Universidade Federal de Santa Catarina - UFSC

Departamento de Informática e Estatística - INE

Curso de Sistemas de Informação

SISTEMA WEB PARA GERÊNCIA DE PROJETO SOCIAL

Petterson Sued Trindade

Florianópolis - SC

29/11/2020

UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
DEPARTAMENTO DE INFORMÁTICA ESTATÍSTICA - INE
CURSO DE SISTEMAS DA INFORMAÇÃO

SISTEMA WEB PARA GERÊNCIA DE PROJETOS SOCIAIS

PETTERSON SUED TRINDADE

Trabalho de Conclusão de Curso para a obtenção do grau de Bacharel no Curso de Sistemas de Informação na Universidade Federal de Santa Catarina.

Florianópolis - SC
2020 – 2

PETTERSON SUED TRINDADE

SISTEMA WEB PARA GERÊNCIA DE PROJETOS SOCIAIS

Trabalho de Conclusão de Curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Sistemas de Informação.

_____

Orientador: Professor Dr. Leandro José Komosinski

Universidade Federal de Santa Catarina – INE

Banca Examinadora

_____

Avaliador: Professor Dr. Jean Carlo Rossa Hauck

Universidade Federal de Santa Catarina – INE

_____

Avaliador: Professor Dr. Jose Eduardo De Luca

Universidade Federal de Santa Catarina - INE

“Ninguém comete erro maior do que não fazer nada porque só pode fazer um pouco”

(Edmund Burke)

“Na menor percepção da realidade percebe-se que, seu e quando são, são somente as idéias”

(Petterson Sued Trindade)

Resumo

Apesar de todo o avanço científico e tecnológico conquistado pela civilização humana nos últimos séculos, é bem expressiva e acentuada a condição de desigualdade socioeconômica encontrada nas estruturas da atual civilização. Neste contexto social, a desigualdade socioeconômica é visivelmente uma característica encontrada em todos os cinco continentes. Esse contraste pode ser visto na sociedade da seguinte forma, em grandes médias e pequenas cidades, enquanto no centro e arredores próximos aos centros, os moradores vivem assistidos por condições favoráveis de existência, nos subúrbios e periferias a alta densidade demográfica, junto com a má distribuição de renda, produzida pela desigualdade socioeconômica, tem criado uma quantidade grande de favelas. É uma população que vive abaixo da linha da pobreza, sem a mínima condição de subsistência. Nesse contexto de marginalizados pelos conceitos e convenções que sustém os objetivos atuais da presente sociedade, a ausência do Estado e serviços públicos é nítida. Isso faz com que nasça da própria sociedade civil, instituições e organizações que tentam amenizar esta situação. No entanto, muitas destas organizações na labuta desta nobre missão, deparam-se com a necessidade de sistematizar seus processos, para melhor gerenciar-se. Este trabalho tem então, a condição de estar sendo feito sobre demanda, comportando as particularidades da Associação Pró Brejaru, encontrada na comunidade Frei Damião.

Palavras chave: Servidor Web, Associação Pró Brejaru, Processo, Sistema web,

SUMÁRIO

Lista de abreviaturas e siglas

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

JSF – Java Server Faces

MVC – Modelo-Visão-Controlador

SSL – Secure Socket Layer

TCC – Trabalho de Conclusão de Curso

UFSC – Universidade Federal de Santa Catarina

OS – Organizações Sociais

USA - Estados Unidos da América

J2EE - Java 2 Enterprise Edition

Lista de Figuras

1 Introdução

Atualmente é imprescindível a utilização das Tecnologias da Informação para manter relações socioeconômica entre sociedade civil e instituições, seja ela privada ou pública, é o vem sendo observado por pesquisadores ao longo do tempo, como Alavi e Bergeron entre outros citados abaixo. No contexto atual, a Tecnologia da Informação desempenha tarefa crucial para o alcance dos objetivos das instituições (ALAVI & JOACHIMSTHALER, 1992; BERGERON, BATEU & RAYMOND, 1991). O avanço tecnológico tem exercido papel relevante nos diversos setores da economia de maneira que as organizações necessitam buscar mecanismos adequados diante da nova realidade.

É comum à maior parte dos trabalhadores atualmente o uso de tecnologias na execução de suas atividades e tarefas no âmbito das organizações, devido às mudanças nas relações sociais e institucionais causada pelo avanço da tecnologia. Administrar uma organização sem o auxílio de um sistema de informação é extremamente difícil e trabalhoso. Muitas vezes em pequenas organizações, que por algum motivo ainda não tem seus processos administrativo informatizados, muitos dos dados resultantes das atividades de seus processos são registrados em documentos de papel e armazenados em um conjunto de arquivos. Um dos motivos é caracterizado segundo SOUZA (2004, p 65) porque, os micros e pequenos empresários negligenciam as atividades de planejamento e controle dos seus negócios, considerando-as como uma burocracia desnecessária. Segundo estudos disponibilizados pelo VIIEGEPE(2012). Quanto aos recursos tecnológicos disponíveis, verificou-se que a maioria das empresas ainda não os utiliza, porém apresentam-se dispostas em adquirir tal recurso. Tal estudo afirma que em uma

população de 90 entrevistados, 30% possuem computador, e só 9% dispõem de sistemas de informação ou de programas de automação comercial. A falta de um sistema de informação para gerenciar uma organização, e executar seu processos, acaba gerando alguns problemas como por exemplo, perda de tempo, tanto para armazenar, quanto para pesquisar determinado documento neste conjunto de arquivos, perda de informações tendo em vista que muitos destes documentos podem em algum momento ser extraviado ou deteriorados pela ação do tempo. Além de existir à necessidade crescente de espaço, à medida que a organização e seu conjunto de dados e documentos crescem.

A importância de um sistema para gerenciar documentos e processos em uma pequena empresa, está diretamente ligado à sobrevivência da mesma no sistema ambiente. Como pode ser facilmente constatado através de pesquisas elaboradas pelo SEBRAE. Observa-se, portanto, que dos dois fatores condicionantes de sucesso apontado pelos empresários, ambos estão relacionados à informação e conhecimento, seja interna ou externamente. Em geral, as organizações não estão preparadas para o gerenciamento de seu ativo informacional, originando o acúmulo desnecessário, ou o descarte de informações importante. A execução das atividades cotidianas também é afetada pela deficiência informacional, não havendo uma metodologia ou modelo estabelecido, o funcionário age de forma empírica, ou seja, da forma que acredita estar correto, acarretando em perda de qualidade e gerando retrabalho (SEBRAE, 2010).

Sendo então de extrema importância a uma organização ainda que pequena, um sistema de informação para executar seus processos. De forma que venha garantir

a qualidade dos seus dados e informações ao longo do tempo. Através do próprio armazenamento destes dados e informações em uma base de dados ágil e segura. Permitindo a utilização destes dados e informações de forma muito mais eficiente, preservando a integridade e confiabilidade destes dados e informações, além de providenciar agilidade para executar os processos da organização, e que por consequência, seja mais sistemático e eficiente gerenciar as organizações.

1.1 Motivação

Conhecendo a organização associação Pró Brejaru, foi possível perceber que apesar de seus processos estarem bem definidos e um tanto complexos, não era utilizado um sistema de software para execução ao menos da maior parte deles. Sendo que na execução dos seus processos, eram produzidos muitos dados e informações em papel e armazenados em pastas de arquivos, fazendo com que a integridade e a utilidade de muitos destes dados e informações estejam um tanto limitado ao longo do tempo. Além de exigir que uma quantidade de papel e espaço seja necessário para armazenar estes arquivos. Enquanto por outro lado, dados e informações importantes para a execução eficiente de alguns dos processos críticos da organização, estejam indisponíveis. Considerando tais aspectos, buscou-se no mercado um sistema de informação que pudesse solucionar estes problemas prévios. No entanto, as condições oportunas da organização em relação às necessidades financeira, restringiram a busca à algumas possibilidades de soluções.

Em um primeiro momento, foram consideradas as características dos sistemas disponíveis no mercado, com a real necessidade da organização. Neste aspecto, foi constatado que algumas das particularidades dos processos da organização não

estavam contidos nas funcionalidades providas pelos sistemas. Em contrapartida, os sistemas disponibilizavam funcionalidades que fugiam o escopo dos processos da organização. Em um segundo momento foi levado em consideração a limitação financeira da organização. Nesse aspecto, foi constatado efetivamente que todos os sistemas encontrados só eram possíveis de serem obtidos, através da compra ou licenciamento, o que o atual momento da organização tornaria inoportuno. Sendo assim então, este trabalho visa buscar uma solução para esta organização, levando em consideração estes dois pontos de vista, e sobre demanda específica.

1.2 Objetivo Geral

Especificar, modelar e implementar um sistema WEB para gerenciar projeto social na associação Pró Brejaru. Sistematizar processos relevantes no âmbito operacional, que permita estruturar uma base de dados consistente confiável e informativa aos funcionários da associação.

1.3 Objetivos específicos

1. Especificar um conjunto de requisitos que expresse as necessidades da organização demandante do sistema.

2. Modelar um sistema considerando o conjunto de requisitos especificado pela associação, de forma a servir como diretriz para a implementação

3. Definir arquitetura de sistema para padrão de projeto, que proponha a melhor solução para as demandas dos requisitos do sistema.

4. Validar junto à organização o modelo da arquitetura e a modelagem do sistema da solução do problema.

5. Pesquisar e selecionar um conjunto de ferramentas de código aberto apta a implementar uma solução considerando o modelo arquitetado.

6. Implementar um sistema WEB que atenda na solução da arquitetura e modelagem dos requisitos do sistema.

7. Fazer testes unitários de componentes codificados e de integração entre os módulos do sistema.

1.4 Delimitação do Trabalho

A priori o sistema a ser construído por se tratar de um software sobre demanda, especificamente da necessidade da associação Pró Brejaru é sensato prevenir que os primeiros requisitos a serem codificados, serão aqueles que contemplem as necessidades mais específicas da associação. Considerando os requisitos que levantados junto ao administrador da organização, sejam mais críticos para sistematização dos processos da associação, para que ao término deste trabalho o objetivo seja alcançado. Além dos requisitos e características básicas a um sistema web que necessariamente precisam estar presentes.

1.5 Metodologia

Para que os objetivos deste projeto fossem alcançados, foi adotada uma metodologia de trabalho que consiste nas seguintes etapas:

- Realização de reuniões com o responsável pela definição dos processos na Associação Pró Brejaru, com objetivo de delimitar as necessidades;

- Conhecer as atuais formas de execução dos processos dentro da associação, e delimitar o que pode ser sistematizado e melhorado;

- Levantamento dos requisitos junto aos executores dos processos e modelagem do sistema segundo a definição objetiva para construção do sistema;

- Implementação do sistema web considerando os requisitos previamente definidos e a modelagem especificada do sistema.

- Fazer teste do sistema web considerando os requisitos específicos e modular segundo as características específicas.

- Validar os requisitos implementados junto a associação segundo o executor do processo de forma individual e modular.

2 Fundamentação Teórica

2.1 Uma Organização Social

Partindo de um conceito mais abrangente sobre o próprio termo “organização social”, fazemos referência a **Raymond Firth** que o define como, consiste na ordenação sistemática de relações sociais pelos atos da escolha e decisão. A partir de uma organização social os indivíduos fazem escolhas baseando-se nas estruturas sociais. Implica algum grau de unificação, ou união de diversos elementos numa relação comum” (IANNI, 1973, p. 41).

No entanto se especificarmos o conceito segundo os objetivos deste trabalho, os termos são definidos segundo **Eurico de Andrade Azevedo e Meirelles**, A organização social é uma qualificação, um título, que a Administração outorga a uma entidade privada, sem fins lucrativos, para que ela possa receber determinados benefícios do Poder Público (dotações orçamentárias, isenções fiscais etc.), para a realização de seus fins, que devem ser necessariamente de interesse da comunidade (Meirelles, 2007, p. 383). Comungado por outros autores como por exemplo, **Modesto** em, A denominação organização social é um enunciado elíptico. Denominam-se sinteticamente organizações sociais as entidades privadas, fundações ou associações sem fins lucrativos, que usufruem do título de organização social (Modesto, 1997, p. 199).

2.2 Processos de uma Organização Social

As organizações sociais em geral tem sido vista com bons olhos pelos executivos de governo. Pois fomentam boas parcerias, tendo em vista que executam serviços públicos mesmo sendo privada. Como pode ser observado no Caderno do Ministério da Administração Federal e Reforma do Estado, As OS são um modelo de parceria entre o Estado e a sociedade. O Estado continuará a fomentar as atividades publicizadas e exercerá sobre elas um controle estratégico: demandará resultados necessários ao atingimento dos objetivos das políticas públicas. O contrato de gestão é o instrumento que regulará as ações das OS (MARE, 1998, p. 13). As OS tornam mais fácil e direto o controle social, por meio da participação nos conselhos

de administração dos diversos segmentos representativos da sociedade civil, ao mesmo tempo que favorece seu financiamento via compra de serviços e doações por parte da sociedade. Não obstante, gozam de uma autonomia administrativa muito maior do que aquela possível dentro do aparelho do Estado. Em compensação, seus dirigentes são chamados a assumir uma responsabilidade maior, em conjunto com a sociedade, na gestão da instituição e na melhoria da eficiência e da qualidade dos serviços, atendendo melhor o cidadão-cliente a um custo menor (MARE, 1998, p. 14).

2.3 Web 2.0

A web 2.0 nada mais é do que uma espécie de expansão, de provisão de novos serviços e conceitos adicionados a tradicional web 1.1, cuja estrutura básica de comunicação e serviços é a internet. Este termo surgiu em 2003 nos USA através da empresa O'Reilly e de seu diretor que definiu que: "Web 2.0 é a mudança para uma internet como plataforma, e um entendimento das regras para obter sucesso nesta nova plataforma. Entre outras, a regra mais importante é desenvolver aplicativos que aproveitem os efeitos de rede para se tornarem melhores quanto mais são usados pelas pessoas, aproveitando a inteligência coletiva" - Tim O'Reilly. Em si, a web 2.0 é o meio pelo qual a interação entre softwares e pessoas tornou-se mais simples, usufruída e eficiênte.

2.4 Internet

Uma rede de computadores é formada por um ou mais computadores conectados um ao outro por meio de transmissão, sendo capaz de trocar informações e compartilhar recursos. Estas são constituídas por um grupo de módulos processadores onde qualquer dispositivo é capaz de notificar através do sistema de comunicação por troca de dados(Tanenbaum, 2003).

Segundo a história a Internet nasceu de um projeto de pesquisa militar (**ARPA: *Advanced Research Projects Agency***), no período da guerra fria, no final dos anos cinqüenta e início dos anos sessenta (Lígia Maria Ribeiro, 2014). No início conectou apenas quatro universidades dentro do território dos Estados Unidos, posteriormente conectou junto a elas uma universidade na Inglaterra. Atualmente de forma

convencional, a internet é o conjunto de redes de computadores que, espalhados por todas as regiões do planeta, conseguem trocar dados e mensagens usando um protocolo comum.

