td><input type='submit' value='Cadastra' name='enviaCadastroCafeTorrado'></td>

```

```

        <td align="right"><a href=" .././index.php"><input type='submit' value='voltar'
        name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

Adm/cliente

- **cadastroCliente.php**

```

<?php
try {
    session_start();
    require_once(" .././classes/Conexao.class.php");
    require_once(" .././classes/Cliente.class.php");
    require_once(" .././dao/ClienteDAO.class.php");

    $conexao = new Conexao();
    $cliente = new Cliente();
    $clienteDAO = new ClienteDAO();
    $conexao->conectar();

    if ( isset($_POST["enviaCadastroCliente"]) ){
        $cliente->setId($_POST['idCliente']);
        $cliente->setNome($_POST['nomeCliente']);
        $cliente->setEndereco($_POST['enderecoCliente']);
        $cliente->setTelefone($_POST['telefoneCliente']);
        $cliente->setEmail($_POST['emailCliente']);
        $clienteDAO->salvar( $cliente );
        $cliente = new Cliente();
        echo "<p>Cliente cadastrado com sucesso.</p>"; }

    } catch ( Exception $e ) {
        echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Usuário</title>
<link href=" .././css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action=" method='POST' id="cadastroCliente">
<table>
    <tr>
        <th>Cadastro Clientes:</th>
    </tr>
</table>
</div>
</table>

```

```

<tr>
<td>CPF/CNPJ:</td>
<td><input type='text' maxlength="15" name='idCliente' size='30' value='<?php $cliente->getId()
?>'></td>
<td><i>(só números)</i></td>
</tr>
<tr>
<td>Nome:</td>
<td><input type='text' maxlength="200" name='nomeCliente' size='30' value='<?php $cliente->getNome() ?>'></td>
</tr>
<tr>
<td>Endereço:</td>
<td><input type='text' maxlength="200" name='endereçoCliente' size='30' value='<?php $cliente->getEndereço() ?>'></td>
</tr>
<tr>
<td>Telefone:</td>
<td><input type='text' maxlength="30" name='telefoneCliente' size='30' value='<?php $cliente->getTelefone() ?>'></td>
</tr>
<tr>
<td>Email:</td>
<td><input type='text' maxlength="30" name='emailCliente' size='30' value='<?php $cliente->getEmail() ?>'></td>
<td><i>(Ex: usuario@mail.com)</i></td>
</tr>
<tr>
<td><input type='submit' value='Cadastra' name='enviaCadastroCliente'></td>
<td align="right"><a href=".../index.php"><input type='submit' value='voltar'
name='voltar'></a></td>
</tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **exclusaoCliente.php**

```

<?php
try {
    session_start();
    require_once("../classes/Cliente.class.php");
    require_once("../classes/Conexao.class.php");
    require_once("../dao/ClienteDAO.class.php");

    $conexao = new Conexao();
    $cliente = new Cliente();
    $clienteDAO = new ClienteDAO();
    $conexao->conectar();

    if ( isset($_POST["excluiCliente"]) ){
        $cliente->setId($_POST['idCliente']);
        $clienteDAO->excluir( $cliente );
        $cliente = new cliente();
        echo "Cliente excluido com sucesso.<br/>"; }
}

```

```

} catch ( Exception $e ) {
    echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Exclusão de Cliente</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="exclusaoCliente">
<table>
    <tr>
        <th>Exclusão de Clientes:</th>
    </tr>
</table>
<table>
    <tr>
        <td>CPF/CNPJ:</td>
        <td><input type='text' maxlength="15" name='idCliente' size='30' value= '<?php $cliente->getId()
?>'></td>
    </tr>
    <tr>
        <td><input type='submit' value='Exclui' name='excluiCliente'></td>
        <td align="right"><a href="../../../index.php"><input type='submit' value= 'voltar'
name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **cadastroFuncionario.php**

```

<?php
try {
    session_start();
    require_once("../classes/Conexao.class.php");
    require_once("../classes/Funcionario.class.php");
    require_once("../dao/FuncionarioDAO.class.php");

    $conexao = new Conexao();
    $funcionario = new Funcionario();
    $funcionarioDAO = new FuncionarioDAO();
    $conexao->conectar();

    if ( isset($_POST["enviaCadastroFuncionario"]) ){
        $funcionario->setId($_POST['idFuncionario']);
        $funcionario->setNome($_POST['nomeFuncionario']);
        $funcionario->setEndereco($_POST['enderecoFuncionario']);
        $funcionario->setTelefone($_POST['telefoneFuncionario']);
    }
}

```

```

        $funcionario->setEmail($_POST['emailFuncionario']);
        $funcionario->setTipo($_POST['tipoFuncionario']);
        $funcionarioDAO->salvar( $funcionario );
        $funcionarioDAO->inicializaFolhaPagamento($funcionario);
        $funcionario = new Funcionario();
        echo "<p>Funcionário cadastrado com sucesso.</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Funcionário</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="cadastroFuncionario">
<table>
    <tr>
        <th>Cadastro Funcionário:</th>
    </tr>
</table>
<table>
    <tr>
        <td>CPF:</td>
        <td><input type='text' maxlength="15" name='idFuncionario' size='30' value='<?php $funcionario->getId() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
        <td>Nome:</td>
        <td><input type='text' maxlength="200" name='nomeFuncionario' size='30' value='<?php $funcionario->getNome() ?>'></td>
    </tr>
    <tr>
        <td>Endereço:</td>
        <td><input type='text' maxlength="200" name='enderecoFuncionario' size='30' value='<?php $funcionario->getEndereco() ?>'></td>
    </tr>
    <tr>
        <td>Telefone:</td>
        <td><input type='text' maxlength="30" name='telefoneFuncionario' size='30' value='<?php $funcionario->getTelefone() ?>'></td>
    </tr>
    <tr>
        <td>Email:</td>
        <td><input type='text' maxlength="30" name='emailFuncionario' size='30' value='<?php $funcionario->getEmail() ?>'></td>
        <td><i>(Ex: usuario@mail.com)</i></td>
    </tr>
    <tr>
        <td>Tipo:</td>
        <td><input type='text' maxlength="5" name='tipoFuncionario' size='5' value='<?php $funcionario->getTipo() ?>'></td>
        <td><i>(F ou A)</i></td>

```

```

        </tr>
        <tr>
        <td><input type='submit' value='Cadastra' name= 'enviaCadastroFuncionario'></td>
        <td align="right"><a href=" ../index.php"><input type='submit' value='voltar'
        name='voltar'></a></td>
        </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **exclusaoFuncionario.php**

```

<?php
try {
    session_start();
    require_once("../classes/Funcionario.class.php");
    require_once("../classes/Conexao.class.php");
    require_once("../dao/FuncionarioDAO.class.php");

    $conexao = new Conexao();
    $funcionario = new Funcionario();
    $funcionarioDAO = new FuncionarioDAO();
    $conexao->conectar();

    if ( isset($_POST["excluiFuncionario"]) ){
        $funcionario->setId($_POST['idFuncionario']);
        $funcionarioDAO->excluir($funcionario);
        $funcionarioDAO->excluiFolhaPagamento($funcionario);
        $funcionario = new Funcionario();
        echo "<p>Funcionário excluído com sucesso.</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Exclusão de Funcionário</title>
<link href=" ../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method='POST' id="exclusaoFuncionario">
<table>
        <tr>
        <th>Exclusão de Funcionario:</th>
        </tr>
</table>
<table>
        <tr>
        <td>CPF:</td>

```



```

        <td><input type='text' maxlength="15" name='idFuncionario' size='30' value='<?php $funcionario-
        >getId() ?>'></td>
        <td><i>(só números)</i></td>
    </tr>
    <tr>
    <td><input type='submit' value='Exclui' name='excluiFuncionario'></td>
    <td align="right"><a href=" ../index.php"><input type='submit' value='voltar'
    name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **atualizaFuncionario.php**

```

<?php
try {
    session_start();
    require_once("../classes/Funcionario.class.php");
    require_once("../classes/Conexao.class.php");
    require_once("../dao/FuncionarioDAO.class.php");

    $conexao = new Conexao();
    $funcionario1 = new Funcionario();
    $funcionarioDAO = new FuncionarioDAO();
    $conexao->conectar();

    if ( isset($_POST["procuraFuncionario"]) ){
        $funcionario1->setId($_POST['idFuncionario']);
        $funcionario = $funcionarioDAO->retornaFuncionario($funcionario1->getId());
        $folhaPagamento = $funcionarioDAO->retornaFolhaPagamento($funcionario1->getId());
        $funcionario1 = new Funcionario();
    }
    if (isset ($_POST["atualizaFuncionario"])){
        $funcionario = new Funcionario();
        $folhaPagamento = new FolhaPagamento();
        $funcionario->setId($_POST['idFuncionario']);
        $funcionario->setNome($_POST['nomeFuncionario']);
        $funcionario->setEndereco($_POST['enderecoFuncionario']);
        $funcionario->setTelefone($_POST['telefoneFuncionario']);
        $funcionario->setEmail($_POST['emailFuncionario']);
        $funcionario->setTipo($_POST['tipoFuncionario']);
        $folhaPagamento->setId($_POST['idFuncionario']);
        $folhaPagamento->setSalario($_POST['salario']);
        $folhaPagamento->setHorasExtra($_POST['horasExtra']);
        $folhaPagamento->setFaltas($_POST['faltas']);
        $folhaPagamento->setSeguro($folhaPagamento->getSalario() * 13/100);
        $folhaPagamento->setTotalSalario($folhaPagamento->getSalario() + (15 * $folhaPagamento-
        >getHorasExtra()) - (15 * $folhaPagamento->getFaltas()) + $folhaPagamento->getSeguro());
        $funcionarioDAO->atualizarFuncionario($funcionario);
        $funcionarioDAO->insereFolhaPagamento($folhaPagamento);
        echo "<p>Funcionário atualizado com sucesso</p>";
    }
} catch ( Exception $e ) {
    echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

```

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Atualização de Funcionário</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method="POST" id="procuraFuncionario">
<table>
<tr>
<th>Atualiza Funcionario:</th>
</tr>
</table>
<table>
<tr>
<td>CPF:</td>
<td><input type="text" maxlength="15" name="idFuncionario" size="30" value="<?php $funcionario1-
>getId() ?>"></td>
<td><i>(só números)</i></td>
</tr>
<tr>
<td><input type="submit" value="Procura" name="procuraFuncionario"></td>
</tr>
</form>
<form action="" method="POST" id="atualizaFuncionario"><?php
if ( isset($_POST["procuraFuncionario"]) ){
if ($funcionario){
echo "<tr>
<td>CPF:</td>
<td><input type="text" maxlength='15' name= 'idFuncionario' readonly='readonly'
size='30' value="".$funcionario->getId()."></td>
</tr>
<tr>
<td>Nome:</td>
<td><input type="text" maxlength='15' name= 'nomeFuncionario' size='30'
value="".$funcionario->getNome()."></td>
</tr>
<tr>
<td>Endereço:</td>
<td><input type="text" maxlength='15' name= 'enderecoFuncionario' size='30'
value="".$funcionario->getEndereco()."></td>
</tr>
<tr>
<td>Telefone:</td>
<td><input type="text" maxlength='15' name= 'telefoneFuncionario' size='30'
value="".$funcionario->getTelefone()."></td>
</tr>
<tr>
<td>Email:</td>
<td><input type="text" maxlength='15' name= 'emailFuncionario' size='30'
value="".$funcionario->getEmail()."></td>
</tr>
<tr>
<td>Tipo:</td>

```