No atual estágio de desenvolvimento das tecnologias é conhecido de todos subjetivamente e ao menos aqueles que atuam na área da TI objetivamente, a fundamental importância da internet para as relações sociais atuais. Tanto do ponto de vista social, quanto econômico. Praticamente tudo hoje é possível fazer por meio da internet, desde requerimentos ao governo, como compras em sites e-comerce, etc.

A internet vem transformando as formas de relações sociais e econômicas, tanto dos indivíduos, quanto das instituições, seja ela pública ou privada, por todo o mundo. Está em toda parte e influencia a vida da maior parte da sociedade, nos mais variados seguimento sociais. Na verdade não é só pela internet, mas pelo conjunto atual das tecnologias. Que podemos citar como exemplo, não mais unicamente o computador, mas também os dispositivos móveis, tablet, smartphones, celulares, etc. Tanto que atualmente as normas de usabilidades de sistemas web, preveem na métrica de disponibilidade, que a construção de sistemas web deve considerar sempre a possibilidade de acesso não só através do computador, mas também e principalmente pelos dispositivos móveis, tendo em vista que estudos apontam estes dispositivos como os mais utilizados para acessar a internet. Segundo algumas fontes específicas e extremamente relevantes, como por exemplo, um estudo recente da União Internacional de Telecomunicações (UIT), agência das Nações Unidas, apontou que mais da metade da população mundial está conectada à internet. São 3,9 bilhões de pessoas (o equivalente a 51% da população mundial) ligadas à rede. Em dezembro de 2017 a quantidade de usuários da internet era acima de 3,9 bilhões de pessoas no mundo Tracto(2017). Sendo que a quantidade do percentual de crescimento ao longo dos anos pode ser observado no gráfico abaixo.

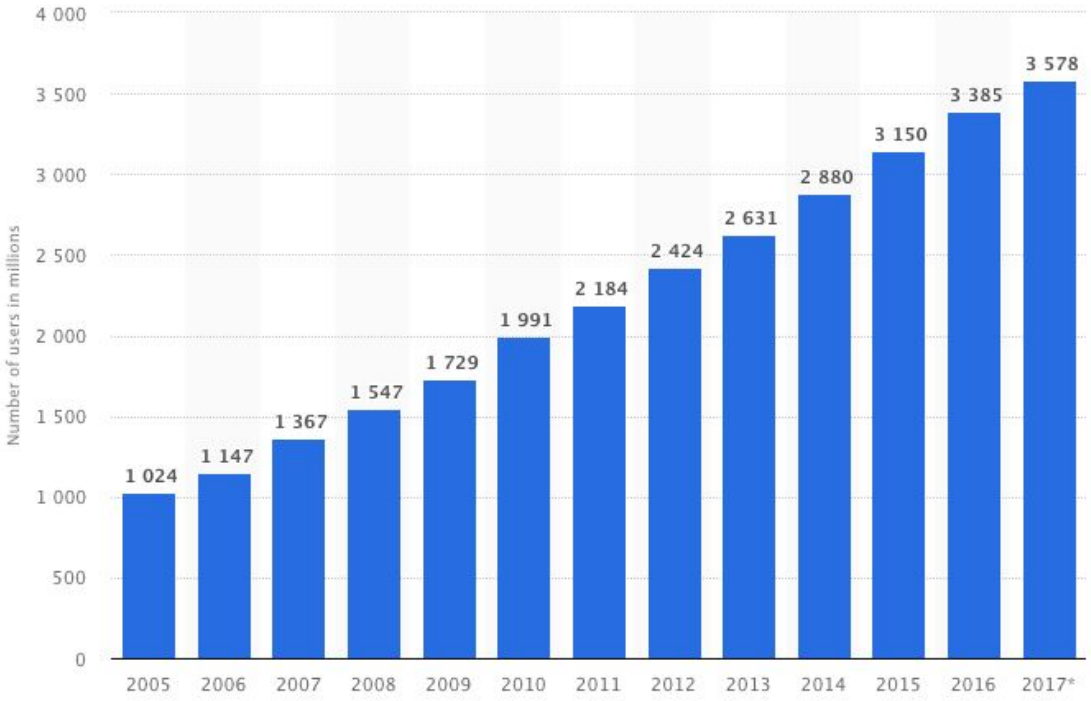


Figura 1: Gráfico do percentual de crescimento dos usuários da internet ao longo dos anos Tracto(2017).

2.5 Sistema web

Para Press (1999, p.13), “se considerarmos os sistemas de processamento em lote, os sistemas de compartilhamento de tempo e as aplicações cliente-servidor como as três primeiras gerações de processamento de dados empresariais, a quarta geração é a dos Sistemas de Informação baseados na Web”, os quais são caracterizados pela universalização do acesso às redes de computadores e pela utilização de sistemas de padrões abertos para a comunicação. Várias denominações têm sido dadas a tais sistemas, como: Web Sites, Web-Based Information Systems (WBIS), Web Information Systems (WIS), Sistemas Web, Aplicações Web e Sistemas de Informação Web. Neste trabalho, serão chamados de Sistemas de Informação Baseados na Tecnologia Web (SIW).

3 Estado da Arte

A princípio considera a questão do quanto é necessário tentar atualizar as condições atuais da web 2.0, devido a complexidade de se tratar o assunto, da própria velocidade de transformação das tecnologias por ela contemplada, e do conhecimento da realidade das tecnologias por aqueles que irão julgar este trabalho. Considerando estas primícias, fez-se necessário tratar aqui a questão de forma objetiva, mostrando as condições atuais dos sistemas de gerenciamento de associações e projetos sociais, e as causas que determinaram a necessidade de projeção deste presente sistema e que está completamente vinculada às necessidades particulares da associação Pró Brejaru. Neste contexto, foi trabalhado apenas com sistemas web com tais características, ficando de fora sistemas web não objetivo, que não apresentem características relevantes nesse sentido.

A pesquisa foi feita na internet através do google sobre algumas primícias: primeiro, o sistema web deveria ser gratuito; segundo, contemplar o cadastro de famílias da comunidade, as quais mensalmente são beneficiadas com doações de cestas-básicas, cadastrar os alunos participantes na associação, além de outros; terceiro, o sistema web deveria conter como introdução de interface um site com o objetivo de promover as ações da associação. Considerando essas três primícias, após a conclusão da pesquisa evidenciou-se que, todos os sistemas web encontrado na busca, nem um contemplava estas três características. Abaixo segue dois sistemas encontrado na busca. Por eliminação, já ambos gratuito;

- **Ongfácil http://portalongfacil.com.br/**

Faz uma espécie de elo entre empresas, ongs e o governo em alto nível. Apesar de disponibilizar o uso de parte do seu sistema web gratuito para ongs através da internet, o site não seria integrado ao sistema, assim também não seria possível prover relatórios específicos do módulo operacional.

- **Observatório do terceiro setor: TechSoup Brasil**

Segue praticamente a mesma linha da empresa e do software anterior, considerando pouca especificidade. No entanto, o site não seria integrado ao

sistema, assim também não seria possível prover relatórios específicos do módulo operacional.

https://observatorio3setor.org.br/carrossel/plataforma-oferece-tecnologias-gratuitas-para-ongs/

A partir então da pesquisa concluída sobre sistemas web para gerência de projeto social, foi possível constatar uma grande variedade de sistemas web, que contemplam através de seus módulos, boa parte de particularidades e objetivos requisitados pela Associação Pró Brejaru. Evidente que, existe um conjunto de empresas particulares e profissionais que atuam neste nicho de seguimento de software, como pode ser observado alguma delas abaixo.

- SANKHYA, Soluções completas e integradas que auxiliam na gestão da sua equipe comercial e potencializam suas vendas**;**
- BHBit, software customizado para o terceiro setor, com funcionalidade para crescer com você;
- GestãoClick, é um sistema de gestão empresarial completo, para o controle ideal da sua ONG focar no negócio.

Acima descrevi apenas três de muitas conhecidas através da pesquisa, que atuam de forma extremamente profissional e com muito sucesso no mercado. Nesse sentido, considerando a questão do ponto de vista de aquisição destes sistema, o mercado disponibiliza a aquisição de várias formas, como por exemplo, a compra do sistema web, a compra do direito de uso do sistema web, etc. Em alguns casos essas empresas disponibilizam gratuitamente de forma parcial nestes sistemas web, alguns serviços através de seus módulos, como por exemplo: financeiro, beneficiários, doador, contas a receber, contas a cobrar, para atrair ongs e instituições com viés de cliente.

Apesar então de existir uma gama de sistemas web como solução para os problemas da associação Pró Brejaru, existem algumas particularidades que não são contempladas por nenhum deles. Como por exemplo, o cadastro de famílias que recebe algum tipo de doação, o próprio cadastro das doações, especialmente cesta básica, entre outros. Relatórios específicos solicitados pela prefeitura, além de não

ter como interface um site para promover a associação. Sendo que quando eles contemplam todas essas características, a utilização destes sistemas estão de uma forma ou de outra, vinculado a um contrato de uso, que dispende evidentemente um custo, contrapondo a primícia primeira, usado para busca de um sistema web, tendo em vista que a associação Pró Brejaru, em si é mantida pela prefeitura. A condição financeira então foi parte crucial para a tomada de decisão deste TCC, de construir um sistema web para solucionar características específicas da associação Pró Brejaru.

4 Desenvolvimento

A partir da decisão deste TCC de desenvolver um sistema web como solução para, os problemas característicos da associação Pró Brejaru, seguiu-se o desenvolvimento do trabalho no qual utilizou-se as técnicas de desenvolvimento de software. Como primeira parte foi realizado o levantamento de requisito, não como produto final absoluto, mas como parâmetro tanto para delimitar os requisitos dos processos mais críticos e necessários do sistema web, quanto para direcionar as próximas atividades do desenvolvimento. Após então o término de todo o processo de elaboração do projeto deste trabalho, e das técnicas de desenvolvimento de software subsequente, o levantamento de requisitos foi definido como segue abaixo.

4.1 Especificação de requisitos

4.1.1 Requisitos funcionais

Requisito Gerência funcional 1 – Login: O sistema deve permitir que seja feito o login considerando a hierarquia de utilização do sistema, disponibilizando as funcionalidades correta a cada usuário, caso o usuário tenha suas credenciais no sistema.

Requisito Gerência funcional 2 – Cadastrar Programas: O sistema deve permitir o cadastro de programas, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 3 – Cadastrar Projetos: O sistema deve permitir o cadastro de projeto, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 4 – Cadastrar Atividades: O sistema deve permitir o cadastro de atividade, caso o login do usuário contemple esta funcionalidade e tal atividade pertença ao projeto. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 5 – Cadastrar Instituições Parceiras: O sistema deve permitir o cadastro de Instituições parceiras, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 6 – Cadastrar Tipo Instituições: O sistema deve permitir o cadastro de tipo de Instituições, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência I funcional 7 – Cadastrar Funcionários: O sistema deve permitir o cadastro de funcionário, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 8 – Cadastrar Departamentos: O sistema deve permitir que seja feito o cadastro de departamento, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 9 – Cadastrar Salários: O sistema deve permitir que seja feito o cadastro de salário, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 10 – Cadastrar Cargo: O sistema deve permitir que seja feito o cadastro de cargo, caso haja no departamento esse cargo e caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 11 – Cadastrar Beneficiários: O sistema deve permitir o cadastro de beneficiários, caso o beneficiário não esteja cadastrado e o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 12 – Cadastrar Benefícios: O sistema deve permitir o cadastro de benefícios, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 13 – Cadastrar Tipo Benefícios: O sistema deve permitir o cadastro de tipo de benefícios, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 14 – Listar os Benefícios Concedidos: O sistema deve permitir a listagem dos benefícios concedidos, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 15 – Cadastrar Doações: O sistema deve permitir o cadastro de doações, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 16 – Cadastrar Doadores: O sistema deve permitir o cadastro de doador, caso o doador não esteja cadastrado e o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Gerência funcional 17 – Cadastrar Tipo de Doações: O sistema deve permitir que seja cadastrado os tipos de doações, caso o login do usuário contemple esta funcionalidade.

Requisito Gerência funcional 18 – Cadastrar Papéis: O sistema deve permitir que seja feito o cadastro dos papéis, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não

Requisito Gerência funcional 19 – Listar Doações: O sistema deve permitir que seja listada as doações, caso o login do usuário contemple esta funcionalidade.

Requisito Gerência funcional 20 – Cadastrar Endereço: O sistema deve permitir que seja cadastrado endereços, caso o login do usuário contemple esta funcionalidade.

Requisito Financeiro funcional 21 – Cadastrar Balancetes: O sistema deve permitir o cadastro de balancete, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 22 – Atualizar Balancete: O sistema deve permitir a atualização dos balancetes, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 23 – Cadastrar Conta: O sistema deve permitir o cadastro da conta, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 24 – Cadastrar Agência: O sistema deve permitir o cadastro da agência, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 25 – Registrar Saída: O sistema deve permitir que o usuário registrar saída de valores no balancete, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 26 – Cadastrar Cheque: O sistema deve permitir que o usuário cadastre cheque, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 27 – Cadastrar Descrição: O sistema deve permitir que o usuário cadastre descrição, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 28 – Cadastrar Prestadoras de Serviços: O sistema deve permitir o cadastro de empresas que prestam algum tipo de serviço a associação, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 29 – Cadastrar Prestador de serviços Físicos: O sistema deve permitir o cadastro de pessoas que prestam algum tipo de serviço a associação, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 30 – Cadastrar Fornecedores: O sistema deve permitir o cadastro de empresas que fornecem algum tipo de produto a associação, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 31 – Registrar Entrada: O sistema deve permitir que o usuário registre entrada de valores no balancete, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 32 – Cadastrar Associados Contribuintes: O sistema deve permitir o cadastro de associados contribuintes, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrada ou não.

Requisito Financeiro funcional 33 – Cadastrar Instituição: O sistema deve permitir que seja feito o cadastro de Instituição, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 34 – Cadastrar Tipo Instituição: O sistema deve permitir que seja feito o cadastro de tipo de instituição, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Financeiro funcional 35 – Gerar Balanço: O sistema deve permitir gerar balanço anual, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 36 – Cadastrar Alunos: O sistema deve permitir o cadastro de aluno, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 37 – Cadastrar Endereço: O sistema deve permitir que seja feito o cadastro de endereço, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 38 – Cadastrar Aluno Espera: Caso não haja vaga para as atividades no momento, o sistema deve permitir que seja feito o cadastro de aluno em espera, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 39 – Cadastrar Pai: O sistema deve permitir que seja feito o cadastro de pai, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 40 – Cadastrar Mãe: O sistema deve permitir que seja feito o cadastro de mãe, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não

Requisito Operacional funcional 41 – Cadastrar Escola: O sistema deve permitir que seja feito o cadastro de escola, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 42 – Cadastrar Cartório: O sistema deve permitir que seja feito o cadastro de cartório, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 43 – Cadastrar Problema de Saúde: O sistema deve permitir que seja feito o cadastro de problema de saúde, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 44 – Cadastrar Alergia: O sistema deve permitir que seja feito o cadastro de alergia, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 45 – Cadastrar Irmãos: O sistema deve permitir que seja feito o cadastro de irmãos, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 46 – Cadastrar Turma: O sistema deve permitir que seja feito o cadastro de turmas, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 47 – Registrar Frequência de Alunos: O sistema deve permitir que seja registrado a frequência dos alunos, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 48 – Atualizar Frequência: O sistema deve permitir que seja feito a atualização das frequências já cadastradas, se o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

Requisito Operacional funcional 49 – Relatórios Geral de Alunos: O sistema deve permitir gerar relatório retornando todos os alunos cadastrado, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 50 – Relatório Geral de Documentos: O sistema deve permitir gerar relatório retornando todos os documentos identificadores dos alunos e de seus responsáveis, caso esteja cadastrado e o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 51 – Relatório Mensal de Alunos Por Atividades: O sistema deve permitir gerar relatório retornando por mês, à quantidade de alunos por atividade, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 52 – Relatório Lista Familiar: O sistema deve permitir gerar relatório retornando o aluno e sua família, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 53 – Gerar Relatório de Desistentes por ano: O sistema deve permitir gerar relatório retornando por ano, todos os alunos desistentes das atividades, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 54 – Relatório Lista de Espera: O sistema deve permitir gerar relatório retornando os alunos cadastrados que estão na lista de espera, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 55 – Relatório de Funcionário por Departamento: O sistema deve permitir gerar relatório retornando todos os funcionários de cada departamento, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 56 – Relatório Mensal de Frequência dos alunos Por Turma: O sistema deve permitir gerar relatório mensal da frequência dos alunos por turma, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 57 – Relatório Mensal de Atendimento Por Turno e Sexo: O sistema deve permitir gerar relatório mensal retornando à quantidade de alunos ativos por turno e sexo, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi retornado ou não.

Requisito Operacional funcional 58 – Gerar PDF de Todos os Relatórios: O sistema deve permitir que seja gerado PDF de todos os relatórios, caso o login do usuário contemple esta funcionalidade. O sistema informará se foi cadastrado ou não.

4.1.2 Requisitos não funcionais

Requisito não funcional 1 – Especificação de projeto: além do código do sistema, deve ser produzida especificação de projeto baseada em UML, para os seguintes diagramas: de classe, para abstração das classes necessárias, de use case, para definir os atores e seus requisitos, de iteração, para definir o fluxo do programa.

Requisito não funcional 2 – Interface gráfica para usuário: o programa deverá ter interface gráfica, para interação com usuário.

Requisito não funcional 3 - Tecnologia de interface gráfica para usuário: a interface gráfica deve ser baseada no padrão de projeto J2EE.

Requisito não funcional 4 – Especificação do Design: a cor predominante das interfaces do usuário será verde oliva.

Requisito não funcional 5 – Especificação da Arquitetura: O sistema deve conter cinco interfaces de interação, sem necessidade de efetuar login para acessar-las. Utilizadas para promover a associação através do site, sendo elas: home, quem somos, o que fazemos, com quem fazemos, como fazemos.

Esses são os principais requisitos definidos e que serão utilizados para obtenção do êxito na conclusão deste trabalho. Observando sempre que eles não são absolutos, mas servem como base para delimitar os requisitos que serão implementados. Sendo que desta forma, pelas condições de tempo e complexidade, tanto será possível ser acrescentados um ou outros de maior relevância, quanto retirados alguns destes a priori definidos.

4.2 Arquitetura

A princípio poderia ser considerada para satisfazer a necessidade de solução para os problemas característicos de software da associação Pró Brejaru, um software para desktop. Porém, alguns dos requisitos pré-definidos nos remetem às características de um site. O que sendo analisado de forma mais específicas, percebeu-se que seria melhor propor um sistema WEB considerando a arquitetura WEB 2.0, cliente servidor. Alguns dos requisitos preveem páginas que de certa forma representam conteúdo “estático”, as quais são interfaces de comunicação da associação com a sociedade. E poderão ser acessadas por qualquer usuário da internet, pois essas páginas servirão em última análise de promoção do trabalho da associação. Tendo em vista porém que a quantidade de acesso às páginas, não é

uma questão crítica, pois o número de acessos em determinado momento não tende a sobrecarregar o sistema, optou-se por utilizar o padrão de projeto java J2EE.

Os principais serviços disponibilizados pela plataforma J2EE destinam-se a suprir as necessidade de aplicações empresariais distribuídas, isto é, aquelas que necessitam da flexibilidade de disponibilizar acesso à sua lógica de negócios e dados para diferentes tipos de dispositivos cliente (navegadores, dispositivos móveis, aplicações desktop, etc.), ou para outras aplicações residentes na mesma empresa ou fora. A figura abaixo ilustra um ambiente J2EE típico.

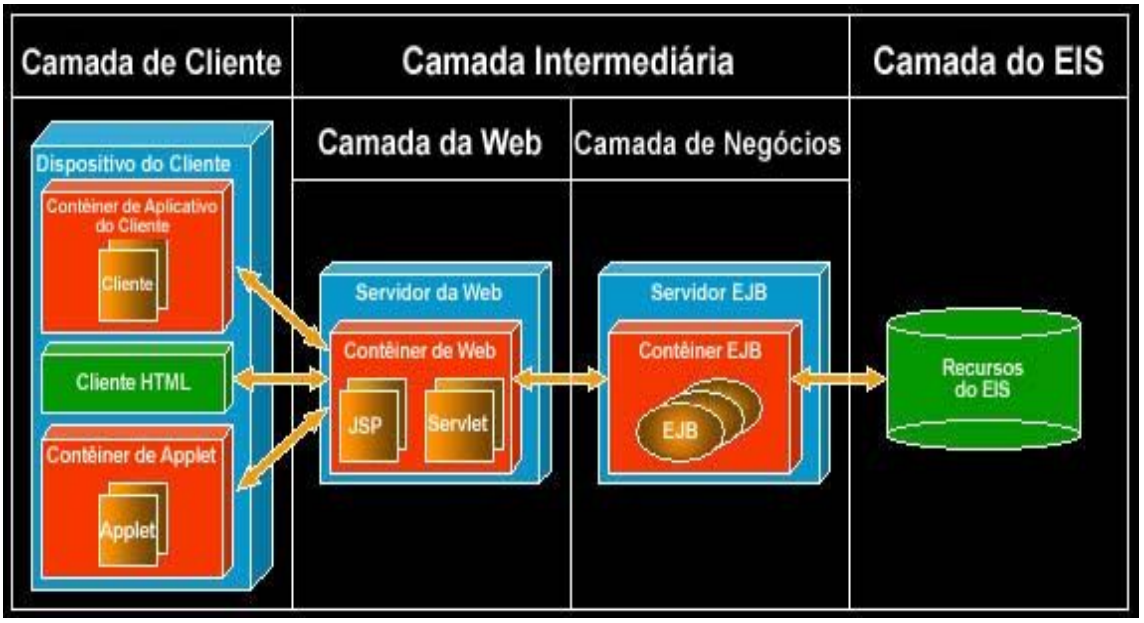


Figura 2 : Representa as camadas do ambiente J2EE.

Aplicações distribuídas são comumente compostas de uma camada cliente, que implementa uma interface com usuário, e uma ou mais camadas intermediárias, que processam a lógica do negócio e provêem serviços a camada cliente, e outra denominada de, Enterprise Information System ou EIS, formado por sistemas legados e banco de dados. A infraestrutura formada pela J2EE possibilita que estas camadas, possivelmente localizadas em máquinas diferentes, possam se comunicar remotamente e juntas compor uma aplicação (Padrões de Projeto, DEVMEDIA).

https://www.devmedia.com.br/principais-padroes-j2ee-para-a-construcao-de-aplicacoes-nao-distribuidas-parte-i/1812

Assim então dentro desta perspectiva de padrão de projeto J2EE, as tecnologias de ferramentas selecionadas para concluir o trabalho, foram as mais tradicionais convencionadas por essa plataforma. De forma que elas são conhecidas de todos os profissionais que atuam na área da TI, e seria desnecessário apresentar-las

completamente aqui. No entanto, para a banca melhor conhecer o projeto, é interessante explicitar-las: para ide foi utilizado o NetBeans 8.2, para as interfaces com clientes foi utilizado o framework o JSF 2.3 primefaces 6.1, para servidor o GlassFish 4.1.1, para os CRUDS JPA utilizando o framework Eclipselink 2.1, para banco de dados o Java DB derby 10.10.2.0, para a modelagem do sistema foi utilizado o Visual Paradigm 16.2.

4.3 Modelagem

Após a definição da arquitetura para melhor compreensão das diretrizes da solução proposta e definição do sistema, fez-se necessário a elaboração de três diagramas. O diagrama de classes, que contém três diagramas que definem por consequência as classes do sistema. Um diagrama de classes referente ao módulo gerente, outro referente ao módulo financeiro e por último um referente ao módulo operacional. O diagrama de Caso de Uso, com o qual foi possível definir os atores do sistema e os casos de uso referente a cada ator dentro do sistema. Um para o módulo gerente, outro para o módulo financeiro e por último um para o módulo operacional. E por último o diagrama de iteração, com o qual foi possível determinar a sequência do fluxo de execução dos caso de uso, para cada ator no sistema. Semelhantemente um para o módulo gerente, outro para o módulo financeiro e por último um para o módulo operacional. Outros diagramas poderiam fazer parte da modelagem, porém, percebeu-se após a modelagem destes três que, estes continham informações suficientemente relevantes para satisfazer o objetivo principal da proposta de modelagem. Outros diagramas, só não acrescentariam relevância objetiva a modelagem, quanto acarretaria a necessidade de mais tempo para a concluir a modelagem. Desta forma a equipe necessariamente optou em utilizar destes três tipos de diagrama para modelar completamente o sistema. Todos os diagramas previamente especificados e modelados serviram como diretriz para implementação e desenvolvimento do sistema web e podem ser utilizadas para compor a documentação do sistema.

Figura 3 : Diagrama de classes módulo operacional.

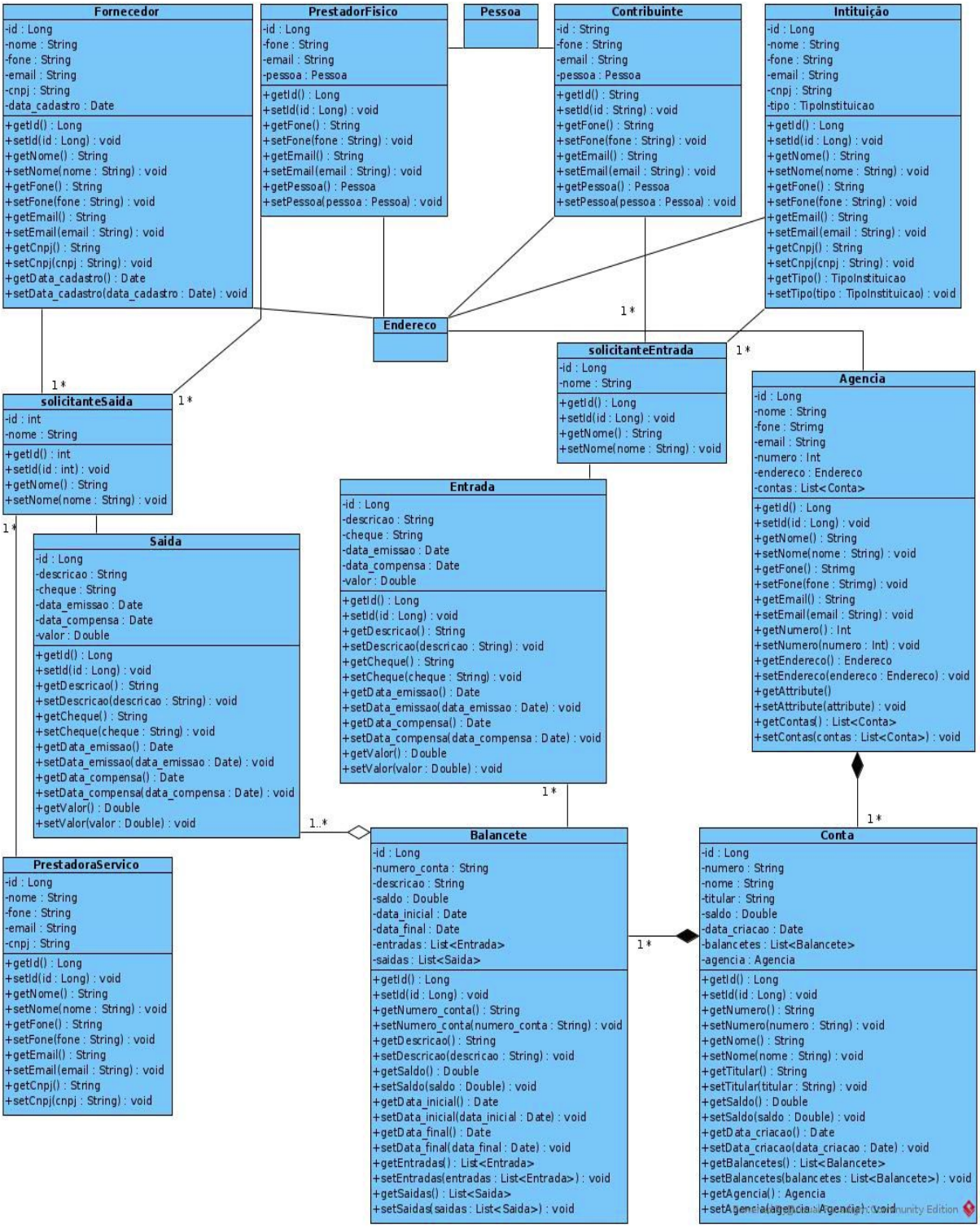


Figura 4 : Diagrama de classes módulo financeiro.