```

        <td><input type='text' maxlength='15' name= 'tipoFuncionario' size='30'
        value="" . $funcionario->getTipo()." "></td>
    </tr>
    <tr>
    <td>Salario:</td>
    <td><input type='text' maxlength='15' name='salario' size='30'
    value="" . $folhaPagamento->getSalario()." "></td>
    </tr>
    <tr>
    <td>Horas Extra:</td>
    <td><input type='text' maxlength='15' name='horasExtra' size='30'
    value="" . $folhaPagamento->getHorasExtra()." "></td>
    </tr>
    <tr>
    <td>Faltas:</td>
    <td><input type='text' maxlength='15' name='faltas' size='30'
    value="" . $folhaPagamento->getFaltas()." "></td> </tr>"; } }?>
    <tr>
    <td><input type='submit' value='Atualiza' name= 'atualizaFuncionario'></td>
    <td align="right"><a href="..."><input type='submit' value='voltar'
    name='voltar'></a></td>
    </tr>
    </form>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **cadastroFolhaPagamento.php**

```

<?php
try {
    session_start();
    require_once("../classes/Conexao.class.php");
    require_once("../classes/FolhaPagamento.class.php");
    require_once("../dao/FuncionarioDAO.class.php");

    $conexao = new Conexao();
    $folhaPagamento = new FolhaPagamento();
    $funcionarioDAO = new FuncionarioDAO();
    $conexao->conectar();

    if ( isset($_POST["enviaCadastroFolha"]) ){
        $folhaPagamento->setId($_POST['id']);
        $folhaPagamento->setSalario($_POST['salario']);
        $folhaPagamento->setHorasExtra($_POST['horasExtra']);
        $folhaPagamento->setFaltas($_POST['faltas']);
        $seguro = $folhaPagamento->getSalario() * 13/100;
        $totalSalario = $folhaPagamento->getSalario() + (15 * $folhaPagamento->getHorasExtra()) -
        (15 * $folhaPagamento->getFaltas()) + $seguro;
        $folhaPagamento->setSeguro($seguro);
        $folhaPagamento->setTotalSalario($totalSalario);
        $funcionarioDAO->insereFolhaPagamento($folhaPagamento);
        $folhaPagamento = new FolhaPagamento();
        echo "<p>Folha de pagamento cadastrada com sucesso.</p>"; }
    } catch ( Exception $e ) {
        echo $e->getMessage(); }?>

```

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Folha de Pagamento</title>
<link href="../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action="" method="POST" id="cadastroFuncionario">
<table>
<tr>
<th>Cadastro Folha Pagamento de Funcionário:</th>
</tr>
</table>
<table>
<tr>
<td>CPF:</td>
<td><input type='text' maxlength="15" name='id' size='30' value='<?php $folhaPagamento->getId() ?>'></td>
<td><i>(só números)</i></td>
</tr>
<tr>
<td>Salário:</td>
<td><input type='text' maxlength="200" name='salario' size='30' value='<?php $folhaPagamento->getSalario() ?>'></td>
<td><i>(Separar decimais com ponto. Ex: 1500.00)</i></td>
</tr>
<tr>
<td>Horas Extra:</td>
<td><input type='text' maxlength="200" name='horasExtra' size='30' value='<?php $folhaPagamento->getHorasExtra() ?>'></td>
<td><i>(Colocar horas completas. Número inteiro)</i></td>
</tr>
<tr>
<td>Faltas:</td>
<td><input type='text' maxlength="30" name='faltas' size='30' value='<?php $folhaPagamento->getFaltas() ?>'></td>
<td><i>(Se não tiver nenhuma colocar 0)</i></td>
</tr>
</table>
<table>
<tr>
<td><input type='submit' value='Cadastra Folha Pagamento' name='enviaCadastroFolha'></td>
</tr>
<tr>
<td align="left"><a href="../../index.php"><input type='submit' value='voltar' name='voltar'></a></td>
</tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>

```

```
</html>
```

- **relatorioFolhaPagamento.php**

```
<?php  
try {
```

```
    session_start();  
    require_once("../classes/Conexao.class.php");  
    require_once("../classes/FolhaPagamento.class.php");  
    require_once("../dao/FuncionarioDAO.class.php");
```

```
    $conexao = new Conexao();  
    $funcionarioDAO = new FuncionarioDAO();  
    $conexao->conectar();
```

```
    $listaFuncionarios = $funcionarioDAO->retornaListaFuncionarios();  
} catch (Exception $e) {  
    echo $e->getMessage(); }?>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
<title>Relatório Folha de Pagamento</title>
```

```
<link href="../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
```

```
</head>
```

```
<body class="corpo">
```

```
<div id="background">
```

```
<div id="wrapper">
```

```
<div id="container" class="clearfix">
```

```
<div id="content">
```

```
<table>
```

```
    <tr>
```

```
    <td><strong>Relatório das folhas de pagamento dos funcionários: </strong></td>
```

```
    </tr>
```

```
</table>
```

```
<table border="1" width="65%">
```

```
    <tr align="center">
```

```
    <td>CPF</td>
```

```
    <td>Nome</td>
```

```
    <td>Salário</td>
```

```
    <td>Seguro</td>
```

```
    <td>Horas Extra</td>
```

```
    <td>Faltas</td>
```

```
    <td>Salário Total</td>
```

```
</tr>
```

```
<?php
```

```
foreach($listaFuncionarios as $funcionario){
```

```
    $folhaPagamento = new FolhaPagamento();
```

```
    $folhaPagamento = $funcionarioDAO->retornaFolhaPagamento ($funcionario->getId());
```

```
    echo "<tr>
```

```
    <td>". $funcionario->getId(). "</td>
```

```
    <td>". $funcionario->getNome(). "</td>
```

```
    <td align='right'>R$ ". $folhaPagamento->getSalario(). "</td>
```

```
    <td align='right'>R$ ". $folhaPagamento->getSeguro(). "</td>
```

```
    <td align='center'>". $folhaPagamento->getHorasExtra(). "</td>
```

```
    <td align='center'>". $folhaPagamento->getFaltas(). "</td>
```

```
    <td align='right'>R$ ". $folhaPagamento->getTotalSalario(). "</td>
```

```
    </tr>"; }?>
```

```
</table>
```

```

<table>
  <tr>
    <td align="left"><a href="../../../index.php"><input type='submit' value='voltar' name='voltar'></a></td>
  </tr>
</table>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

Adm/pedido

- **pedido.php**

```

<?php
try {
    require_once("../classes/Conexao.class.php");

    $conexao = new Conexao();
    $conexao->conectar();
} catch ( Exception $e ) {
    echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Pedido de Café Torrado</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<form action='pedido1.php' method='POST' id="pedido">
<table>
  <tr>
    <th>Pedido:</th>
  </tr>
</table>
<table>
  <tr>
    <td>CPF/CNPJ Cliente:</td>
    <td><input type='text' maxlength="15" name='idCliente' size='30' value=""></td>
    <td><i>(só números)</i></td>
  </tr>
  <tr>
    <td><input type='submit' value='Verifica Cliente' name= 'verificaCliente'></td>
    <td align="right"><a href="../../../index.php"><input type='submit' value= 'voltar'
name='voltar'></a></td>
  </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->

```

```

</div><!-- ends background -->
</body>
</html>

```

- **pedido1.php**

```

<?php
    session_start();
?>
<script> //guarda o id do select option
function getStates(what) {
    if (what.selectedIndex != "") {
        var id = what.value;
        document.location=('pedido1.php?id=' + id);}
}
</script>
<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../classes/Cliente.class.php");
    require_once("../classes/ItemPedido.class.php");
    require_once("../classes/CafeTorrado.class.php");
    require_once("../dao/PedidoDAO.class.php");
    require_once("../dao/CafeTorradoDAO.class.php");
    require_once("../dao/CafeVerdeDAO.class.php");
    require_once("../dao/ClienteDAO.class.php");

    //criando objetos das classes
    $conexao = new Conexao();
    $cliente = new Cliente();
    $cafeTorrado = new CafeTorrado();
    $itemPedido = new ItemPedido();
    $clienteDAO = new ClienteDAO();
    $pedidoDAO = new PedidoDAO();
    $cafeTorradoDAO = new CafeTorradoDAO();
    $cafeVerdeDAO = new CafeVerdeDAO();
    $conexao->conectar();

    //verifica se existe o cliente para fazer o pedido
    if ( isset($_POST["idCliente"]) ){
        $cliente->setId($_POST['idCliente']);
        $_SESSION['CLIENTE'] = $_POST['idCliente'];
        $_SESSION['temCliente'] = $clienteDAO->verificaCliente ($_SESSION['CLIENTE']);
        $_SESSION['PEDIDO'] = $pedidoDAO->verificaNroPedido();
        $_SESSION['ORDEM'] = $pedidoDAO->verificaNroOrdem();

        //busca os Ids para preencher o select option
        $listaIDs = $cafeTorradoDAO->buscarIDs();
        $listaVazia = false;
        if ( count( $listaIDs ) == 0 ){
            $listaVazia = true;
            echo "Nenhum Cafe Torrado cadastrado";}

        //coloca na variável $id o valor selecionado no select option (script)
        $id = $_GET["id"];

        //atribui valores ao itemPedido, soma o total de cada item ao valor total
        if ( isset($_POST["itemPedido"]) ){
            $itemPedido->setIdCafe($id);
            $itemPedido->setIdPedido($_SESSION['PEDIDO']);
            $itemPedido->setIdCliente($_SESSION['CLIENTE']);

```



```

        echo "</tr>";
        echo "<tr>";
        echo "<td>Apresentação:</td>";
        echo "<td><input type='text' maxlength='50' name= 'tamanhoCafeTorrado' readonly='readonly'
        value='". $cafe->getTamanho(). "'></td>";
        echo "</tr>";
        echo "<tr>";
        echo "<td>Tamanho:</td>";
        echo "<td><input type='text' maxlength='20' name= 'apresentacaoCafeTorrado'
        readonly='readonly' value='". $cafe->getApresentacao(). "'></td>";
        echo "</tr>"; }?>
        <tr>
        <td>Quantidade (em pacotes):</td>
        <td><input type='text' maxlength="15" name='quantidade' value=""></td>
        </tr>
        <tr>
        <td><input type='submit' value='Adiciona item' name= 'itemPedido'></td>
        <td align="right"><a href="pedido.php">
        <input type='submit' value='voltar' name='voltar'></a></td>
    </tr>
</table>
</form>
<form action="pedido2.php">
    <table>
        <tr>
        <td colspan="2"><input type='submit' value='Termina Venda' name='itemPedido'></td>
        </tr>
    </table>
</form>
<?php }else{
    echo "<table>
    <tr>
    <td align='right'><a href='../cliente/cadastroCliente.php'> <input type='submit' value='voltar'
    name='voltar'></a></td> </tr>
    </table>";?>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

- **pedido2.php**

```

<?php
session_start();
try {
    require_once("../classes/Conexao.class.php");
    require_once("../classes/Pedido.class.php");
    require_once("../classes/Cliente.class.php");
    require_once("../classes/ItemPedido.class.php");
    require_once("../classes/CafeTorrado.class.php");
    require_once("../classes/Venda.class.php");
    require_once("../dao/PedidoDAO.class.php");
    require_once("../dao/CafeTorradoDAO.class.php");
    require_once("../dao/ClienteDAO.class.php");
    require_once("../dao/VendaDAO.class.php");

    $conexao = new Conexao();
    $cliente = new Cliente();

```

```

    $venda = new Venda();
    $vendaDAO = new VendaDAO();
    $itemPedido = new ItemPedido();
    $clienteDAO = new ClienteDAO();
    $pedidoDAO = new PedidoDAO();
    $pedido = new Pedido();
    $cafeTorradoDAO = new CafeTorradoDAO();
    $conexao->conectar();