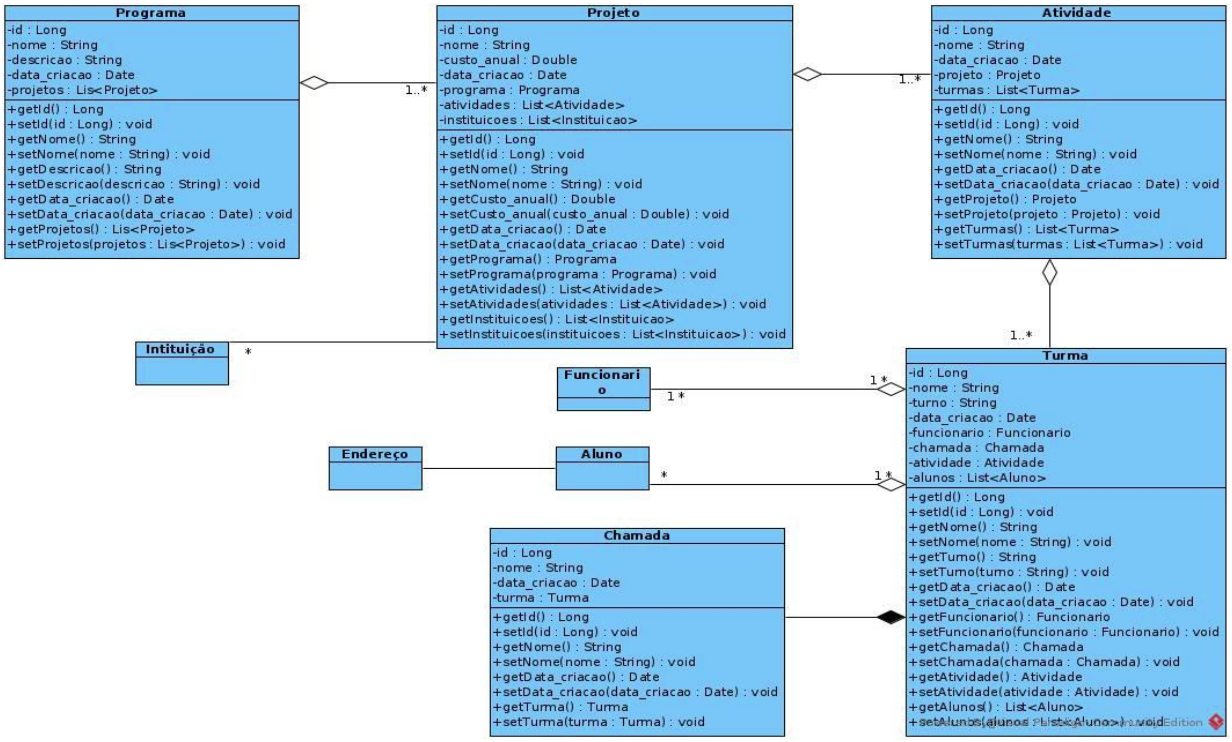


Figura 5 : Diagrama de classes módulo gerente.

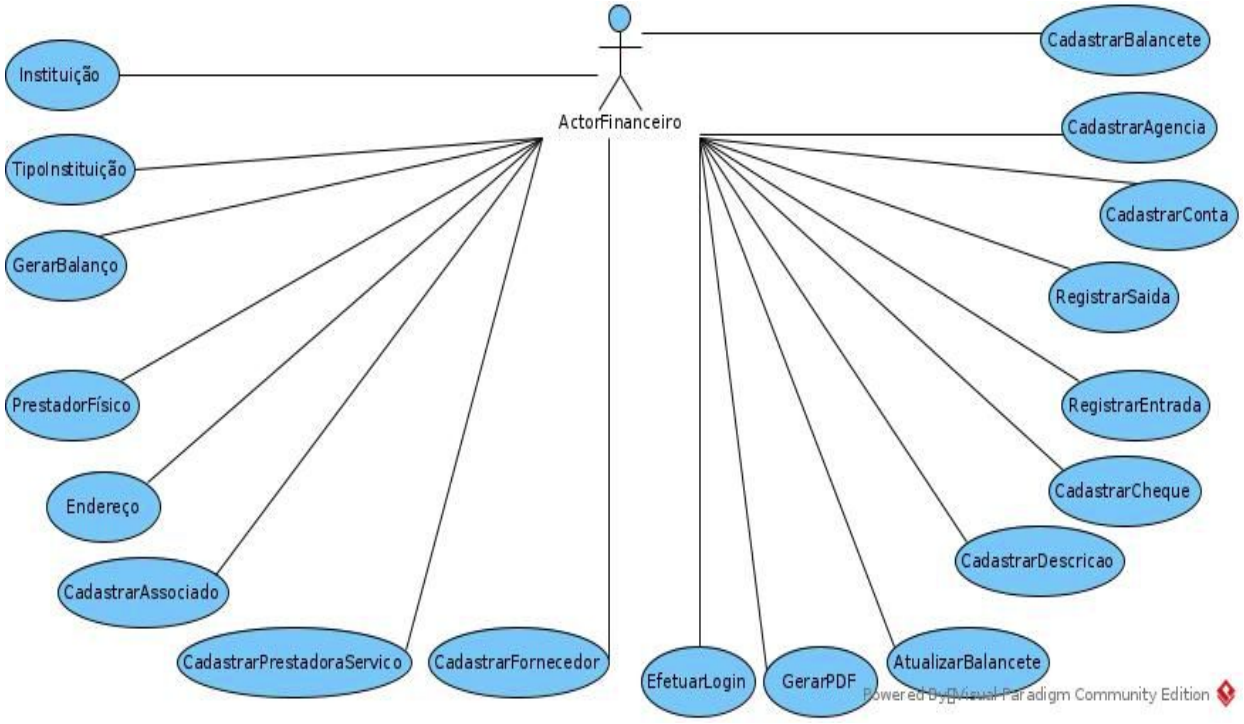


Figura 6 : Diagrama de caso de uso ator financeiro.

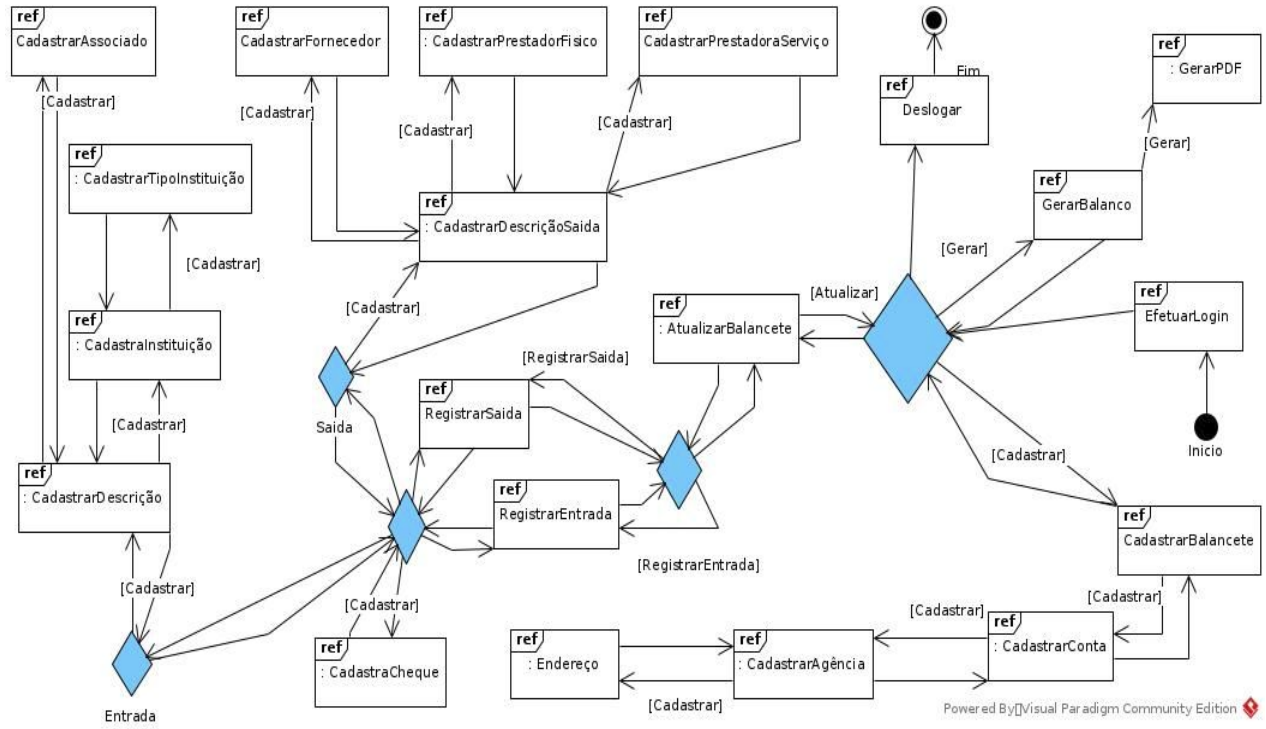


Figura 7 : Diagrama de iteração ator financeiro.

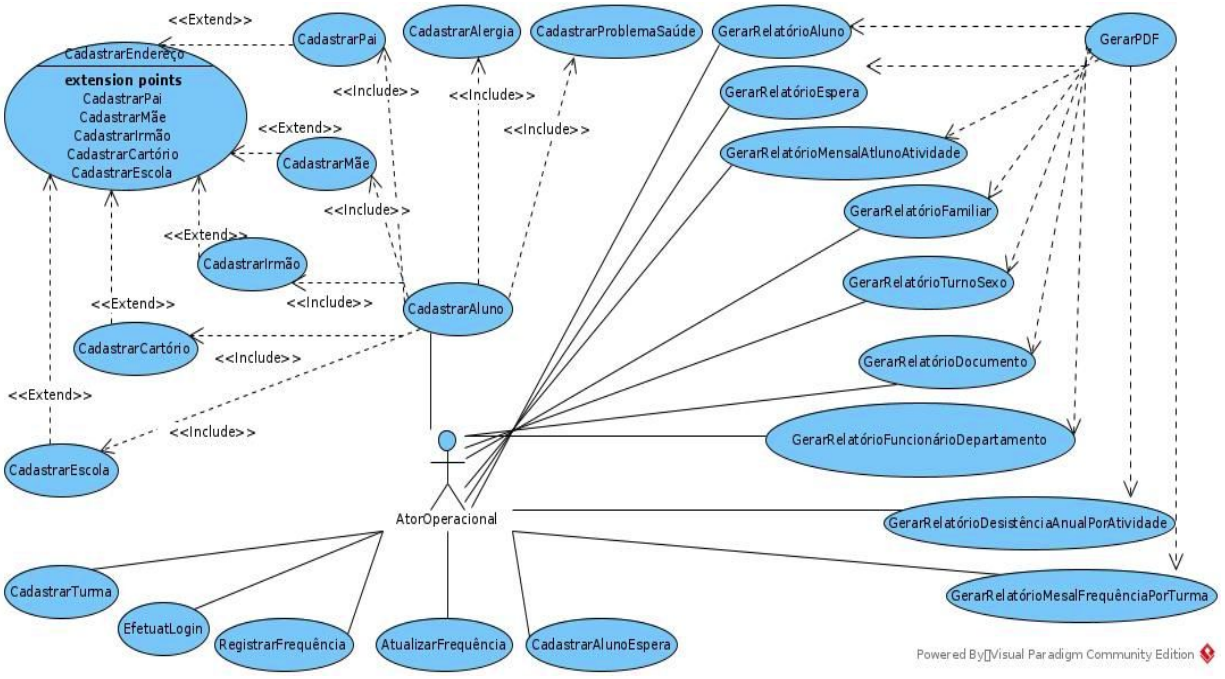


Figura 8 : Diagrama de caso de uso ator operacional.

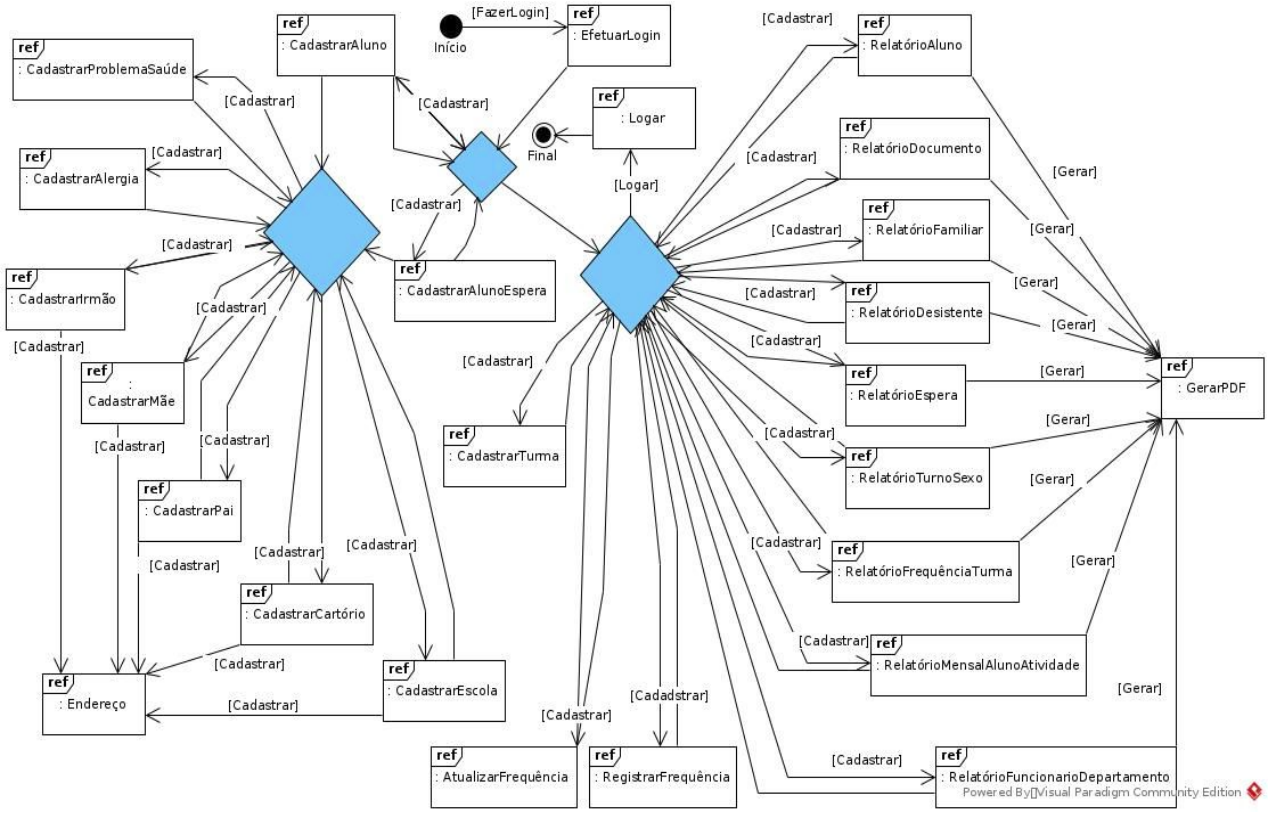


Figura 9 : Diagrama de iteração operacional.

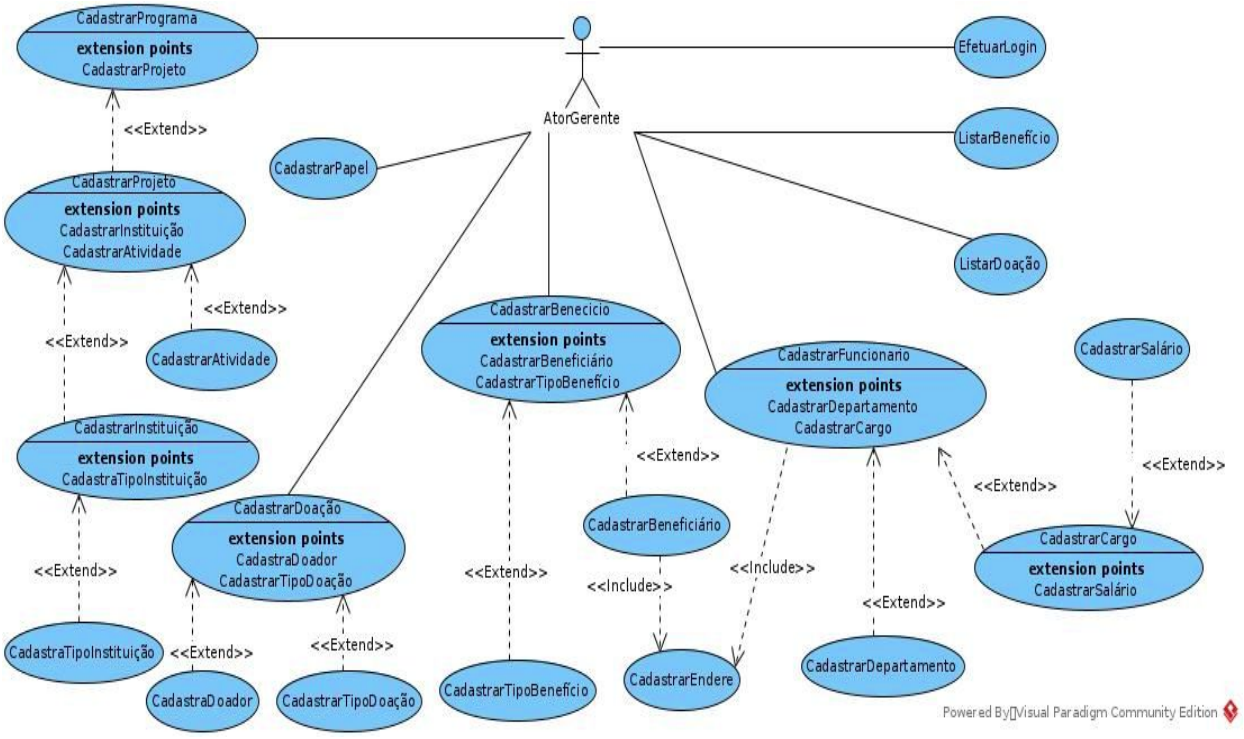


Figura 10 : Diagrama de caso uso ator gerente.

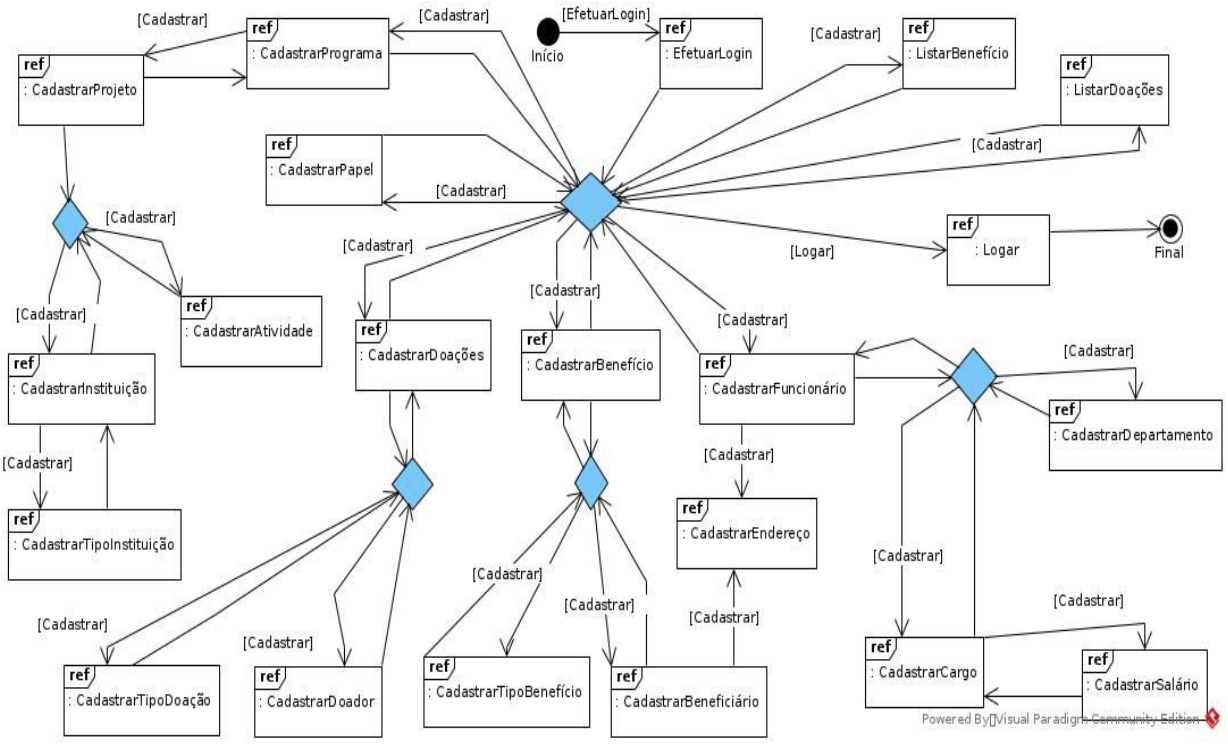


Figura 11 : Diagrama de iteração ator gerente.

A partir dos requisitos então chegou-se a esses conjunto de diagramas, os quais servem como diretriz para a implementação do sistema. O diagrama de classes define o conjunto de classes e atributos que serão implementados, obviamente não em sua totalidade, mas em sua forma de classes e atributos mais convencionais. O diagrama de caso de uso define os atores, e como cada um deles poderá interagir com o sistema. O diagrama de iteração, prevê uma sequência de possibilidades de interação que cada ator poderá executar a partir da efetuação do login com sucesso.

4.4 Implementação

A implementação foi concluída considerando a engenharia de requisitos a modelagem do sistema, e a arquitetura a priori definida. Utilizando as ferramentas e tecnologias de desenvolvimento se software do padrão JEE cliente servidor, para codificar o sistema. Cabendo ressaltar ou propriamente definir no entanto que o presente sistema deve ser classificado como uma versão 1.0. Dentro desta perspectiva como uma primeira versão o sistema objetivamente foi implementado da seguinte forma, um site contendo seis páginas: home, quem somos, o que fazemos,

com quem fazemos, como fazemos e contatos. Além de um conjunto de CRUDs que contemplou, toda a sistematização dos processos da associação Pró Brejaru, segundo as características especificada na modelagem construída através da especificação. Após teste dos requisitos codificados em modo de desenvolvimento, é passível de afirmar que todos os requisitos foram concluídos aparentemente de forma eficiente. Pois todos os requisitos codificados foram submetidos a testes, de forma específica, como também compondo módulo e por último o sistema de forma completa. Tendo como delimitação do escopo deste presente trabalho, todos os requisitos pré-definidos neste documento.

A página inicial home ficou definida assim, no topo seis fotos sombreadas alternando-se a cada seis segundos, logo abaixo na coluna central, um vídeo que pode ser compartilhado em redes sociais, nas colunas laterais fotos de atividades da associação.

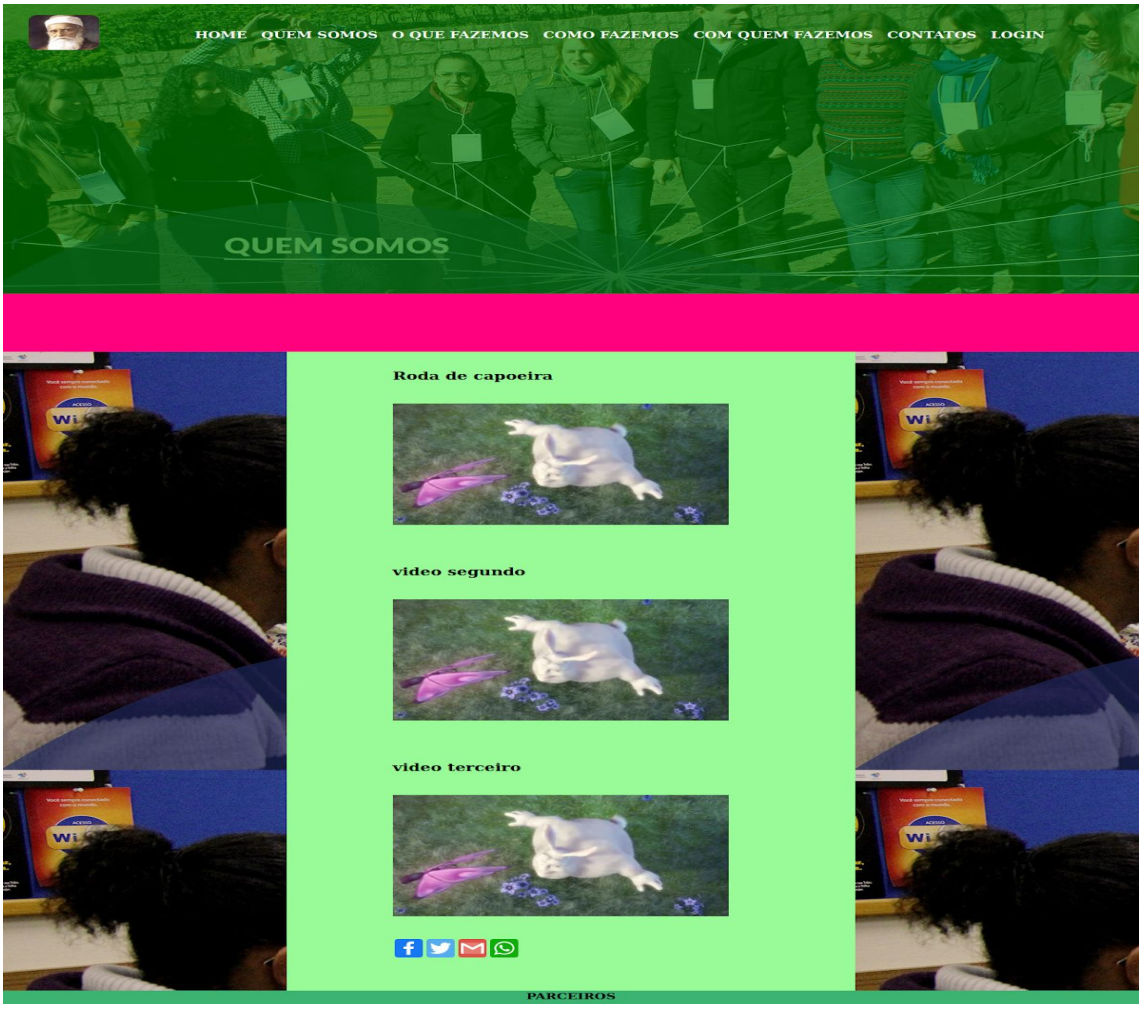


Figura 12 : Página inicial do site.

A página quem somos mostra, a visão, missão, crenças e valores, a fundadora da associação Pró Brejaru, bem como o atual presidente, além da estrutura hierárquica e dos membros colaboradores da associação.

Figura 13: Página quem somos do site.

A página o que fazemos apresenta o conjunto de projetos e atividades disponibilizadas pela associação Pró Brejaru. Basta clicar no lado direito sombreado das transparências do projeto, para ser encaminhado para uma página específica sobre o projeto.

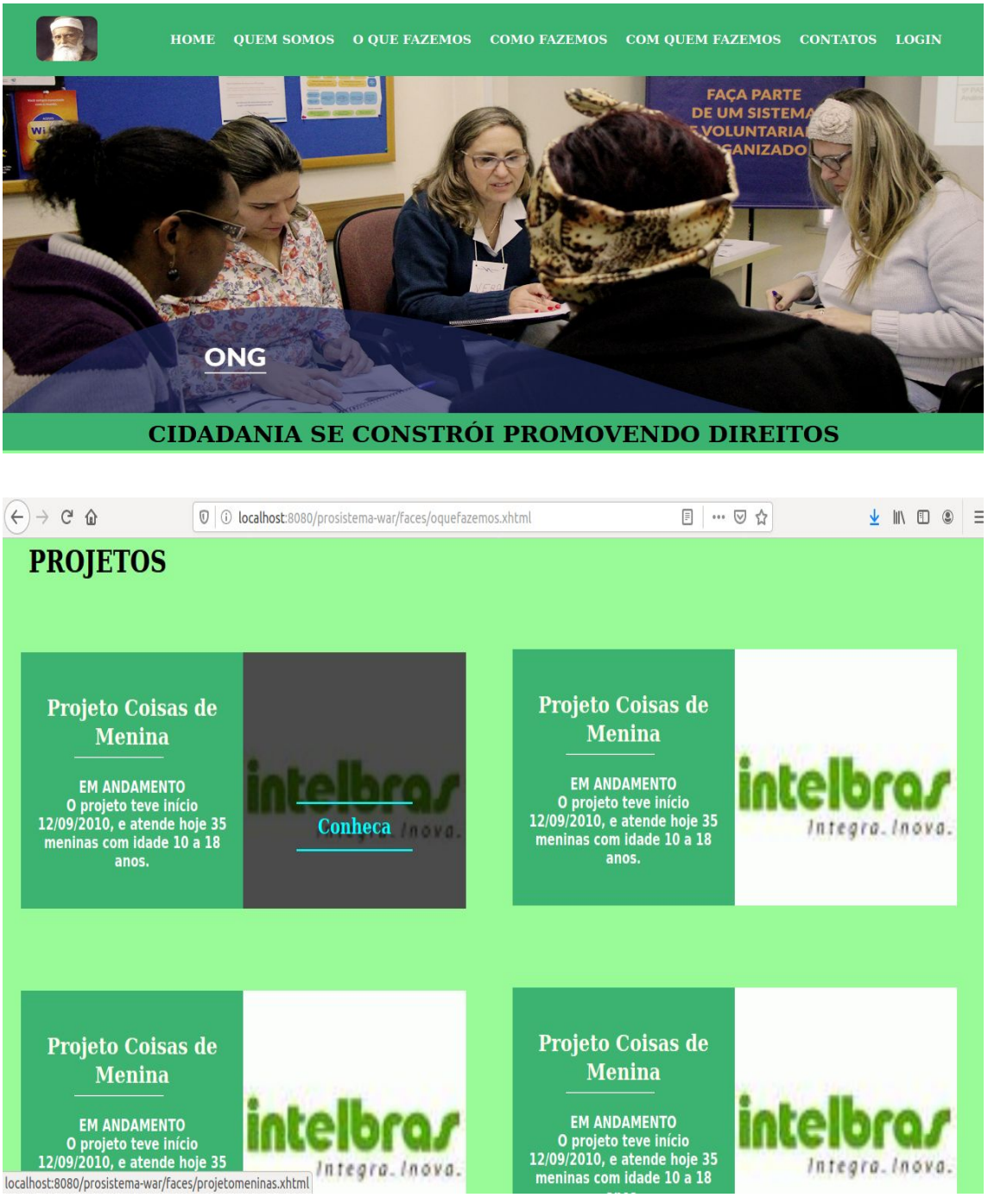


Figura 14: Página o que fazemos do site.

A página como fazemos informam quais as atividades são disponibilizadas e os meios pelos quais são executadas essas atividades, além de imagens de algumas das atividades sendo executadas pelas crianças que são atendidas diariamente pela associação Pŕo Brejaru.

Figura 15: Página como fazemos do site.

A página com quem fazemos traz uma mensagem e apresenta as empresas que patrocinam, programas, projetos e atividades que a associação Pró Brejaru disponibiliza para seus alunos; específica todas as empresas que fazem parte dos mantenedores, parceiros e patrocinadores.