    $lista = array();
    $cliente = $clienteDAO->retornaCliente($_SESSION['CLIENTE']);
    $pedidoDAO->novoPedido($_SESSION['PEDIDO'], $_SESSION['totalPedido'],
    $_SESSION['ORDEM']);
    $pedido = $pedidoDAO->retornaPedido($_SESSION['PEDIDO']);
    $lista = $pedido->getListaItens();
    $_SESSION['listaPedido'] = $lista;
    $nroFatura = $vendaDAO->retornaNumeroFatura();
    $venda->setIdVenda($nroFatura);
    $venda->setDataVenda($pedido->getData());
    $venda->setValorVenda($pedido->getTotalPedido());
    $venda->setFoiPaga(false);
    $vendaDAO->insereVenda($venda);
} catch ( Exception $e ) {
    echo $e->getMessage(); }?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Café Torrado</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
<table class="dados">
    <tr>
        <td><strong>Cliente:</strong></td>
        <td colspan="5"><?php echo $cliente->getNome(); ?></td>
    </tr>
    <tr>
        <td><strong>Identificação do Pedido:</strong> </td>
        <td colspan="5"><?php echo $pedido->getId(); ?></td>
    </tr>
    <tr>
        <td><strong>Ordem de processamento:</strong> </td>
        <td colspan="5"><?php echo $pedido->getOrdemProcessamento(); ?></td>
    </tr>
    <tr>
        <td><strong>Data do Pedido: </strong></td>
        <td colspan="5"><?php echo $pedido->getData(); ?></td>
    </tr>
</table>
<table class="pedido2">
    <tr class="itens">
        <td><strong>Id Café</strong></td>
        <td><strong>Nome Café</strong></td>
        <td><strong>Tamanho Café</strong></td>
        <td><strong>Apresentação Café</strong></td>

```


- **criaFatura.php**

```

<?php
session_start();

try {
    define('FPDF_FONTPATH', 'font/');
    require('../fpdf/fpdf.php');

    include("../classes/Conexao.class.php");
    require_once("../classes/ItemPedido.class.php");
    require_once("../classes/Pedido.class.php");
    require_once("../classes/Cliente.class.php");
    require_once("../dao/PedidoDAO.class.php");
    require_once("../dao/ClienteDAO.class.php");

    $conexao = new Conexao();
    $cliente = new Cliente();
    $clienteDAO = new ClienteDAO();
    $itemPedido = new ItemPedido();
    $pedidoDAO = new PedidoDAO();
    $pedido = new Pedido();
    $lista = array();
    $conexao->conectar();

    $cliente = $clienteDAO->retornaCliente($_SESSION['CLIENTE']);
    $pedido = $pedidoDAO->retornaPedido($_SESSION['PEDIDO']);
    $lista = $pedido->getListaItens();

    $sql = "SELECT * FROM itemPedido WHERE idPedido= ".$_SESSION ['PEDIDO']. """;
    $busca = mysql_query($sql);
    $pdf = new FPDF();
    $pdf->Open();
    $pdf->AddPage();
    $pdf->SetFont('Arial', 'B', 8);
    $largura = 40;
    $altura = 6;
    $pdf->Cell(20, $altura, 'CPF/CNPJ: ', 0, 0, 'L');
    $pdf->Cell(60, $altura, $cliente->getId(), 0, 0, 'L');
    $pdf->ln($altura);
    $pdf->Cell(20, $altura, 'Cliente: ', 0, 0, 'L');
    $pdf->Cell(60, $altura, $cliente->getNome(), 0, 0, 'L');
    $pdf->ln($altura);
    $pdf->Cell(20, $altura, 'Endereço: ', 0, 0, 'L');
    $pdf->Cell(60, $altura, $cliente->getEndereco(), 0, 0, 'L');
    $pdf->ln(20);
    $pdf->Cell(20, $altura, 'Código Café', 1, 0, 'C');
    $pdf->Cell(60, $altura, 'Produto', 1, 0, 'L');
    $pdf->Cell(20, $altura, 'Qtde', 1, 0, 'C');
    $pdf->Cell(30, $altura, 'Preço', 1, 0, 'C');
    $pdf->Cell(30, $altura, 'Total', 1, 0, 'C');
    $pdf->SetFont('Arial', "", 8);
    foreach ($lista as $itemPedido){
        $pdf->ln($altura);
        $pdf->Cell(20, $altura, $itemPedido->getIdCafe(), 1, 0, 'C');
        $pdf->Cell(60, $altura, $itemPedido->getNomeCafe(), 1, 0, 'L');
        $pdf->Cell(20, $altura, $itemPedido->getQuantidade(), 1, 0, 'C');
        $pdf->Cell(30, $altura, $itemPedido->getTotalItemPedido() /$itemPedido->getQuantidade(),
            1, 0, 'C');
        $pdf->Cell(30, $altura, "R$ ".$itemPedido->getTotalItemPedido() ."00", 1, 0, 'C');}
    $pdf->ln($altura);
}

```

```

$pdf->Cell(20, $altura, ", 1, 0, 'C');
$pdf->Cell(60, $altura, ", 1, 0, 'L');
$pdf->Cell(20, $altura, ", 1, 0, 'C');
$pdf->Cell(30, $altura, 'Total', 1, 0, 'C');
$pdf->Cell(30, $altura, "R$ ".$pedido->getTotalPedido().",00", 1, 0, 'C');
$pdf->Output("fatura.pdf", "D");
} catch ( Exception $e ) {
    echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Cadastro de Café Torrado</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">
<div id="container" class="clearfix">
<div id="content">
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body><?php
session_destroy();?>
</html>

```

- **pagamento.php**

```

<?php
try {
    require_once("../classes/Conexao.class.php");
    require_once("../classes/Venda.class.php");
    require_once("../dao/VendaDAO.class.php");

    $conexao = new Conexao();
    $venda = new Venda();
    $vendaDAO = new VendaDAO();

    $conexao->conectar();

    if ( isset($_POST["idVenda"]) ){
        $venda->setIdVenda($_POST['idVenda']);
        $vendaDAO->confirmaPagamento($venda->getIdVenda());
        echo "<p>Pagamento confirmado com sucesso</p>";
    }
} catch ( Exception $e ) {
    echo $e->getMessage();}?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Pedido de Café Torrado</title>
<link href="../../../css/styles.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body class="corpo">
<div id="background">
<div id="wrapper">

```

```

<div id="container" class="clearfix">
<div id="content">
<form action="" method="POST" id="pedido">
<table>
    <tr>
        <th>Pagamento:</th>
    </tr>
</table>
<table>
    <tr>
        <td>Número Fatura:</td>
        <td><input type='text' maxlength="15" name='idVenda' size='30' value=""></td>
    </tr>
    <tr>
        <td><input type='submit' value='Confirma Pagamento' name='confirmaPagamento'></td>
        <td align="right"><a href="..."><input type='submit' value='voltar'
name='voltar'></a></td>
    </tr>
</table>
</form>
</div><!-- ends content -->
</div><!-- ends container -->
</div><!-- ends wrapper -->
</div><!-- ends background -->
</body>
</html>

```

Classes/

- **Venda.class.php**

```

<?php
class Venda{
    private $idVenda;
    private $dataVenda;
    private $valorVenda;
    private $foiPaga;

    public function Venda(){
    }

    public function getIdVenda(){
        return $this->idVenda;
    }
    public function getDataVenda(){
        return $this->dataVenda;
    }
    public function getValorVenda(){
        return $this->valorVenda;
    }
    public function getFoiPaga(){
        return $this->foiPaga;
    }

    public function setIdVenda( $valor ){
        $this->idVenda = $valor;
    }
    public function setDataVenda( $valor ){
        $this->dataVenda = $valor;
    }
}

```

```

        public function setValorVenda( $valor ){
            $this->valorVenda = $valor;
        }
        public function setFoiPaga( $valor ){
            $this->foiPaga = $valor;
        }
    }
?>

```

- **Tipo.class.php**

```

<?php
class Tipo{
    private $idTipoDespesa;
    private $nomeTipoDespesa;

    public function Tipo(){
    }

    public function getIdTipoDespesa(){
        return $this->idTipoDespesa;
    }
    public function getNomeTipoDespesa(){
        return $this->nomeTipoDespesa;
    }

    public function setIdTipoDespesa( $valor ){
        $this->idTipoDespesa = $valor;
    }
    public function setNomeTipoDespesa( $valor ){
        $this->nomeTipoDespesa = $valor;
    }
}
?>

```

- **Pedido.class.php**

```

<?php
class Pedido{
    private $id;
    private $data;
    private $ordemProcessamento;
    private $listaItens;
    private $totalPedido;

    public function Pedido(){
    }

    public function getId(){
        return $this->id;
    }
    public function getData(){
        return $this->data;
    }
    public function getOrdemProcessamento(){
        return $this->ordemProcessamento;
    }
    public function getListaItens(){
        return $this->listaItens;
    }
}

```

```

public function getTotalPedido(){
    return $this->totalPedido;
}

public function setId( $valor ){
    $this->id = $valor;
}

public function setData( $valor ){
    $this->data = $valor;
}

public function setOrdemProcessamento( $valor ){
    $this->ordemProcessamento = $valor;
}

public function setListaItens( $valor ){
    $this->listaItens = $valor;
}

public function setTotalPedido( $valor ){
    $this->totalPedido = $valor;
}
}
?>

```

- **ItemPedido.class.php**

```

<?php
class ItemPedido{
    private $idItem;
    private $idPedido;
    private $idCliente;
    private $idCafe;
    private $nomeCafe;
    private $tamanhoCafe;
    private $apresentacaoCafe;
    private $quantidade;
    private $totalItemPedido;

    public function ItemPedido(){
    }

    public function getIdItem(){
        return $this->idItem;
    }

    public function getIdPedido(){
        return $this->idPedido;
    }

    public function getIdCafe(){
        return $this->idCafe;
    }

    public function getIdCliente(){
        return $this->idCliente;
    }

    public function getNomeCafe(){
        return $this->nomeCafe;
    }

    public function getTamanhoCafe(){
        return $this->tamanhoCafe;
    }

    public function getApresentacaoCafe(){
        return $this->apresentacaoCafe;
    }
}

```



```

        public function getQuantidade(){
            return $this->quantidade;
        }
        public function getTotalItemPedido(){
            return $this->totalItemPedido;
        }

        public function setIdItem ($valor){
            $this->idItem = $valor;
        }
        public function setIdPedido ($valor){
            $this->idPedido = $valor;
        }
        public function setIdCafe( $valor ){
            $this->idCafe = $valor;
        }
        public function setIdCliente( $valor ){
            $this->idCliente = $valor;
        }
        public function setNomeCafe( $valor ){
            $this->nomeCafe = $valor;
        }
        public function setTamanhoCafe( $valor ){
            $this->tamanhoCafe = $valor;
        }
        public function setApresentacaoCafe( $valor ){
            $this->apresentacaoCafe = $valor;
        }
        public function setQuantidade( $valor ){
            $this->quantidade = $valor;
        }
        public function setTotalItemPedido( $valor ){
            $this->totalItemPedido = $valor;
        }
    }
?>

```

- **Funcionario.class.php**

```

<?php
class Funcionario{
    private $id;
    private $nome;
    private $endereco;
    private $telefone;
    private $email;
    private $tipo;

    public function Funcionario(){
    }

    public function getId(){
        return $this->id;
    }

    public function getNome(){
        return $this->nome;
    }

    public function getEndereco(){
        return $this->endereco;
    }
}

```

```

        public function getTelefone(){
            return $this->telefone;
        }
public function getEmail(){
    return $this->email;
}
public function getTipo(){
    return $this->tipo;
}

public function setId( $valor ){
    $this->id = $valor;
}
public function setNome( $valor ){
    $this->nome = $valor;
}
public function setEndereco( $valor ){
    $this->endereco = $valor;
}
public function setTelefone( $valor ){
    $this->telefone = $valor;
}
public function setEmail( $valor ){
    $this->email = $valor;
}
public function setTipo( $valor ){
    $this->tipo = $valor;
}
}
?>

```

- **FolhaPagamento.class.php**