Figura 16: Página com quem fazemos do site.

Por fim, temos a pagina contatos que informa os meios pelos quais se pode entrar em contato com a associação Pró Brejaru.

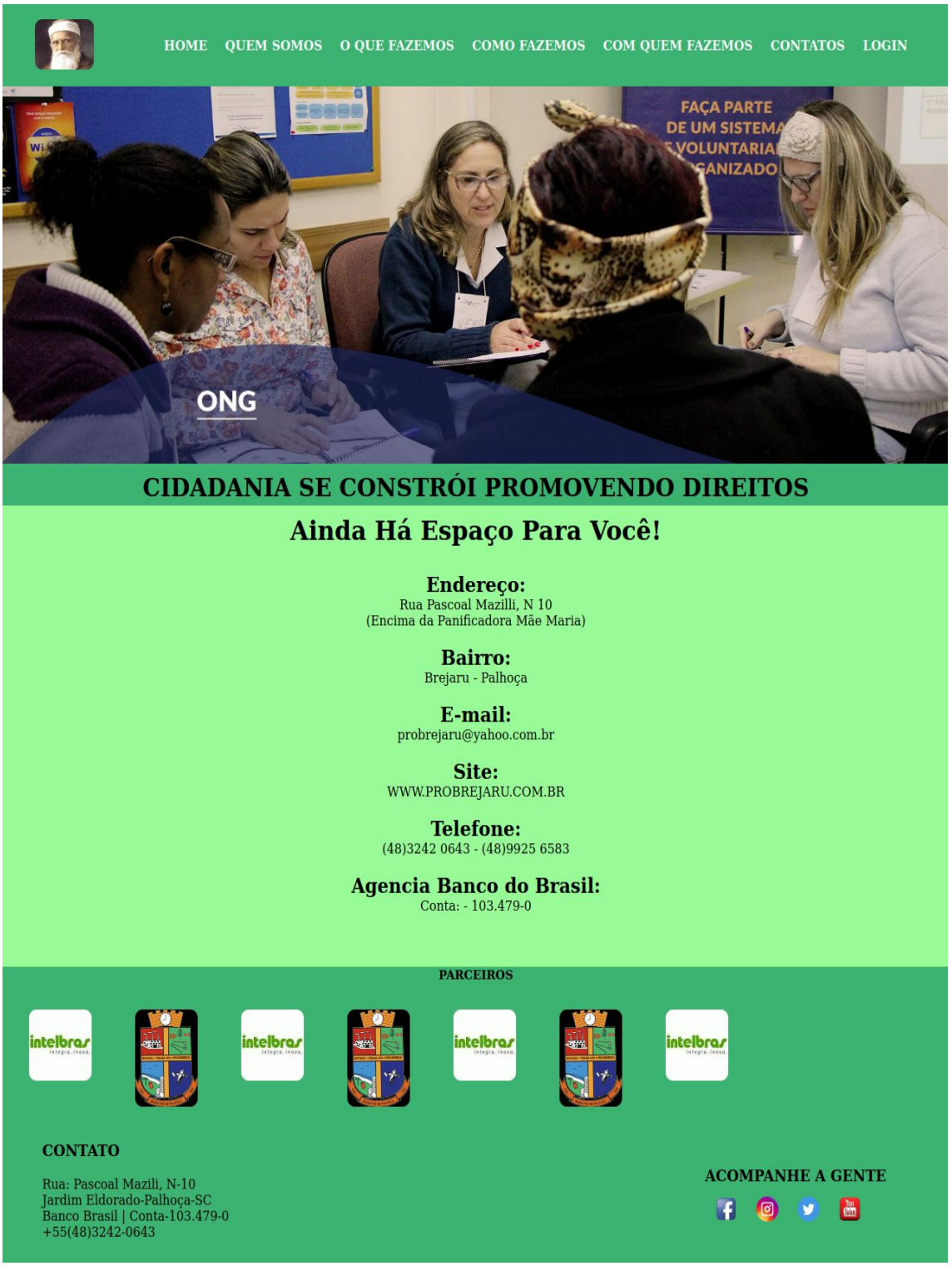


Figura 17: Página contatos do site.

A partir deste momento iremos descrever o sistema concluído em sua versão 1.0: a página login permite aos funcionários da associação Pró Brejaru se logar ao sistema; especificamente definido os papéis de, gerente, financeiro, operacional; após efetuar o login com sucesso, o usuário vai obter acesso a uma página correspondente a sua função na associação, considerando os módulos, gerência, financeiro e operacional.

Figura 18: Página login do site.

Desta forma, após a um login bem sucedido o usuário do módulo Gerência tem acesso a uma página contendo todos os casos de uso correspondente a sua função na associação.

Figura 19: Página função gerente do site.

Entrando no módulo gerente, vamos visualizar algumas das interfaces e requisitos mais relevantes, para que o sistema seja melhor conhecido, sempre da direita para á esquerda; começamos com o cadastro de doação; a doação tem quatro atributos respectivamente, o tipo de doação, pode ser dinheiro, roupa, alimento, etc, o valor correspondente em espécie, o doador e a data da doação; e como percebido dois links, um para cadastro de doador, e outro para cadastro de tipo de doação. Vale ressaltar que tem apenas o botão cadastrar, porque a exclusão de uma doação é feita na tabela de listagem das doações.

Figura 20: Tela de cadastro de doação.

Caso o tipo de doação não esteja contido no checkBox tipo de doação, é possível clicar no link cadastrar tipo doação, para cadastrar ou remover um tipo de doação, como na tela abaixo.

Figura 21: Tela de cadastro de tipo de doação.

Caso o doador não esteja cadastrado no checkBox doador, é possível clicar no link cadastro doador, para cadastrar, atualizar ou remover doador, como pode ser observado na tela abaixo. O conjunto de atributos que descreve um doador, tem por objetivo preservar a identidade do doador. Vale ressaltar que nesta versão 1.0, o sistema aceita apenas doação feita por doador físico. Empresas e instituições que queira doar em espécie, entra no balanço como, mantenedor, parceiro ou patrocinador.

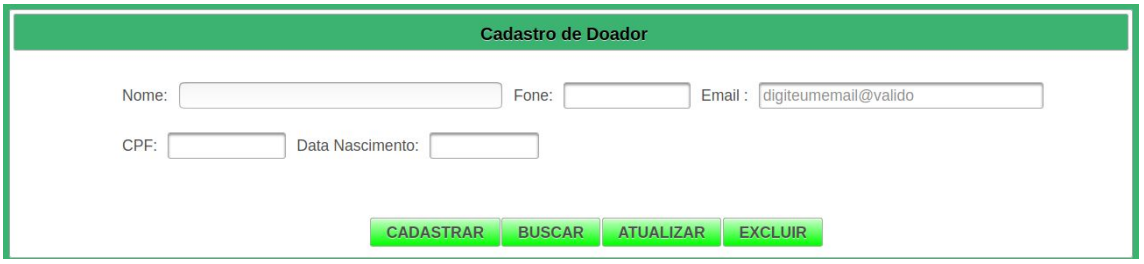


Figura 22: Tela de cadastro de doador.

Após o cadastro de doações é possível listar as doações, para saber quem fez as doaçẽs e o tipo das doações, além da data da doação. Para isso basta acessar na página gerente o link listar doações, selecionar o nome específico do doador, ou todas as doações, selecionar a data e clicar no botão buscar para que seja listados as doações na data especificada, como mostrado na página abaixo, para excluir basta selecionar a linha da tabela e clicar no botão excluir, e um link em azul para voltar para a página gerente. Vale ressaltar que este requisito é crítico e específico nos processos da associação, tendo em vista que, tanto os doadores, quanto os beneficiários das doações são mantidos em documentos de papel em armário de arquivos, dificultando saber quais os beneficiários já obtiveram algum tipo de benefício durante o mês.

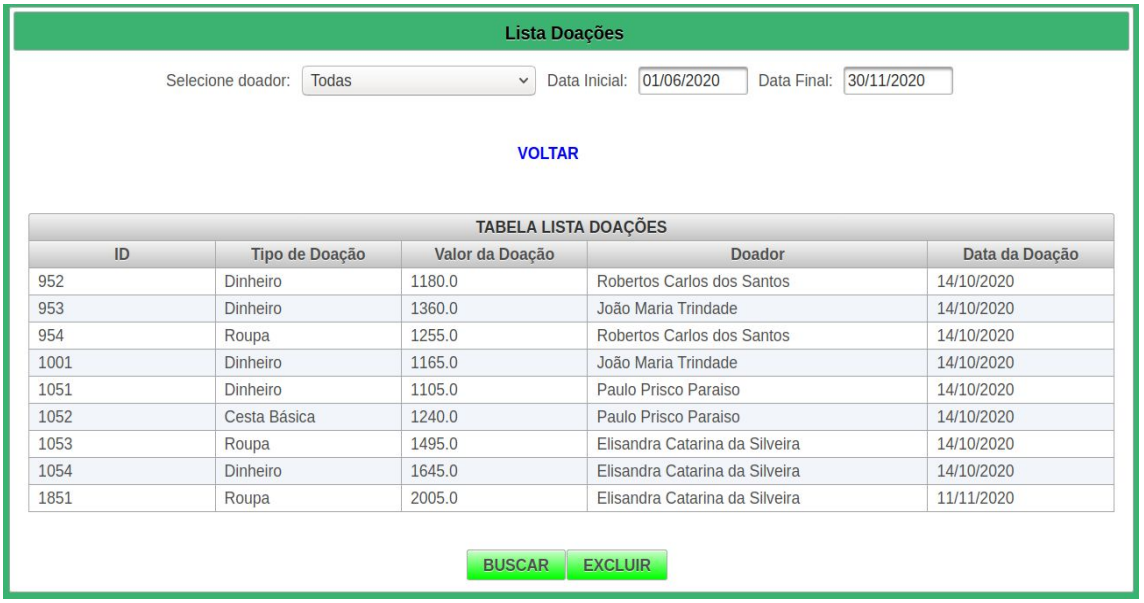


Figura 23: Tela lista de doações.

Outro processo crítico e específico da associação Pró Brejaru e que complementa o cadastro de doações é o cadastro de benefícios. Enquanto no cadastro de doações pode ser obtido informações referentes aos doadores, no cadastro de benefícios é obtido informações referentes aos beneficiários. O cadastro de benefícios contém quatro atributos, o tipo de benefício, o valor de benefício, o beneficiário que recebeu o benefício e a data que o benefício foi concedido. Além de dois links, para cadastro de tipo de benefício e cadastro de beneficiário. Como mostrado na tela abaixo.

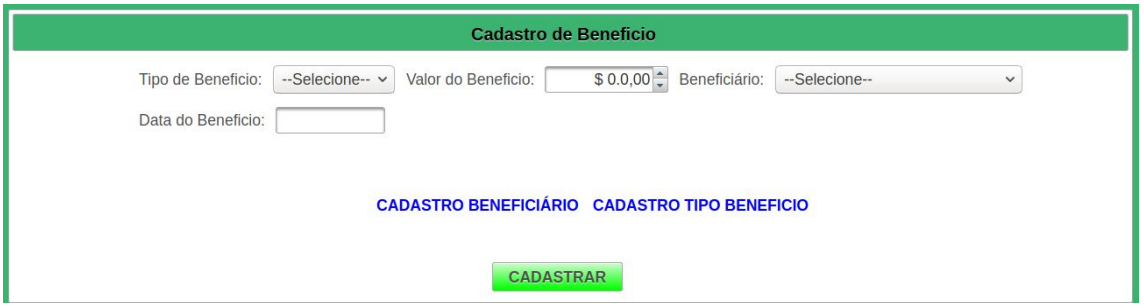


Figura 24: Tela cadastro de benefício.

Também é possível cadastrar o tipo de benefício caso não esteja cadastrado, ou remover algum benefício que não seja mais utilizado, basta clicar no link cadastro de tipo benefício, para acessar a tela abaixo.

Figura 25: Tela cadastro de tipo benefício.

Caso o beneficiário não esteja cadastrado basta clicar no link cadastro de beneficiário para acessar a tela abaixo e cadastrar atualizar ou remover um beneficiário; na tela cadastro de beneficiário tem um link para cadastro de endereço, caso o endereço do beneficiário não esteja previamente cadastrado. Vale ressaltar que é possível uma empresa fazer doações que não seja em espécies, desde que tenha uma pessoa física disponível para cadastro.

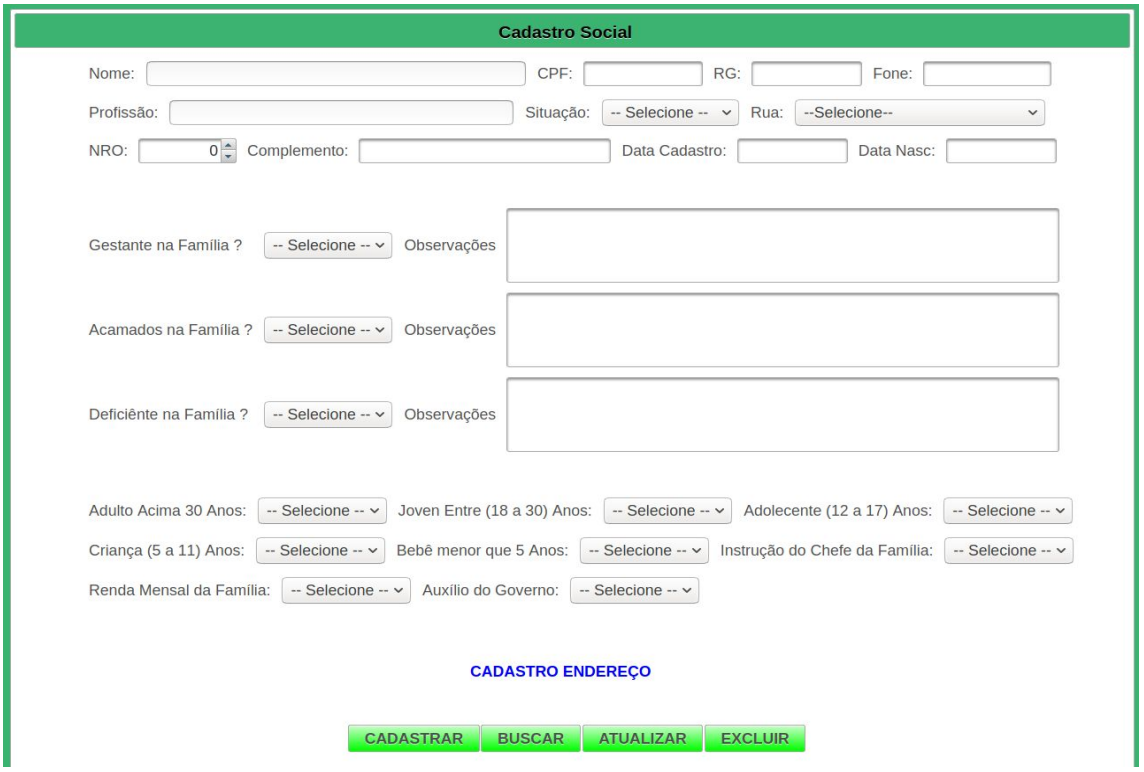


Figura 26: Tela cadastro de beneficiário.