```

<?php
class FolhaPagamento{
    private $id;
    private $salario;
    private $horasExtra;
    private $faltas;
    private $seguro;
    private $totalSalario;

    public function FolhaPagamento(){
    }

    public function getId(){
        return $this->id;
    }
    public function getSalario(){
        return $this->salario;
    }
    public function getHorasExtra(){
        return $this->horasExtra;
    }
    public function getFaltas(){
        return $this->faltas;
    }
    public function getSeguro(){
        return $this->seguro;
    }
}

```

```

public function getTotalSalario(){
    return $this->totalSalario;
}

public function setId( $valor ){
    $this->id = $valor;
}

public function setSalario( $valor ){
    $this->salario = $valor;
}

public function setHorasExtra( $valor ){
    $this->horasExtra = $valor;
}

public function setFaltas( $valor ){
    $this->faltas = $valor;
}

public function setSeguro( $valor ){
    $this->seguro = $valor;
}

public function setTotalSalario( $valor ){
    $this->totalSalario = $valor;
}
}
?>

```

- **Despesa.class.php**

```

<?php
class Despesa{
    private $idDespesaGeral;
    private $idDespesa;
    private $idTipoDespesa;
    private $nomeDespesa;
    private $dataDespesa;
    private $valorDespesa;

    public function Despesa(){
    }

    public function getIdDespesaGeral(){
        return $this->idDespesaGeral;
    }
    public function getIdDespesa(){
        return $this->idDespesa;
    }
    public function getIdTipoDespesa(){
        return $this->idTipoDespesa;
    }
    public function getNomeDespesa(){
        return $this->nomeDespesa;
    }
    public function getDataDespesa(){
        return $this->dataDespesa;
    }
    public function getValorDespesa(){
        return $this->valorDespesa;
    }

    public function setIdDespesaGeral( $valor ){
        $this->idDespesaGeral = $valor;
    }
}

```

```

    }
    public function setIdDespesa( $valor ){
        $this->idDespesa = $valor;
    }
    public function setIdTipoDespesa( $valor ){
        $this->idTipoDespesa = $valor;
    }
    public function setNomeDespesa( $valor ){
        $this->nomeDespesa = $valor;
    }
    public function setDataDespesa( $valor ){
        $this->dataDespesa = $valor;
    }
    public function setValorDespesa( $valor ){
        $this->valorDespesa = $valor;
    }
}
?>

```

- **Conexao.class.php**

```

<?php
class Conexao{

    private $servidor;
    private $usuario;
    private $senha;
    private $bancoDeDados;

    public function Conexao(){
        $this->setServidor("mysql.inf.ufsc.br");
        $this->setUsuario("giuffra");
        $this->setSenha("senha");
        $this->setBancoDeDados("giuffra");
    }

    public function conectar(){
        $conexao = @mysql_connect($this->getServidor(),$this->getUsuario(),$this->getSenha());
        if ( !$conexao )
            throw new Exception("Não foi possível conectar ao servidor de banco de dados.");
        $conexaoBD = @mysql_select_db($this->getBancoDeDados(), $conexao);
        if ( !$conexaoBD )
            throw new Exception("Não foi possível selecionar o banco de dados.");
    }

    public function getServidor(){
        return $this->servidor;
    }
    public function getUsuario(){
        return $this->usuario;
    }
    public function getSenha(){
        return $this->senha;
    }
    public function getBancoDeDados(){
        return $this->bancoDeDados;
    }

    public function setServidor( $valor ){
        $this->servidor = $valor;
    }
}

```

```

    }
    public function setUsuario( $valor ){
        $this->usuario = $valor;
    }
    public function setSenha( $valor ){
        $this->senha = $valor;
    }
    public function setBancoDeDados( $valor ){
        $this->bancoDeDados = $valor;
    }
}
?>

```

- **Cliente.class.php**

```

<?php
class Cliente{
    private $id;
    private $nome;
    private $endereco;
    private $telefone;
    private $email;

    public function Cliente(){
    }

    public function getId(){
        return $this->id;
    }

    public function getNome(){
        return $this->nome;
    }

    public function getEndereco(){
        return $this->endereco;
    }

    public function getTelefone(){
        return $this->telefone;
    }

    public function getEmail(){
        return $this->email;
    }

    public function setId( $valor ){
        $this->id = $valor;
    }

    public function setNome( $valor ){
        $this->nome = $valor;
    }

    public function setEndereco( $valor ){
        $this->endereco = $valor;
    }

    public function setTelefone( $valor ){
        $this->telefone = $valor;
    }

    public function setEmail( $valor ){
        $this->email = $valor;
    }

}
?>

```

- **CafeVerde.class.php**

```
<?php
class CafeVerde{
    private $id;
    private $nome;
    private $estoque;

    public function CafeVerde(){
    }

    public function getId(){
        return $this->id;
    }
    public function getNome(){
        return $this->nome;
    }
    public function getEstoque(){
        return $this->estoque;
    }

    public function setId( $valor ){
        $this->id = $valor;
    }
    public function setNome( $valor ){
        $this->nome = $valor;
    }
    public function setEstoque( $valor ){
        $this->estoque = $valor;
    }
}
?>
```

- **CafeTorrado.class.php**

```
<?php
class CafeTorrado{
    private $id;
    private $nome;
    private $apresentacao;
    private $tamanho;
    private $preco;
    private $estoque;

    public function CafeTorrado(){
    }

    public function getId(){
        return $this->id;
    }
    public function getNome(){
        return $this->nome;
    }
    public function getApresentacao(){
        return $this->apresentacao;
    }
    public function getTamanho(){
        return $this->tamanho;
    }
    public function getPreco(){
```

```

        return $this->preco;
    }
    public function getEstoque(){
        return $this->estoque;
    }

    public function setId( $valor ){
        $this->id = $valor;
    }
    public function setNome( $valor ){
        $this->nome = $valor;
    }
    public function setApresentacao( $valor ){
        $this->apresentacao = $valor;
    }
    public function setTamanho( $valor ){
        $this->tamanho = $valor;
    }
    public function setPreco( $valor ){
        $this->preco = $valor;
    }
    public function setEstoque( $valor ){
        $this->estoque = $valor;
    }
}
?>

```

Dao/

- **VendaDAO.class.php**

<?php

```

require_once("../classes/Venda.class.php");

class VendaDAO{

    public function VendaDAO(){
    }

    public function retornaNumeroFatura(){
        $sql = "select idVenda from venda order by idVenda desc";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao buscar o número da
fatura.</strong></p>";
            throw new Exception();
        }
        $linha = mysql_fetch_object($retorno);
        $nro = $linha->idVenda;
        return $nro + 1;
    }

    public function insereVenda($venda){
        $sql = "INSERT INTO venda (idVenda, dataVenda, valorTotalVenda, foiPaga)
VALUES ('".addslashes($venda->getIdVenda())."', '".addslashes($venda-
>getDataVenda())."', '".addslashes($venda->getValorVenda())."', '".addslashes($venda->getFoiPaga())."')";

        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao salvar a venda.</strong></p>";

```

```

        throw new Exception();
    }
}

public function confirmaPagamento($id){
    $sql = "SELECT * FROM venda WHERE idVenda = ".$id."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>A venda com esse código não existe.</strong></p>";
        throw new Exception();
    }
    $sql = mysql_query ("UPDATE venda SET foiPaga=true WHERE
idVenda=".$id."");
}

public function relatorio($dataInicio, $dataFim){
    $dataInicio = $this->ValidaData($dataInicio);
    $dataFim = $this->ValidaData($dataFim);
    $sql = "SELECT * FROM venda WHERE dataVenda >= ".$dataInicio." AND
dataVenda <= ".$dataFim." ORDER BY dataVenda";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar as Vendas.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
echo "<p>Não existe nenhuma venda</p>";
    while ( $objeto = mysql_fetch_object($retorno) ){
        $venda = new Venda();
        $data = $this->invert($objeto->dataVenda,"");
        $venda->setIdVenda($objeto->idVenda);
        $venda->setDataVenda($data);
        $venda->setValorVenda($objeto->valorTotalVenda);
        $venda->setFoiPaga($objeto->foiPaga);
        $listaVendas[] = $venda;
    }
    return $listaVendas;
}

public function calculaTotalVendas($dataInicio, $dataFim){
    $listaVendas = array();
    $totalVendas = 0;
    $listaVendas = $this->relatorio($dataInicio, $dataFim);
    foreach ($listaVendas as $vendas){
        $totalVendas = $totalVendas + $vendas->getValorVenda();
    }
    return $totalVendas;
}

public function devolveMes($data){
    $data = explode("/", $data); // fatia a string $dat em pedados, usando / como
referência
    $d = $data[0];
    $m = $data[1];
    $y = $data[2];
    switch ($data[1]) {
        case "01":
            return "Janeiro";
        case "02":

```



```

        return "Fevereiro";
    case "03":
        return "Março";
    case "04":
        return "Abril";
    case "05":
        return "Maio";
    case "06":
        return "Junho";
    case "07":
        return "Julho";
    case "08":
        return "Agosto";
    case "09":
        return "Setembro";
    case "10":
        return "Outubro";
    case "11":
        return "Novembro";
    case "12":
        return "Dezembro";
    }
}

referência public function devolveAno($data){
    $data = explode("/", $data); // fatia a string $dat em pedados, usando / como

    $d = $data[0];
    $m = $data[1];
    $y = $data[2];
    $ano = "20".$data[2];
    return $ano;
}

referência public function verificaMes($data, $data1){
    $data = explode("/", $data); // fatia a string $dat em pedados, usando / como

referência $data1 = explode("/", $data1); // fatia a string $dat em pedados, usando / como

    $d1 = $data1[0];
    $m1 = $data1[1];
    $y1 = $data1[2];
    if ($data[1] === $data1[1]){
        return true;
    }else{
        return false;
    }
}

referência public function verificaAno($data, $data1){
    $data = explode("/", $data); // fatia a string $dat em pedados, usando / como

referência $data1 = explode("/", $data1); // fatia a string $dat em pedados, usando / como

```

```

        $d1 = $data1[0];
        $m1 = $data1[1];
        $y1 = $data1[2];
        if ($data[2] === $data1[2]){
            return true;
        }else{
            return false;
        }
    }

    public function ValidaData($dat){
        $data = explode("/", $dat); // fatia a string $dat em pedacos, usando / como
referência
        $d = $data[0];
        $m = $data[1];
        $y = $data[2];
        $res = checkdate($m,$d,$y); // verifica se a data é válida! // 1 = true (válida) // 0 =
false (inválida)
        if ($res == 1){
        } else {
            echo "<p>data inválida!</p>";
        }
        $data1 = ("".$y."/".$m."/".$d."");
        return $data1;
    }

    public function invert($datainv,$sep){//recebe a data e o separador e inverte a data de
yyyy/mm/dd para dd/mm/yyyy
        $ano=substr("$datainv",0, 4);
        $mes=substr("$datainv",5, 2);
        $dia=substr("$datainv",8, 2);
        $datainv="$dia$sep$mes$sep$ano";
        return $datainv;
    }
}
?>

```

- **PedidoDAO.class.php**

```

<?php
require_once("../classes/Pedido.class.php");

class PedidoDAO{

    public function PedidoDAO(){
    }

    public function adicionaItem( $itemPedido ){
        $this->validar($itemPedido);
        $sql = "INSERT INTO itemPedido (idPedido, idCliente, idCafe, nomeCafe,
TamanhoCafe, apresentacaoCafe, quantidade, totalItem) VALUES (" . addslashes($itemPedido-
>getIdPedido())."," . addslashes($itemPedido->getIdCliente())."," . addslashes($itemPedido-
>getIdCafe())."," . addslashes($itemPedido->getNomeCafe())."," . addslashes($itemPedido-
>getTamanhoCafe())."," . addslashes($itemPedido->getApresentacaoCafe())."," . addslashes($itemPedido-
>getQuantidade())."," . addslashes($itemPedido->getTotalItemPedido()).")";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){

```

```

        echo "<p><strong>Ocorreu um erro ao salvar o item Pedido</strong></p>";
        throw new Exception();
    }
}

public function verificaNroPedido(){
    $sql = "select idPedido from itemPedido order by idPedido desc";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o número do
pedido.</strong></p>";
        throw new Exception();
    }
    $linha = mysql_fetch_object($retorno);
    $nro = $linha->idPedido;
    return $nro + 1;
}

public function novoPedido($nroPedido, $totalPedido, $ordem){
    $data=date("Y/m/d");
    $sql = "INSERT INTO pedido (idPedido, dataPedido, ordemProcessamentoPedido,
totalPedido) VALUES
('".addslashes($nroPedido)."', '".addslashes($data)."', '".addslashes($ordem)."', '".addslashes($totalPedido)."' )";

    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao salvar o pedido.</strong></p>";
        throw new Exception();
    }
}

public function verificaNroOrdem(){
    $sql = "select ordemProcessamentoPedido from pedido order by
ordemProcessamentoPedido desc";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o número de ordem de
processamento.</strong></p>";
        throw new Exception();
    }
    $linha = mysql_fetch_object($retorno);
    $nroOrdem = $linha->ordemProcessamentoPedido;
    return $nroOrdem + 1;
}

public function retornaPedido($nroPedido){
    $itensPedido = array();
    $sql = "SELECT * FROM itemPedido WHERE idPedido = ".$nroPedido."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os
itensPedidos.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return $itensPedido;
    while ( $objeto = mysql_fetch_object($retorno)){
        $itemPedido = new ItemPedido();
        $itemPedido->setIdItem ($objeto->idItem);
    }
}

```

```

        $itemPedido->setIdPedido ($objeto->idPedido);
        $itemPedido->setIdCafe ($objeto->idCafe);
        $itemPedido->setNomeCafe ($objeto->nomeCafe);
        $itemPedido->setTamanhoCafe ($objeto->tamanhoCafe);
        $itemPedido->setApresentacaoCafe ($objeto->apresentacaoCafe);
        $itemPedido->setQuantidade ($objeto->quantidade);
        $itemPedido->setTotalItemPedido ($objeto->totalItem);
        $itensPedido[] = $itemPedido;
    }

    $sql = "SELECT * FROM pedido WHERE idPedido = ".$nroPedido."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o pedido.</strong></p>";
        throw new Exception();
    }
    $objeto = mysql_fetch_object($retorno);
    $pedido = new Pedido();
    $pedido->setId($objeto->idPedido);
    $pedido->setData($objeto->dataPedido);
    $pedido->setOrdemProcessamento($objeto->ordemProcessamentoPedido);
    $pedido->setListaItens($itensPedido);
    $pedido->setTotalPedido($objeto->totalPedido);
    return $pedido;
}

public function validar($itemPedido){
    if ( is_null( $itemPedido->getQuantidade() ) || rtrim( $itemPedido->getQuantidade() )
== "" ){
        echo "<p><strong>O campo quantidade é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $itemPedido->getQuantidade() ) < 1 ){
        echo "<p><strong>O pedido não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($itemPedido->getQuantidade())){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
    $var = filter_var($itemPedido->getQuantidade(), FILTER_VALIDATE_INT);
//retorna o valor de $numero se for inteiro, senão, retorna false
    if (!$var){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
}
}
?>

```

- **FuncionarioDAO.class.php**

```

<?php
require_once("../classes/Funcionario.class.php");
require_once("../classes/FolhaPagamento.class.php");

class FuncionarioDAO{

```

```

public function FuncionarioDAO(){
}

public function salvar( $funcionario ){
    $this->validar($funcionario);
    $sql = "INSERT INTO funcionario (idFuncionario, nomeFuncionario,
enderecoFuncionario, telefoneFuncionario, emailFuncionario, tipoFuncionario) VALUES
('" . addslashes($funcionario->getId()) . "','" . addslashes($funcionario->getNome()) . "','" . addslashes($funcionario-
>getEndereco()) . "','" . addslashes($funcionario->getTelefone()) . "','" . addslashes($funcionario-
>getEmail()) . "','" . addslashes($funcionario->getTipo()) . "')";
    $retorno = mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao salvar o funcionário.</strong></p>";
        throw new Exception();
    }
}

public function atualizarFuncionario( $funcionario ){
    $this->validar($funcionario);
    $sql = mysql_query ("UPDATE funcionario SET nomeFuncionario = ".$funcionario-
>getNome().", enderecoFuncionario = ".$funcionario->getEndereco().", telefoneFuncionario = ".$funcionario-
>getTelefone().", emailFuncionario=".$funcionario->getEmail().", tipoFuncionario=".$funcionario->getTipo()."
WHERE idFuncionario = ".$funcionario->getId().");";
}

public function inicializaFolhaPagamento($funcionario){
    $sql = "INSERT INTO folhaPagamento (idFuncionario, salario, horasExtra, faltas,
seguro, totalSalario) VALUES ('" . addslashes($funcionario-
>getId()) . "','" . addslashes(0) . "','" . addslashes(0) . "','" . addslashes(0) . "','" . addslashes(0) . "','" . addslashes(0) . "')";

    $retorno = mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao guardar dados do
funcionário.</strong></p>";
        throw new Exception();
    }
}

public function excluir ($funcionario){
    $sql = mysql_query("DELETE funcionario FROM funcionario WHERE
idFuncionario=".$funcionario->getId().");";
    $retorno = mysql_affected_rows();
    if ( $retorno <= 0 ){
        echo "<p><strong>Não existe funcionário com esse id.</strong></p>";
        throw new Exception();
    }
}

public function excluiFolhaPagamento ($funcionario){
    $sql = mysql_query("DELETE folhaPagamento FROM folhaPagamento WHERE
idFuncionario=".$funcionario->getId().");";
    $retorno = mysql_affected_rows();
    if ( $retorno <= 0 ){
        echo "<p><strong>Não existe funcionário com esse id.</strong></p>";
        throw new Exception();
    }
}

public function verificaFuncionario($id){
    $sql = "SELECT * FROM funcionario WHERE idFuncionario = ".$id."";
}

```

```

        $resultado = mysql_query($sql) or die("Ocorreu um erro ao buscar o cliente. Tente
novamente mais tarde");
        $numLinhas = mysql_num_rows($resultado);
        if($numLinhas>=1){
            return true;
        }else{
            echo "<p><strong>O funcionário não está cadastrado. Clique em voltar e
faça o cadastro</strong></p>";
            return false;
        }
    }

    public function retornaFuncionario($id){
        $sql = "SELECT * FROM funcionario WHERE idFuncionario = ".$id."";
        $resultado = mysql_query($sql) or die("Ocorreu um erro ao buscar o funcionário.
Tente novamente mais tarde");
        $numLinhas = mysql_num_rows($resultado);
        $objeto = mysql_fetch_object($resultado);
        if($numLinhas>=1){
            $funcionario = new Funcionario();
            $funcionario->setId($objeto->idFuncionario);
            $funcionario->setNome($objeto->nomeFuncionario);
            $funcionario->setEndereco($objeto->enderecoFuncionario);
            $funcionario->setTelefone($objeto->telefoneFuncionario);
            $funcionario->setEmail($objeto->emailFuncionario);
            $funcionario->setTipo($objeto->tipoFuncionario);
            return $funcionario;
        }else{
            echo "<p><strong>O funcionário não está cadastrado. Clique em voltar e
faça o cadastro</strong></p>";
            return false;
        }
    }

    public function retornaListaFuncionarios(){
        $listaFuncionarios = array();
        $sql = "SELECT * FROM funcionario ORDER BY nomeFuncionario";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao buscar os IDs.</strong></p>";
            throw new Exception();
        }
        if ( empty($retorno) )
return $listaFuncionarios;
        while ( $objeto = mysql_fetch_object($retorno) ){
            $funcionario = new Funcionario();
            $funcionario->setId($objeto->idFuncionario);
            $funcionario->setNome($objeto->nomeFuncionario);
            $funcionario->setEndereco($objeto->enderecoFuncionario);
            $funcionario->setTelefone($objeto->telefoneFuncionario);
            $funcionario->setEmail($objeto->emailFuncionario);
            $funcionario->setTipo($objeto->tipoFuncionario);
            $listaFuncionarios[] = $funcionario;
        }
        return $listaFuncionarios;
    }

    public function validar($funcionario){
        if ( is_null( $funcionario->getId() ) || rtrim( $funcionario->getId() ) == "" ){

```

```

        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $funcionario->getNome() ) || rtrim( $funcionario->getNome() ) == "" ){
        echo "<p><strong>O campo nome é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $funcionario->getEndereco() ) || rtrim( $funcionario->getEndereco() ) ==
"" ){
        echo "<p><strong>O campo endereço é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $funcionario->getTelefone() ) || rtrim( $funcionario->getTelefone() ) ==
"" ){
        echo "<p><strong>O campo telefone é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $funcionario->getEmail() ) || rtrim( $funcionario->getEmail() ) == "" ){
        echo "<p><strong>O campo e-mail é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $funcionario->getId() ) ) > 15 ){
        echo "<p><strong>O campo id não pode ultrapassar 15 (quinze)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $funcionario->getNome() ) ) > 200 ){
        echo "<p><strong>O campo nome não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $funcionario->getEndereco() ) ) > 200 ){
        echo "<p><strong>O campo endereço não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $funcionario->getTelefone() ) ) > 30 ){
        echo "<p><strong>O campo telefone não pode ultrapassar 30 (trinta)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $funcionario->getEmail() ) ) > 30 ){
        echo "<p><strong>O campo email não pode ultrapassar 30 (trinta)
caracteres.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($funcionario->getId())){
        echo "<p><strong>O CPF/CNPJ só pode ter números</strong></p>";
        throw new Exception();
    }
}

public function insereFolhaPagamento($folhaPagamento){
    $this->validarFolha($folhaPagamento);
    $sql = mysql_query ("UPDATE folhaPagamento SET salario = ".$folhaPagamento-
>getSalario().", horasExtra = ".$folhaPagamento->getHorasExtra().", faltas = ".$folhaPagamento->getFaltas().",
seguro = ".$folhaPagamento->getSeguro().", totalSalario=".$folhaPagamento->getTotalSalario()." WHERE
idFuncionario = ".$folhaPagamento->getId().");
}

```