Após a cadastro completo dos benefícios é possível obter informações de quem já recebeu algum benefício durante o mês, quais tipos de benefícios o beneficiário em questão recebeu, basta acessar a tela gerente e clicar no link listar benefícios, para acessar a tela de listagem, onde pode ser feito a busca pelo nome do beneficiário ou por todos, os benefícios são filtrados por data.

Figura 27: Tela listagem de benefício.

O próximo caso de uso seguindo a regra da esquerda para direita é o cadastro de funcionário, que tem por objetivo entre outros possíveis, atender o relatório de funcionário por departamento. Tão por isso, um dos atributos do funcionário é o departamento onde ele trabalha, bem como o cargo que ele ocupa. Dois links são disponibilizados para o cadastro, caso o departamento ou o cargo não esteja previamente cadastrados. Na tela gerente clicando no link do funcionário, o usuário é levado para a tela de cadastro abaixo.

Figura 28: Tela cadastro de funcionário.

Clicando no link cadastro de departamento o usuário vai acessar a tela de cadastro de departamento abaixo, bem mais simples, mas com opção para cadastrar e remover departamento.

Figura 29: Tela cadastro de departamento.

Clicando no link cadastro de cargo o usuário terá acesso ao cadastro de cargo, bem mais simples, mas com opção para cadastrar, atualizar e remover. Sendo disponibilizado um link para cadastro do salário, caso ainda não exista um salário compatível com o cargo, como mostrado na tela abaixo.

Figura 30: Tela cadastro de cargo.

Caso não tenha um salário compatível com o cargo, basta clicar no link cadastro de salário, para o usuário ter acesso a página de cadastro e remoção de salário, como mostrado na tela abaixo.

Figura 31: Tela cadastro de salário.

Próximo caso de uso é o cadastro de programa, que tem por objetivo conter informações relevantes dos programas e projetos da associação, com o objetivo de promover a associação nas páginas do site. Um dos atributos do cadastro de programa é um ou mais projetos, pois cada projeto contém dados relevantes de sua execução, como por exemplo, custo, instituições parceiras, etc. Basta na tela gerente clicar em cadastrar programa para o usuário ter acesso a tela de cadastro de programa abaixo.

Figura 32: Tela cadastro de programa.

Caso o projeto não esteja previamente cadastrado, basta clicar no link cadastro de projeto na tela anterior, para poder cadastrar, atualizar ou remover projeto. Dois atributos são interessantes no projeto: as instituições parceiras e as atividades que são executadas pelo projeto, ambas podem ser uma ou uma lista. Instituições parceiras, são aquelas que contribuem com um determinado valor anualmente, para a execução de um projeto; as mesmas que são registradas em uma entrada nas atualizações do balancete, como poderá ser averiguado logo adiante, no módulo financeiro. Dois links são disponibilizados, um para cadastro de atividade e outro para cadastro de instituições parceira, caso seja necessário um novo cadastro.

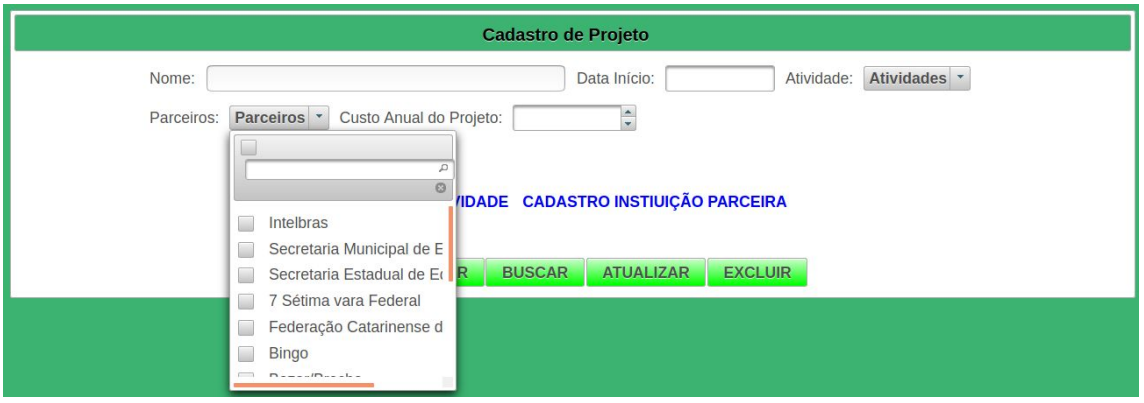


Figura 33: Tela cadastro de projeto.

Após clicar no link cadastro de atividade na tela anterior o usuário acessa a tela de cadastro de atividade. Um dos atributos da atividade é uma ou uma lista de turmas que executam essa atividade. Uma atividade denominada Capoeira pode conter Turma Capoeira A, Turma Capoeira B, etc, dependendo da quantidade de dias da semana que à atividade é ministrada e do turno: tarde, manhã. No entanto o cadastro das turmas fica a cargo do módulo operacional.

Figura 34: Tela cadastro de atividade.

O outro link permite ao usuário o cadastro de instituição parceira, caso seja necessário o cadastro de uma nova instituição. Um dos atributos de uma instituição parceira é o tipo, por isso é disponibilizado um link, para o cadastro do tipo de instituição. A relevância do atributo tipo de instituição faz sentido ser melhor abordado no módulo financeiro.

Figura 35: Tela cadastro de instituição parceira.

Basta clicar no link tipo instituição para o usuário acessar a tela de cadastro de tipo de instituição, como mostrado abaixo.

Figura 36: Tela cadastro de tipo instituição.

O último requisito do módulo gerente é o cadastro de papel, que tem por objetivo, delegar as responsabilidades dos usuários dos módulos operacional e financeiro, através do cadastro do papel dos usuários.

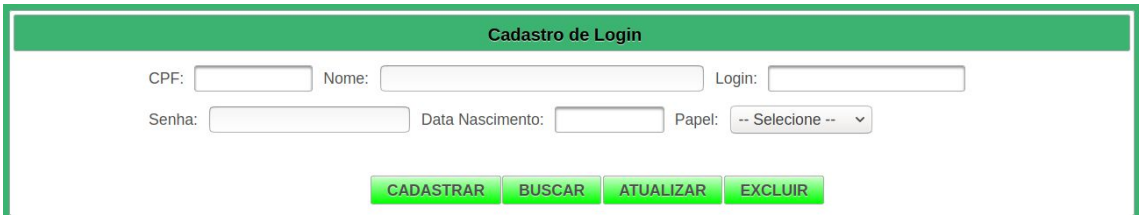


Figura 37: Tela cadastro de papel.

A partir de agora, vamos abordar o módulo financeiro e para melhor conhecimento de sua estrutura, vamos analisar os requisitos, da esquerda para a direita na tela do financeiro, mas de acordo com a necessidade. A partir de um login efetuado com sucesso o usuário tem acesso ao módulo financeiro como na página abaixo.

Figura 38: Tela módulo financeiro.

Desta forma o primeiro requisito a ser abordado é o cadastro de balancete. O primeiro atributo do balancete é o número da conta, que está vinculada a uma agência; depois temos a data inicial e a data final, que correspondem ao dia inicial do balancete e o dia final do balancete; o atributo saldo inicial, pode ser o saldo da conta selecionada trazido de forma automática ou especificado para melhor conveniência; no atributo descrição é definido o mês do respectivo balancete, para que a posteriori, seja feito a busca na base de dados por esse atributo. É disponibilizado um link para cadastro de conta, caso seja necessário cadastrar outra conta.

Figura 39: Tela cadastro de balancete.

Clicando no link cadastro de conta o usuário tem acesso a tela de cadastro de conta abaixo. Um atributo interessante no cadastro de conta é o número da agência, que por isso é disponibilizado um link para o cadastro de agência, caso necessário. Outro atributo é o saldo da conta, que é atualizado pelas entradas e saídas registradas no balancete, que tem esta conta previamente selecionada durante um determinado mês.

Figura 40: Tela cadastro de conta.

Caso seja necessário cadastrar uma agência basta clicar no link cadastro de agência, para o usuário ter acesso a tela abaixo. São disponibilizados dois links, o voltar é mera formalidade e volta para página financeiro e o endereço é para cadastro de endereço, caso seja necessário

Figura 41: Tela cadastro de agência.

Bem, como no módulo gerente não foi mostrado o requisito cadastro de endereço, vai ser mostrado aqui. O cadastro de endereço tem o atributo, logradouro, cep e distrito. A estratégia foi feita para que uma vez, uma rua cadastrada para um endereço, não seja necessário cadastrá outra vez para outro, apenas é cadastrado cada vez o número ao respectivo endereço na rua.

Figura 42: Tela cadastro de logradouro.

Caso o distrito não esteja cadastrado, basta clicar no link cadastro de distrito, para o usuário ter acesso a tela abaixo.

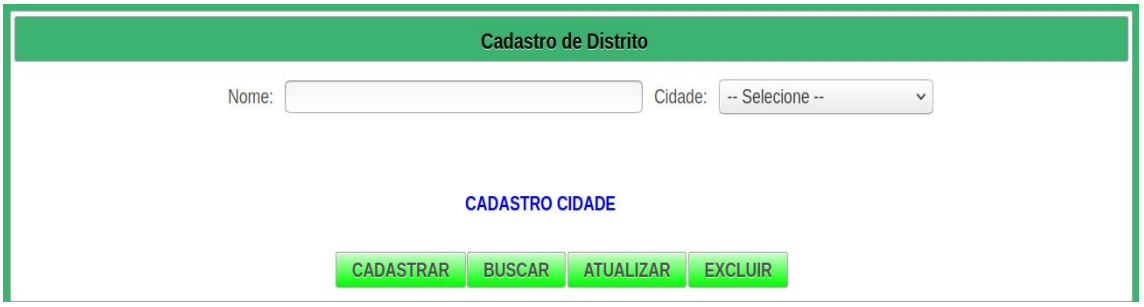


Figura 43: Tela cadastro de distrito.

Caso a cidade não esteja cadastrada, basta clicar no link cadastro de cidade, para o usuário ter acesso a tela abaixo.

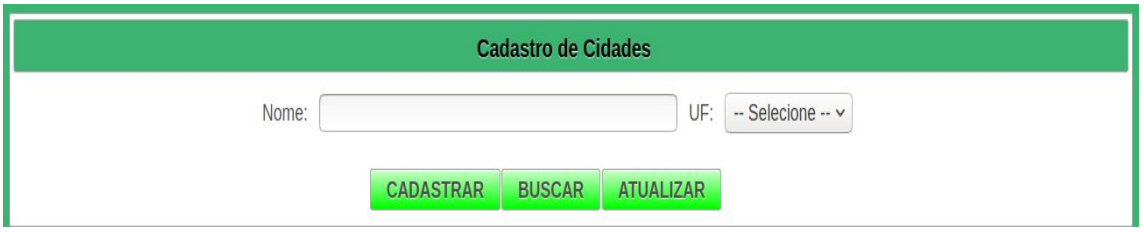


Figura 44: Tela cadastro de cidade.

Tendo um balancete previamente cadastrado e estando devidamente logado o usuário pode atualizar o balancete. Para isso basta clicar em atualizar balancete na página financeiro, para acessar a tela abaixo. Após selecionar o balancete de um determinado mês no checkbox descrição, os outros campos são preenchidos automaticamente. Permitindo apenas a data final para ser modificada. Como pode ser observado a primeira tabela lista as entradas, a segunda lista as saídas e a terceira referência os dados da conta. Os links registrar entrada e registrar saída, levam o usuário as páginas de registro de entrada ou saída, enquanto o link voltar retorna o usuário a tela financeiro. Por último um botão, para imprimir os registros atuais do balancete.

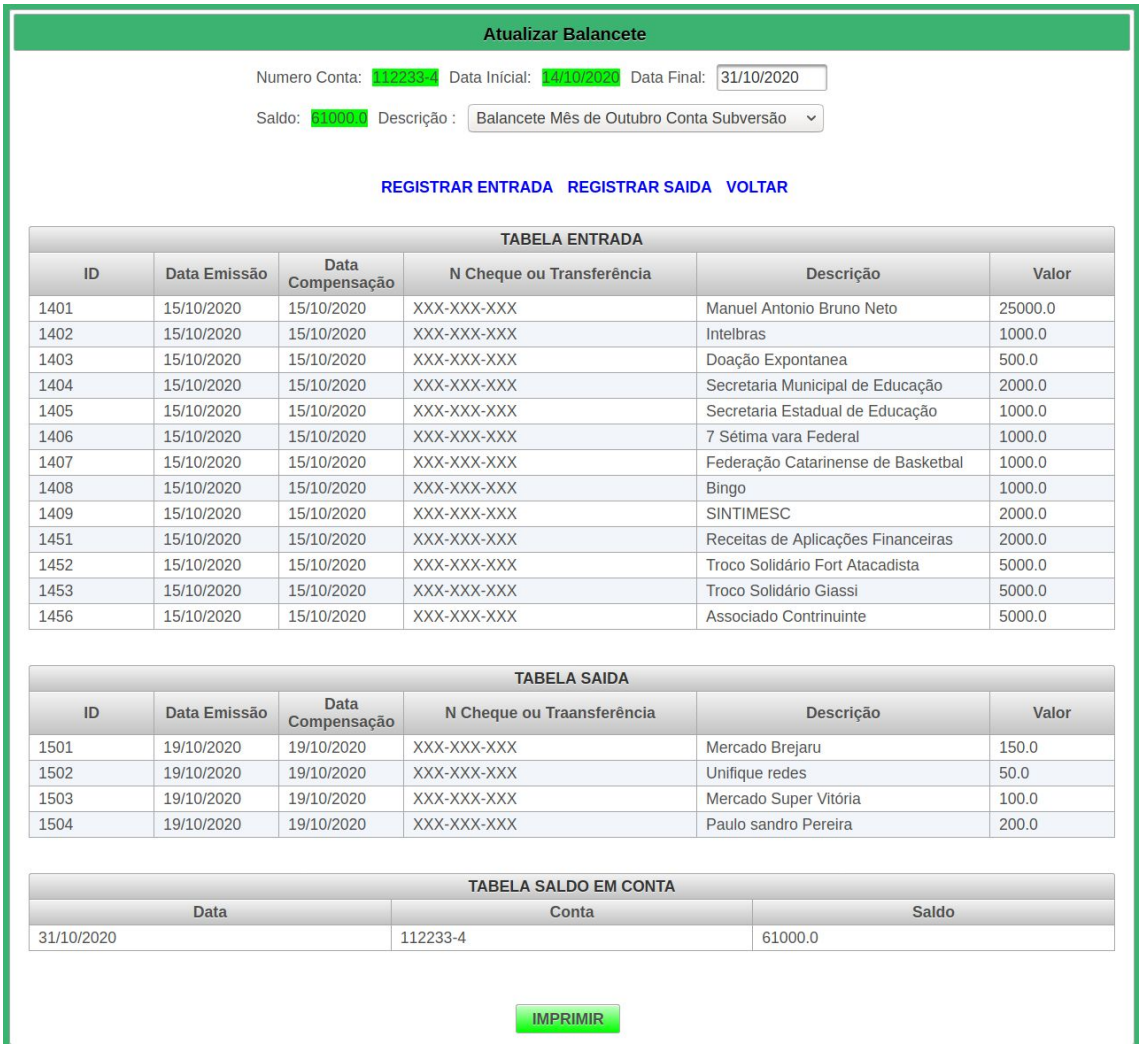


Figura 45: Tela atualiza balancete.