```

public function retornaFolhaPagamento($id){
    $sql = "SELECT * FROM folhaPagamento WHERE idFuncionario = ".$id."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os IDs.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return false;
    while ( $objeto = mysql_fetch_object($retorno) ){
        $folhaPagamento = new FolhaPagamento();
        if ( is_null($objeto->idFuncionario) ){
            $folhaPagamento->setId($id);
            $folhaPagamento->setSalario(0.00);
            $folhaPagamento->setHorasExtra(0);
            $folhaPagamento->setFaltas(0);
            $folhaPagamento->setSeguro(0.00);
            $folhaPagamento->setTotalSalario(0.00);
        }else{
            $folhaPagamento->setId($objeto->idFuncionario);
            $folhaPagamento->setSalario($objeto->salario);
            $folhaPagamento->setHorasExtra($objeto->horasExtra);
            $folhaPagamento->setFaltas($objeto->faltas);
            $folhaPagamento->setSeguro($objeto->seguro);
            $folhaPagamento->setTotalSalario($objeto->totalSalario);
        }
    }
    return $folhaPagamento;
}

public function validarFolha($folhaPagamento){
    if ( is_null( $folhaPagamento->getId() ) || rtrim( $folhaPagamento->getId() ) == "" ){
        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $folhaPagamento->getSalario() ) || rtrim( $folhaPagamento->getSalario() )
== "" ){
        echo "<p><strong>O campo salario é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $folhaPagamento->getHorasExtra() ) || rtrim( $folhaPagamento-
>getHorasExtra() ) == "" ){
        echo "<p><strong>O campo horas extra é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $folhaPagamento->getFaltas() ) || rtrim( $folhaPagamento->getFaltas() )
== "" ){
        echo "<p><strong>O campo faltas é obrigatório.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($folhaPagamento->getId())){
        echo "<p><strong>O CPF/CNPJ só pode ter números</strong></p>";
        throw new Exception();
    }
}

}

?>

```


- **DespesaDAO.class.php**

<?php

```
require_once("../classes/Despesa.class.php");
require_once("../classes/Tipo.class.php");
```

```
class DespesaDAO{
```

```
    public function DespesaDAO(){
    }
```

```
    public function guardaDados($tipo){
        $sql = "INSERT INTO Dados (NomeDados) VALUES ('.addslashes($tipo).')";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao salvar o dado</strong></p>";
            throw new Exception();
        }
    }
}
```

/*

```
public function pegaIdTipo(){
    $sql = "SELECT NomeDados FROM Dados ORDER BY idDados desc";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o dado.</strong></p>";
        throw new Exception();
    }
    $linha = mysql_fetch_object($retorno);
    $idTipo = $linha->NomeDados;
    return $idTipo;
}
```

```
*/
```

```
public function pegaIdTipo1(){
    $sql = "SELECT NomeDados FROM Dados ORDER BY idDados desc";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o dado.</strong></p>";
        throw new Exception();
    }
}
```

```
$cont = 0;
while ($linha = mysql_fetch_object($retorno)){
    if ($cont == 0 &&($linha->NomeDados == 1 || $linha->NomeDados == 2 ||
$linha->NomeDados == 3)){
        $cont = $cont + 1;
        $idTipo = $linha->NomeDados;
```

```
    }
}
return $idTipo;
}
```

```
public function buscarDespesas($idTipo){
    $listaDespesas = array();
    $sql = "SELECT * FROM despesa WHERE idTipoDespesa='".$idTipo.'" ";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
```

```

Despesas.</strong></p>";
        echo "<p><strong>Ocorreu um erro ao buscar os IDs das
        throw new Exception();
    }
    if ( empty($retorno) )
return $listaDespesas;
    while ( $objeto = mysql_fetch_object($retorno) ){
        $despesa = new Despesa();
        $despesa->setNomeDespesa ($objeto->nomeDespesa);
        $despesa->setIdDespesa ($objeto->idDespesa);

        $listaDespesas[] = $despesa;
    }
    return $listaDespesas;
}

public function buscarNomesDespesa($idTipo){
    $listaNomes = array();
    $sql = "SELECT nomeDespesa FROM despesa WHERE idTipoDespesa=".".SidTipo.""
";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os IDs das
Despesas.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return $listaNomes;
    while ( $objeto = mysql_fetch_object($retorno) ){
        $despesa = new Despesa();
        $despesa->setNomeDespesa ($objeto->nomeDespesa);

        $listaNomes[] = $despesa;
    }
    return $listaNomes;
}

public function buscarNomesDespesaComId($idDespesa){
    $sql = "SELECT nomeDespesa FROM despesa WHERE idDespesa=".".SidDespesa.""
";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os IDs das
Despesas.</strong></p>";
        throw new Exception();
    }
    while ( $objeto = mysql_fetch_object($retorno) ){
        $nome = $objeto->nomeDespesa;
    }
    return $nome;
}

public function buscarIdDespesa($nome){
    $sql = "SELECT idDespesa FROM despesa WHERE nomeDespesa = ".$nome." ";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os IDs das
Despesas.</strong></p>";
        throw new Exception();
    }
}

```

```

        if ( empty($retorno) )
return $listaNomes;
        while ( $objeto = mysql_fetch_object($retorno) ){
            $despesa = new Despesa();
            $despesa->setIdDespesa ($objeto->idDespesa);

        }
        return $despesa;
    }

    public function buscarTipos(){
        $listaTipos = array();
        $sql = "SELECT * FROM tipoDespesa";
        $retorno = @mysql_query($sql);
        if (!$retorno){
            echo "<p><strong>Ocorreu um erro ao buscar os tipos de
Despesas.</strong></p>";
            throw new Exception();
        }
        if (empty($retorno))
            return $listaTipos;
        while ($objeto = mysql_fetch_object($retorno)){
            $tipo = new Tipo();
            $tipo->setIdTipoDespesa ($objeto->idTipoDespesa);
            $tipo->setNomeTipoDespesa ($objeto->nomeTipoDespesa);
            $listaTipos[] = $tipo;
        }
        return $listaTipos;
    }

    public function buscaIdTipo($tipo){
        $sql = "SELECT idTipoDespesa FROM tipoDespesa WHERE
nomeTipoDespesa='".$tipo."'";
        $retorno = @mysql_query($sql);
        if ( !$retorno ) {
            echo "<p><strong>Ocorreu um erro ao buscar os IDs das
Despesas.</strong></p>";
            throw new Exception();
        }
        if ( empty($retorno) )
            echo "Não existe nenhuma despesa desse tipo";
        while ( $objeto = mysql_fetch_object($retorno) ){
            $idTipo = $objeto->idTipoDespesa;
        }
        return $idTipo;
    }

    public function buscaDespesasGerais($idDespesa, $dataInicio, $dataFim){
        $dataInicio = $this->ValidaData($dataInicio);
        $dataFim = $this->ValidaData($dataFim);
        $sql = "SELECT * FROM despesasGerais WHERE idDespesa='".$idDespesa.'" AND
dataDespesa >= '".$dataInicio.'" AND dataDespesa <= '".$dataFim.'" ORDER BY dataDespesa DESC";
        $retorno = @mysql_query($sql);
        if ( !$retorno ) {
            echo "<p><strong>Ocorreu um erro ao buscar as despesas
gerais.</strong></p>";
            throw new Exception();
        }
        if ( empty($retorno) )
            echo "Não existe nenhuma despesa desse tipo";
    }

```

```

while ( $objeto = mysql_fetch_object($retorno) ){
    $despesa = new Despesa();
    $data = $this->invert($objeto->dataDespesa,"/");
    $despesa->setIdDespesaGeral($objeto->idDespesaGeral);
    $despesa->setIdDespesa($objeto->idDespesa);
    $despesa->setDataDespesa($data);
    $despesa->setValorDespesa($objeto->valorDespesa);
    $listaGerais[] = $despesa;
}
return $listaGerais;
}

public function calculaTotalDespesas($dataInicio, $dataFim){
    $dataInicio = $this->ValidaData($dataInicio);
    $dataFim = $this->ValidaData($dataFim);
    $sql = "SELECT * FROM despesasGerais WHERE dataDespesa >= ".$dataInicio."
AND dataDespesa <= ".$dataFim." ORDER BY dataDespesa DESC";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar as despesas
gerais.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
echo "Não existe nenhuma despesa desse tipo";
    while ( $objeto = mysql_fetch_object($retorno) ){
        $despesa = new Despesa();
        $data = $this->invert($objeto->dataDespesa,"/");
        $despesa->setIdDespesaGeral($objeto->idDespesaGeral);
        $despesa->setIdDespesa($objeto->idDespesa);
        $despesa->setDataDespesa($data);
        $despesa->setValorDespesa($objeto->valorDespesa);
        $listaGerais[] = $despesa;
    }
    $totalDespesas = 0;
    foreach($listaGerais as $gerais1){
        $totalDespesas = $totalDespesas + $gerais1->getValorDespesa();
    }
    return $totalDespesas;
}

public function adicionaDespesa($despesa){
    $this->validar($despesa);
    $sql = "INSERT INTO despesa (idTipoDespesa, nomeDespesa) VALUES
("".addslashes($despesa->getIdTipoDespesa())."".addslashes($despesa->getNomeDespesa())."");";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao salvar a despesa</strong></p>";
        throw new Exception();
    }
}

public function insereDespesaGeral($despesa){
    $data = $this->ValidaData($despesa->getDataDespesa());
    $this->validarValor($despesa->getValorDespesa());
    $this->validarDG($despesa);
    $sql = "INSERT INTO despesasGerais (idDespesa, dataDespesa, valorDespesa)
VALUES ("".addslashes($despesa->getIdDespesa())."".addslashes($data)."".addslashes($despesa-
>getValorDespesa())."");";
    $retorno = @mysql_query($sql);

```

```

        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao salvar a despesa geral</strong></p>";
            throw new Exception();
        }
    }

    public function buscaDespesaGeral($nome, $data){
        $id = $this->buscarIdDespesa($nome);
        $data = $this->ValidaData($data);
        $sql = "SELECT * FROM despesasGerais WHERE idDespesa='".$id-
>getIdDespesa()." AND dataDespesa = '".$data."'";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao buscar a despesa.</strong></p>";
            throw new Exception();
        }
        if ( empty($retorno) )
            echo "Não existe nenhuma despesa com esse nome";
        while ( $objeto = mysql_fetch_object($retorno) ){
            $despesa = new Despesa();
            $data = $this->invert($objeto->dataDespesa,"/");

            $despesa->setIdDespesaGeral($objeto->idDespesaGeral);

            $despesa->setIdDespesa($objeto->idDespesa);
            $despesa->setDataDespesa($data);
            $despesa->setValorDespesa($objeto->valorDespesa);
            if ($objeto->idDespesaGeral != " ") {
                return $despesa;
            }else{
                echo "<p><strong>Não existe despesa com essa
data.</strong></p>";

                return false;
                throw new Exception();
            }
        }
    }

    public function atualizaDespesaGeral($id, $data, $valor){

        $data = $this->ValidaData($data);
        $this->validarValor($valor);
        $sql = mysql_query ("UPDATE despesasGerais SET dataDespesa='$data' ,
valorDespesa='$valor' WHERE idDespesaGeral='$id'");
    }

    public function validar($despesa){
        if ( is_null( $despesa->getNomeDespesa() ) || rtrim( $despesa->getNomeDespesa() )
== "" ){

            echo "<p><strong>O campo nome é obrigatório.</strong></p>";
            throw new Exception();
        }
    }

    public function validarDG($despesa){
        if ( is_null( $despesa->getDataDespesa() ) || rtrim( $despesa->getDataDespesa() ) ==
"" ){

            echo "<p><strong>O campo Data é obrigatório.</strong></p>";
            throw new Exception();
        }
    }

```

```

    if ( is_null( $despesa->getValorDespesa() ) || rtrim( $despesa->getValorDespesa() )
== "" ){
        echo "<p><strong>O campo Data é obrigatório.</strong></p>";
        throw new Exception();
    }
}

public function ValidaData($dat){
    $data = explode("/", $dat); // fatia a string $dat em pedacos, usando / como
referência
    $d = $data[0];
    $m = $data[1];
    $y = $data[2];
    $res = checkdate($m,$d,$y); // verifica se a data é válida! // 1 = true (válida) // 0 =
false (inválida)
    if ($res == 1){
    } else {
        echo "<p>data inválida!</p>";
    }
    $data1 = (".$y."/".$m."/".$d."");
    return $data1;
}

public function invert($datainv,$sep){//recebe a data e o separador e inverte a data de
yyyy/mm/dd para dd/mm/yyyy
    $ano=substr("$datainv",0, 4);
    $mes=substr("$datainv",5, 2);
    $dia=substr("$datainv",8, 2);
    $datainv="$dia$sep$mes$sep$ano";
    return $datainv;
}

public function validarValor($valor){
    if ( is_null( $valor ) || rtrim( $valor ) == "" ){
        echo "<p><strong>O campo valor é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $valor ) < 0){
        echo "<p><strong>O valor não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if ($this->isFloat($valor)){
    }else{
        echo "<p><strong>O campo valor só pode ter números.</strong></p>";
        echo "<p><strong>Se for decimal, não pode ter vírgula. Separe o decimal
com ponto.</strong></p>";
        throw new Exception();
    }

    if(!is_numeric($valor)){
        echo "<p><strong> inteiros</strong></p>";
        throw new Exception();
    }
}

public function isFloat($n){
    return ( $n == strval(floatval($n)) )? true : false;
}

```

```
    }  
  }  
?>
```

- **ClienteDAO.class.php**

```
<?php  
class ClienteDAO{  
  
    private $colunas = array("idCliente","nomeCliente","enderecoCliente","telefoneCliente",  
"emailCliente");  
  
    public function ClienteDAO(){  
    }  
  
    public function salvar( $cliente ){  
        $this->validar($cliente);  
        $sql = "INSERT INTO cliente (idCliente, nomeCliente, enderecoCliente,  
telefoneCliente, emailCliente) VALUES (" .addslashes($cliente->getId())."," .addslashes($cliente->  
>getNome())."," .addslashes($cliente->getEndereco())."," .addslashes($cliente->  
>getTelefone())."," .addslashes($cliente->getEmail()).")";  
        $retorno = mysql_query($sql);  
        if ( !$retorno ){  
            echo "<p><strong>Ocorreu um erro ao salvar o usuário.</strong></p>";  
            throw new Exception();  
        }  
    }  
  
    public function excluir( $cliente){  
        $sql = mysql_query("DELETE cliente FROM cliente WHERE idCliente=" . $cliente->  
>getId().");  
        $retorno = mysql_affected_rows();  
        if ( $retorno <= 0 ){  
            echo "<p><strong>Não existe usuário com esse id.</strong></p>";  
            throw new Exception();  
        }  
    }  
  
    public function verificaCliente($id){  
        $sql = "SELECT * FROM cliente WHERE idCliente = ".$id."";  
        $resultado = mysql_query($sql) or die("Ocorreu um erro ao buscar o cliente. Tente  
novamente mais tarde");  
        $numLinhas = mysql_num_rows($resultado);  
        if($numLinhas>=1){  
            return true;  
        }else{  
            echo "<p><strong>O cliente não está cadastrado. Clique em voltar e faça o  
cadastro</strong></p>";  
            return false;  
        }  
    }  
  
    public function retornaCliente($id){  
        $sql = "SELECT * FROM cliente WHERE idCliente = ".$id."";  
        $resultado = mysql_query($sql) or die("Ocorreu um erro ao buscar o cliente. Tente  
novamente mais tarde");  
        $numLinhas = mysql_num_rows($resultado);  
        $objeto = mysql_fetch_object($resultado);  
        if($numLinhas>=1){  
            $cliente = new Cliente();  
        }  
    }  
}
```

```

        $cliente->setId($objeto->idCliente);
        $cliente->setNome($objeto->nomeCliente);
        $cliente->setEndereco($objeto->enderecoCliente);
        $cliente->setTelefone($objeto->telefoneCliente);
        $cliente->setEmail($objeto->emailCliente);
        return $cliente;
    }else{
        echo "<p><strong>O cliente não está cadastrado. Clique em voltar e faça o
cadastro</strong></p>";
    }
}

public function validar($cliente){
    if ( is_null( $cliente->getId() ) || rtrim( $cliente->getId() ) == "" ){
        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cliente->getNome() ) || rtrim( $cliente->getNome() ) == "" ){
        echo "<p><strong>O campo nome é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cliente->getEndereco() ) || rtrim( $cliente->getEndereco() ) == "" ){
        echo "<p><strong>O campo endereço é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cliente->getTelefone() ) || rtrim( $cliente->getTelefone() ) == "" ){
        echo "<p><strong>O campo telefone é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cliente->getEmail() ) || rtrim( $cliente->getEmail() ) == "" ){
        echo "<p><strong>O campo e-mail é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cliente->getId() ) ) > 15 ){
        echo "<p><strong>O campo id não pode ultrapassar 15 (quinze)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cliente->getNome() ) ) > 200 ){
        echo "<p><strong>O campo nome não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cliente->getEndereco() ) ) > 200 ){
        echo "<p><strong>O campo endereço não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cliente->getTelefone() ) ) > 30 ){
        echo "<p><strong>O campo telefone não pode ultrapassar 30 (trinta)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cliente->getEmail() ) ) > 30 ){
        echo "<p><strong>O campo email não pode ultrapassar 30 (trinta)
caracteres.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cliente->getId())){

```



```

        echo "<p><strong>O CPF/CNPJ só pode ter números</strong></p>";
        throw new Exception();
    }
}
?>

```

- **CafeVerdeDAO.class.php**

```

<?php
require_once("../classes/CafeVerde.class.php");

class CafeVerdeDAO{

    public function CafeVerdeDAO(){
    }

    public function salvar( $cafeVerde ){
        $this->validar($cafeVerde);
        $sql = "SELECT * FROM cafeverde WHERE idCafeVerde = ".$cafeVerde->getId()."";

        $resultado = @mysql_query($sql);
        $numLinhas = mysql_num_rows($resultado);
        if($numLinhas>=1){
            echo "<p><strong>Esse ID de café já foi cadastrado.</strong></p>";
            die();
        }
        $sql = "INSERT INTO cafeverde (idCafeVerde, nomeCafeVerde, estoqueCafeVerde)
VALUES ('".addslashes($cafeVerde->getId())."', '".addslashes($cafeVerde->getNome())."', '".addslashes($cafeVerde->getEstoque())."')";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao salvar o café.</strong></p>";
            throw new Exception();
        }
    }

    public function excluir ($cafeVerde){
        $sql = mysql_query ("SELECT estoqueCafeVerde FROM `cafeverde` WHERE
idCafeVerde = ".$cafeVerde->getID().");
        $objeto = mysql_fetch_object($sql);
        if($objeto->estoqueCafeVerde > 0){
            echo "<p><strong>O cafe ainda tem estoque, não pode ser
excluído.</strong></p>";
            echo "<p><strong><font color=blue>O estoque atual é: ".$objeto->estoqueCafeVerde."</font></strong></p>";
            throw new Exception();
        }
        $sql = mysql_query("DELETE cafeverde FROM cafeverde WHERE
idCafeVerde=".$cafeVerde->getId().");
        $retorno = mysql_affected_rows();
        if ( $retorno <= 0 ){
            echo "<p><strong>Não existe Cafe Verde com esse id.</strong></p>";
            throw new Exception();
        }
    }
}

```

```

public function inserirEstoque ($cafeVerde){
    $sql = "SELECT * FROM cafeverde WHERE idCafeVerde = ".$cafeVerde-
>getId()."";

    $resultado = @mysql_query($sql);
    $numLinhas = mysql_num_rows($resultado);
    if($numLinhas<1){
        echo "<p><strong>Não existe café verde com esse ID</strong></p>";
        throw new Exception();
    }
    $this->validarEstoque($cafeVerde);
    $sql = mysql_query ("UPDATE cafeverde SET estoqueCafeVerde=
(estoqueCafeVerde + ".$cafeVerde->getEstoque().") WHERE idCafeVerde=".$cafeVerde->getId().");" );
}

public function diminuirEstoqueVerde ($itemPedido){
    $id = $itemPedido->getIdCafe();
    if (($id == 1)||($id==2))
        $id = 1;
    if (($id == 3)||($id==4)||($id == 7)||($id==8))
        $id = 2;
    if (($id == 5)||($id==6))
        $id = 3;
    $sql = mysql_query ("SELECT estoqueCafeVerde FROM `cafeverde` WHERE
idCafeVerde = ".$id."");
    $objeto = mysql_fetch_object($sql);
    if($objeto->estoqueCafeVerde >= $itemPedido->getQuantidade())
        $sql = mysql_query ("UPDATE cafeverde SET estoqueCafeVerde=
(estoqueCafeVerde - ".$itemPedido->getQuantidade().") WHERE idCafeVerde=".$id."");
    else{
        echo "<p><strong>O café não tem estoque suficiente, o pedido não pode ser
realizado.</strong></p>";
        echo "<p><strong><font color=blue>O estoque atual é: ".$objeto-
>estoqueCafeVerde."</font></strong></p>";
        throw new Exception();
    }
}

public function estoque(){
    $listaEstoque = array();
    $sql = "SELECT * FROM cafeverde ORDER BY idCafeVerde";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os estoques de café
verde.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return $listaEstoque;
    while ( $objeto = mysql_fetch_object($retorno) ){
        $cafeVerde = new CafeVerde();
        $cafeVerde->setId ( $objeto->idCafeVerde);
        $cafeVerde->setNome ( $objeto->nomeCafeVerde);
        $cafeVerde->setEstoque ( $objeto->estoqueCafeVerde);
        $listaEstoque[] = $cafeVerde;
    }
    return $listaEstoque;
}

public function validar($cafeVerde){
    if ( is_null( $cafeVerde->getId() ) || rtrim( $cafeVerde->getId() ) == "" ){

```

```

        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cafeVerde->getNome() ) || rtrim( $cafeVerde->getNome() ) == "" ){
        echo "<p><strong>O campo nome é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cafeVerde->getEstoque() ) || rtrim( $cafeVerde->getEstoque() ) == "" ){
        echo "<p><strong>O campo estoque é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cafeVerde->getId() ) ) > 15 ){
        echo "<p><strong>O campo id não pode ultrapassar 15 (quinze)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cafeVerde->getNome() ) ) > 200 ){
        echo "<p><strong>O campo nome não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $cafeVerde->getEstoque() ) < 1){
        echo "<p><strong>A quantidade não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cafeVerde->getEstoque())){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
    $var = filter_var($cafeVerde->getEstoque(), FILTER_VALIDATE_INT); //retorna o
valor de $numero se for inteiro, senão, retorna false
    if (!$var){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
}

public function validarEstoque($cafeVerde){
    if ( is_null( $cafeVerde->getId() ) || rtrim( $cafeVerde->getId() ) == "" ){
        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cafeVerde->getEstoque() ) || rtrim( $cafeVerde->getEstoque() ) == "" ){
        echo "<p><strong>O campo estoque é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $cafeVerde->getEstoque() ) < 1){
        echo "<p><strong>A quantidade não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cafeVerde->getEstoque())){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
    $var = filter_var($cafeVerde->getEstoque(), FILTER_VALIDATE_INT); //retorna o
valor de $numero se for inteiro, senão, retorna false
    if (!$var){

```

```

        }
        }
    }
}
?>

```

- **CafeTorradoDAO.class.php**

```

<?php
require_once("../classes/CafeTorrado.class.php");

class CafeTorradoDAO{

    public function CafeTorradoDAO(){
    }

    public function salvar( $cafeTorrado ){
        $this->validar($cafeTorrado);
        $sql = "INSERT INTO cafetorrado (idCafeTorrado, nomeCafeTorrado,
apresentacaoCafeTorrado, tamanhoCafeTorrado, precoCafeTorrado, estoqueCafeTorrado) VALUES
('".addslashes($cafeTorrado->getId())."', '".addslashes($cafeTorrado->getNome())."', '".addslashes($cafeTorrado->getApresentacao())."', '".addslashes($cafeTorrado->getTamanho())."', '".addslashes($cafeTorrado->getPreco())."', '".addslashes($cafeTorrado->getEstoque())."')";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao salvar o café.</strong></p>";
            throw new Exception();
        }
    }

    public function buscarIDs(){
        $listaIDs = array();
        $sql = "SELECT * FROM cafetorrado ORDER BY idCafeTorrado";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){
            echo "<p><strong>Ocorreu um erro ao buscar os IDs.</strong></p>";
            throw new Exception();
        }
        if ( empty($retorno) )
return $listaIDs;
        while ( $objeto = mysql_fetch_object($retorno) ){
            $cafeTorrado = new CafeTorrado();
            $cafeTorrado->setId ( $objeto->idCafeTorrado);
            $cafeTorrado->setNome ( $objeto->nomeCafeTorrado);
            $cafeTorrado->setApresentacao ( $objeto->apresentacaoCafeTorrado);
            $cafeTorrado->setTamanho ( $objeto->tamanhoCafeTorrado);

            $listaIDs[] = $cafeTorrado;
        }
        return $listaIDs;
    }

    public function buscarCafe($id){
        $cafe = new CafeTorrado();
        $sql = "SELECT * FROM cafetorrado WHERE idCafeTorrado=".$id."";
        $retorno = @mysql_query($sql);
        if ( !$retorno ){

```

```

        echo "<p><strong>Ocorreu um erro ao buscar os Cafés.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return $cafeId;
    $objeto = mysql_fetch_object($retorno);
    $cafeTorrado = new CafeTorrado();
    $cafeTorrado->setId ($objeto->idCafeTorrado);
    $cafeTorrado->setNome ($objeto->nomeCafeTorrado);
    $cafeTorrado->setApresentacao ($objeto->apresentacaoCafeTorrado);
    $cafeTorrado->setTamanho ($objeto->tamanhoCafeTorrado);

    $cafe = $cafeTorrado;
    return $cafe;
}

public function retornaPreco($idCafe){
    $sql = "SELECT precoCafeTorrado FROM cafetorrado WHERE
idCafeTorrado=".$idCafe."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o preço.</strong></p>";
        throw new Exception();
    }
    $objeto = mysql_fetch_object($retorno);
    $preco = $objeto->precoCafeTorrado;
    return $preco;
}

public function calculaTotalItem($idCafe, $quantidade){
    $sql = "SELECT precoCafeTorrado FROM cafetorrado WHERE
idCafeTorrado=".$idCafe."";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar o preço do
café.</strong></p>";
        throw new Exception();
    }
    $objeto = mysql_fetch_object($retorno);
    $total = ($objeto->precoCafeTorrado * $quantidade);
    return $total;
}

public function aumentaEstoqueTorrado($itemPedido){
    $sql = mysql_query ("UPDATE cafetorrado SET estoqueCafeTorrado=
(estoqueCafeTorrado + ".$itemPedido->getQuantidade().") WHERE idCafeTorrado=".$itemPedido-
>getIdCafe().");");
}

public function diminuirEstoque ($cafeTorrado){
    $this->validarEstoque($cafeTorrado);
    $sql = mysql_query ("SELECT estoqueCafeTorrado FROM `cafetorrado` WHERE
idCafeTorrado = ".$cafeTorrado->getId().");");
    $objeto = mysql_fetch_object($sql);
    if($objeto->estoqueCafeTorrado >= $cafeTorrado->getEstoque())
        $sql = mysql_query ("UPDATE cafetorrado SET estoqueCafeTorrado=
(estoqueCafeTorrado - ".$cafeTorrado->getEstoque().") WHERE idCafeTorrado=".$cafeTorrado->getId().");");
    else{
        echo "<p><strong>O café não tem estoque suficiente, não pode ser
atualizado.</strong></p>";
    }
}

```

```

        echo "<p><strong><font color=blue>O estoque atual é: ".$objeto-
>estoqueCafeTorrado."</font></strong></p>";
        throw new Exception();
    }
}

public function excluir ($cafeTorrado){
    $sql = mysql_query ("SELECT estoqueCafeTorrado FROM `cafetorrado` WHERE
idCafeTorrado = ".$cafeTorrado->getId().");
    $objeto = mysql_fetch_object($sql);
    if($objeto->estoqueCafeTorrado > 0){
        echo "<p>O cafe ainda tem estoque, não pode ser excluídoO estoque atual é:
".$objeto->estoqueCafeTorrado."</p>";
        throw new Exception();
    }
    $sql = mysql_query("DELETE cafetorrado FROM cafetorrado WHERE
idCafeTorrado=".$cafeTorrado->getId().");
    $retorno = mysql_affected_rows();
    if ( $retorno <= 0 ){
        echo "<p><strong>Não existe Café Torrado com esse id.</strong></p>";
        throw new Exception();
    }
}

public function estoque(){
    $listaEstoque = array();
    $sql = "SELECT * FROM cafetorrado ORDER BY idCafeTorrado";
    $retorno = @mysql_query($sql);
    if ( !$retorno ){
        echo "<p><strong>Ocorreu um erro ao buscar os estoques de café
torrado.</strong></p>";
        throw new Exception();
    }
    if ( empty($retorno) )
return $listaEstoque;
    while ( $objeto = mysql_fetch_object($retorno) ){
        $cafeTorrado = new CafeTorrado();
        $cafeTorrado->setId ($objeto->idCafeTorrado);
        $cafeTorrado->setNome ($objeto->nomeCafeTorrado);
        $cafeTorrado->setApresentacao ($objeto->apresentacaoCafeTorrado);
        $cafeTorrado->setTamanho ($objeto->tamanhoCafeTorrado);
        $cafeTorrado->setPreco ($objeto->precoCafeTorrado);
        $cafeTorrado->setEstoque ($objeto->estoqueCafeTorrado);
        $listaEstoque[] = $cafeTorrado;
    }
    return $listaEstoque;
}

public function validar($cafeTorrado){
    if ( is_null( $cafeTorrado->getId() ) || rtrim( $cafeTorrado->getId() ) == "" ){
        echo "<p><strong>O campo id é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( is_null( $cafeTorrado->getNome() ) || rtrim( $cafeTorrado->getNome() ) == "" ){
        echo "<p><strong>O campo nome é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( strlen( rtrim( $cafeTorrado->getId() ) ) > 15 ){
        echo "<p><strong>O campo id não pode ultrapassar 15 (quinze)
caracteres.</strong></p>";
    }
}

```

```

        throw new Exception();
    }
    if ( strlen( rtrim( $cafeTorrado->getNome() ) ) > 200 ){
        echo "<p><strong>O campo nome não pode ultrapassar 200 (duzentos)
caracteres.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $cafeTorrado->getTamanho() ) < 0){
        echo "<p><strong>O tamanho não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cafeTorrado->getTamanho())){
        echo "<p><strong>O tamanho só pode ter números inteiros</strong></p>";
        throw new Exception();
    }
    $var = filter_var($cafeTorrado->getTamanho(), FILTER_VALIDATE_INT); //retorna
o valor de $numero se for inteiro, senão, retorna false
    if (!$var){
        echo "<p><strong>O tamanho só pode ter números inteiros</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $cafeTorrado->getId() ) < 0){
        echo "<p><strong>O ID não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cafeTorrado->getId())){
        echo "<p><strong>O ID só pode ter números inteiros</strong></p>";
        throw new Exception();
    }
    $var = filter_var($cafeTorrado->getId(), FILTER_VALIDATE_INT); //retorna o
valor de $numero se for inteiro, senão, retorna false
    if (!$var){
        echo "<p><strong>O ID só pode ter números inteiros</strong></p>";
        throw new Exception();
    }
    if ($this->isFloat($cafeTorrado->getPreco())){
    }else{
        echo "<p><strong>O campo preço não pode ter vírgula. Separe o decimal
com ponto.</strong></p>";
        throw new Exception();
    }
    }
}

public function isFloat($n){
    return ( $n == strval(floatval($n)) )? true : false;
}

public function validarEstoque($cafeTorrado){
    if ( is_null( $cafeTorrado->getEstoque() ) || rtrim( $cafeTorrado->getEstoque() ) == ""
){
        echo "<p><strong>O campo estoque é obrigatório.</strong></p>";
        throw new Exception();
    }
    if ( rtrim( $cafeTorrado->getEstoque() ) < 0){
        echo "<p><strong>A quantidade não pode ser menor que 1.</strong></p>";
        throw new Exception();
    }
    if(!is_numeric($cafeTorrado->getEstoque())){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";

```

```

        throw new Exception();
    }
    $var = filter_var($caféTorrado->getEstoque(), FILTER_VALIDATE_INT); //retorna
o valor de $numero se for inteiro, senão, retorna false
    if (!$var){
        echo "<p><strong>A quantidade só pode ter números
inteiros</strong></p>";
        throw new Exception();
    }
}
}
?>

```

CSS/

- **styles.css**

```

/* CSS Document */
@charset "UTF-8";
/*****

Sistema de Vendas Fábrica de Café
Authors: Cecilia Giuffra

*****/

@import url("reset.css");

/*****GENERAL STYLES*****/

body {
    font:62.5% "Helvetica Neue", Helvetica, Arial, sans-serif;
    background: #adc8ad left top;
    margin: 20px 0 20px 0px;
    text-align: center;
}
#background {

}
#wrapper {
    width:980px;
    margin:0 auto;
}

p {
    font-size:1.2em;
    margin:10px 0;
    line-height:1.6em;
}
a {
    text-decoration: none;
}
table {
    text-align:left;
    font-size:1.2em;
}
/***** Interface *****/
p.tituloAdm{
    font-size: 2.0em;
}

```



```

}
table.inicio{
    margin-top: 15px;
    margin-left: 230px;
    margin-bottom: 20px;
}
table.inicio1{
    margin-left: 230px;
    text-align: center;
}
table.inicio tr.um td.pedido{
    text-align: center;
    padding-bottom:30px;
}
table.inicio tr.um td.cliente{
    padding-left: 100px;
    text-align: center;
    padding-bottom:30px;
}
table.inicio tr.dois td.verde{
    text-align: center;
    padding-bottom:30px;
}
table.inicio tr.dois td.torrado{
    padding-left: 100px;
    text-align: center;
    padding-bottom:30px;
}
table.inicio tr td form.fazer{
    padding-bottom: 5px;
}
table.inicio tr.tres td.esq{
    text-align: center;
}
table.inicio tr.tres td.dir{
    padding-left: 100px;
    text-align: center;
}
table.inicio1 tr td.esq{
    text-align: center;
}
table.inicio1 tr td.dir{
    padding-left: 110px;
    text-align: center;
}

/***** tabela pedido2 *****/

table.dados tr td{
    width: 170px;
}
table.pedido2{
    padding-top: 20px;
}
table.pedido2 tr.itens td{
    width: 150px;
}
table.pedido2 tr.total td{
    padding-top: 20px;
}
}

```

```

/*****tabela relatorio *****/
table.relatorio{
    width:40%;
    padding-top: 20px;
}
table.relatorio tr td{
    border: 1px inset black;
    border-collapse:separate;
    border-spacing: 1px;
}

/***** despesas *****/

#content table.cadastroDespesas tr td.botaoCadastra{
    padding-top: 20px;
}
#content table.cadastroDespesas tr.insereDespesa td{
    padding-top: 20px;
}
select{
    width: 130px;
}
#enviar{
    width: 200px;
}

```