Estando o usuário na tela anterior está disponível para ele tanto o registrar entrada, quanto o registrar saída. Para ele registrar uma entrada ou remover uma entrada que se encontra na tabela que lista as entradas na tela anterior, basta a ele clicar em registrar entrada para ele ter acesso a tela abaixo. Nesta tela para ele excluir uma linha da tabela, basta a ele clicar em cima e clicar no botão excluir. Para ele registrar uma entrada é preciso preencher os atributos: data de emissão, a data da compensação, N cheque, descrição e o valor da entrada. O N cheque a princípio para o processo de registrar entrada é obsoleto, no entanto poderá não ser um dia, por isso existe um link para cadastro de N cheque ou transferência. Já para descrição é extremamente necessário que o usuário selecione o nome correto do patrocinador da entrada, para que as informações do balanço quando gerado, corresponda com a verdade. Caso a descrição de quem está patrocinando a entrada não esteja previamente disponível, o usuário deverá cadastrar a descrição, por isso tem um link disponível cadastro de descrição.

Figura 46: Tela registrar entrada.

Em algum momento da história da associação, em que se faça necessário o uso do N cheque, basta ao usuário clicar em N cheque ou transferência, para ele ter acesso a tela de cadastro abaixo.

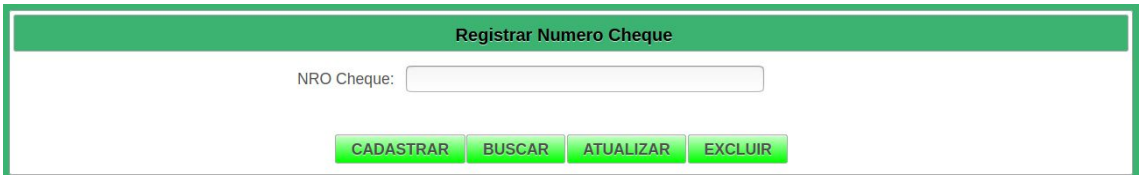


Figura 47: Tela cadastro N cheque ou transferência.

A descrição no entanto é muito importante para o processo de registrar entrada, pois é somado o valor de todas as entradas com o respectivo nome do patrocinador para gerar o balanço, como será averiguado mais adiante. A descrição do patrocinador pode vir de três fontes, uma descrição eventual que será cadastrado na tela de cadastro abaixo, ou das fontes mais prováveis: de instituições parceiras ou de associados contribuintes. Lembrando que os associados contribuintes são pessoas físicas que contribuem com um valor mensal para a associação. Caso o patrocinador seja uma instituição e que não esteja previamente cadastrado, faz-se necessário cadastrar uma instituição clicando no link na tela abaixo. Doutra forma se o patrocinador da entrada for um associado contribuinte, e ele não estiver previamente cadastrado, faz-se necessário clicar no link cadastro de associado contribuinte na tela abaixo para cadastrar. Para acessar o cadastro de descrição como na tela abaixo, é só o usuário clicar no link descrição, disponível na tela registrar entrada.

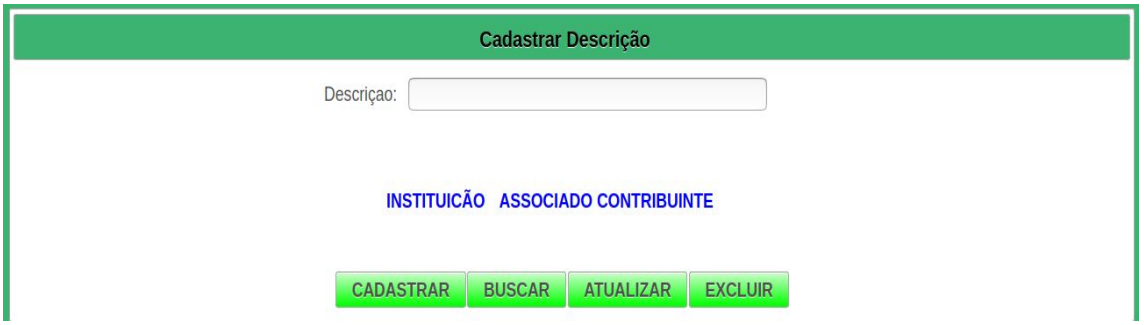


Figura 48: Tela cadastro de descrição.

O cadastro de instituição será abordado aqui no módulo financeiro de forma mais relevante. Caso uma nova parceria da associação com um novo comércio venha acontecer, exemplo, a associação fez uma nova parceria com o troco solidário do supermercado “Baratão”, neste caso é necessário acessar o link instituição na tela anterior para ter acesso a tela abaixo. Perceba que na tela de cadastro de instituição, um dos atributos é o tipo de instituição, que caso não esteja previamente cadastrado e listado no checkBox, é possível cadastrar acessando o link tipo instituição. Vale ressaltar que basicamente os tipos de instituições que podem ser patrocinadoras no registro de entrada atualmente são oitos: empresas privadas, institutos/fundações, eventos, pessoas físicas, governo federal, governo estadual, governo municipal, receitas financeiras. O objetivo principal desta estrutura é a conveniência na hora de gerar o balanço.

Figura 49: Tela cadastro de instituição.

Não é conveniente que outro tipo de instituição fora as definidas acima seja cadastrada, mas todavia porém, caso vier acontecer, está disponível para o usuário um link na tela anterior que ao ser clicado, retorna a tela de cadastro abaixo.

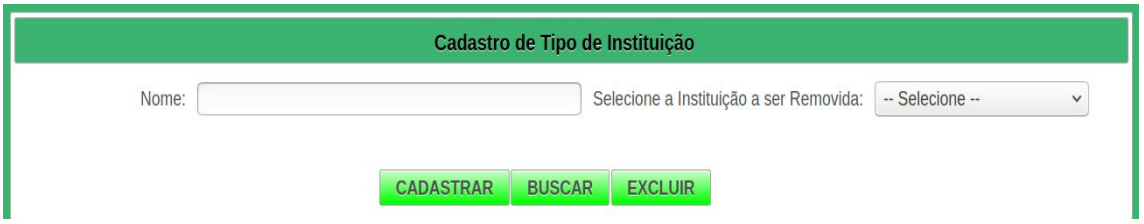


Figura 50: Tela cadastro de tipo instituição.

Caso a fonte patrocinadora do registro de entrada seja um associado contribuinte, e ele não esteja previamente cadastrado, basta clicar no link associado contribuinte na tela de cadastro de descrição, para ter acesso a tela abaixo e cadastrar o associado contribuinte.

Figura 51: Tela cadastro de associado contribuinte.

Outro requisito do módulo financeiro é o registrar saída, em termos gerais ele é semelhante ao registrar entrada, com poucas diferenças que serão abordadas aqui. Por simplificação será abordado só as diferenças entre esses dois requisitos. Na tela de atualizar balancete, basta clicar em registrar saída para o usuário acessar a tela de registro de saída abaixo. A única diferença perceptível está no link, cadastrar descrição saída; pois bem a descrição saída é diferente da descrição entrada. Caso a descrição de saída desejada, não esteja previamente cadastrada, basta clicar no link cadastrar descrição saída para cadastrar uma descrição de saída.

Figura 52: Tela registrar saída.

Uma descrição de saída pode vir de quatro opções diferentes, uma simples descrição eventual que é cadastrada na tela abaixo, ou das outras três, bem mais provável: prestadora de serviço, prestador físico de serviço, ou fornecedor cliente. A estrutura da saída foi construída semelhante a entrada, tendo em vista que a descrição de saída não é utilizada para o balanço convencional da associação. Como poderá ser observado mais a frente. Neste sentido, apenas serão mostrado os cadastros das outras três, a partir da tela abaixo.

Figura 53: Tela cadastro de descrição saída.

Caso necessário cadastrar prestadora de serviço, um link é disponibilizado na tela cadastro da descrição.

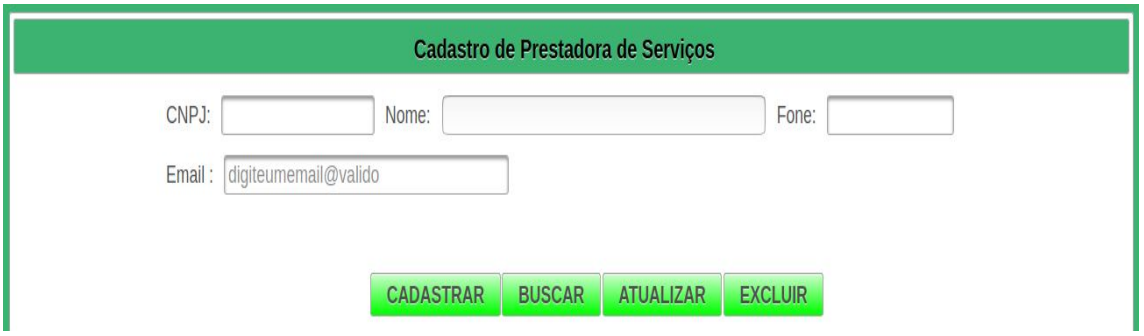


Figura 54: Tela cadastro prestadora de serviço.

Caso necessário cadastrar prestador de serviço físico, é disponibilizado um link na tela cadastro da descrição.

Figura 55: Tela cadastro prestador de serviço físico.

Caso necessário cadastrar fornecedor, é disponibilizado um link na tela cadastro da descrição.

Figura 56: Tela cadastro fornecedor.

Como último requisito do financeiro está o gerar balanço, pode ser anual ou uma prévia entre duas datas quaisquer selecionada. Na tela gerente, basta o usuário clicar em gerar balanço para acessar a página abaixo. Após selecionar as datas é só clicar no botão buscar. Dentro desta perspectiva do balanço, vale ressaltar que por motivos de conveniência da própria associação Pró Brejaru, o respectivo balanço é construído considerando apenas as entradas registradas nos balancetes, durante todo o ano. Perceba que, para cada tipo de instituição cadastrada, existe uma tabela gerada no balanço; na verdade a tabela receitas financeiras, vem dos ativos das contas, que sobram do balancete a cada mês; a tabela eventos, é fruto dos bingos e almoços, etc; e a tabela pessoas físicas, vem dos associados contribuintes, em que se destaca o patrocinador mor: Manoel Antônio Bruno Neto.

Figura 57: Tela gerar balanço.

Chegamos ao módulo operacional cujos requisitos serão abordado da esquerda para a direita, como procedemos nos módulos anteriores. Neste sentido, o usuário tendo efetuado o login com sucesso, terá acesso a página operacional abaixo. Vale ressaltar que o cadastro de endereço é conhecido tal como foi abordado no módulo anterior e não se faz necessário explicitá lo aqui novamente.

Figura 58: Página operacional.

Como observado na página acima o primeiro requisito é o cadastro de aluno e é passível de afirmar que, todo o módulo operacional, transcorre sobre este requisito, menos o relatório funcionário por departamento. Para que o cadastro de aluno seja concluído, é necessário que esteja previamente cadastrado, seu pai, mãe, irmãos, escola, cartório, problema de saúde, alergias, município de nascimento ou endereço; caso contrário, será necessário fazer estes cadastros primeiro, para depois efetivar o cadastro de aluno. Para isso é disponibilizado links, como mostrado na tela abaixo. Estando na página operacional, basta clicar no link aluno para o usuário acessar a tela de cadastro de aluno abaixo.

Figura 59: Tela cadastro aluno.

Perceba que é meio desnecessário mostrar todas as telas de cadastro representadas através dos links na tela anterior, cadastro de aluno. Ainda que, como mencionado, todas são utilizadas para cadastros prévios. Nesse sentido, será mostrada apenas as mais relevantes. Caso o pai ou a mãe de um aluno não esteja previamente cadastrado, basta o usuário clicar no link cadastro de pai ou mãe na tela anterior, para ter acesso a tela do pai ou mãe. Para exemplo vamos utilizar a tela cadastro de pai abaixo.

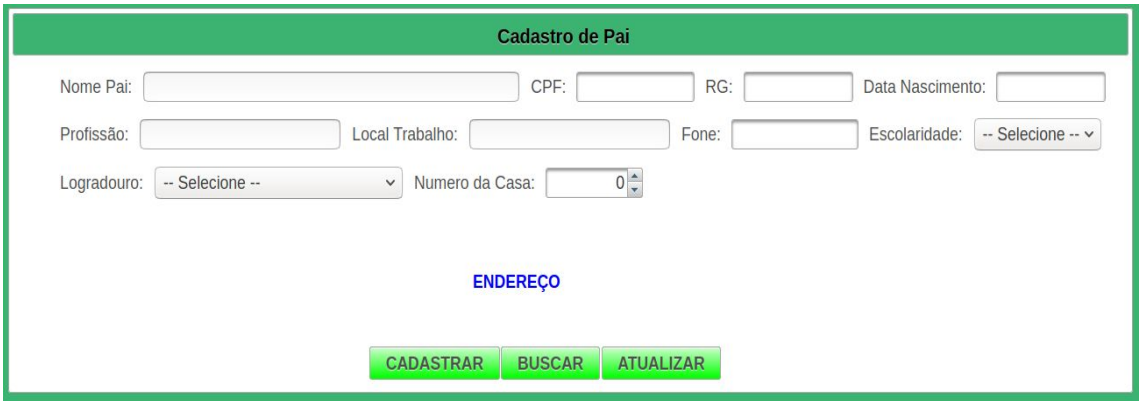


Figura 60: Tela cadastro pai.

Caso a escola que o aluno estuda não estiver previamente cadastrada, basta clicar no link cadastro de escola na tela de cadastro de aluno, para o usuário ter acesso a tela abaixo.

Figura 61: Tela cadastro escola

Se caso for o cartório onde o aluno foi registrado que não está previamente cadastrado, basta clicar no link cadastro de cartório na tela cadastro de aluno, para o usuário ter acesso a tela abaixo.

Figura 62: Tela cadastro cartório.

Assim segue da mesma forma para todos os outros links que na tela de cadastro de aluno, representam um cadastro prévio, requisito para o cadastro de aluno, caso não estiver cadastrado, faz-se necessário a priori o cadastro. Pela nossa convenção o próximo requisito seria registrar frequência, todavia, convém abordar o cadastro de turma, para melhor compreensão do sistema. Entre os atributos mais relevantes do cadastro de turma, está a lista de alunos que pode ser selecionados para fazer parte da turma, o do professor alocado responsável por aquela turma, a chamada da turma e o turno daquela turma. Estando os alunos cadastrados, basta o usuário estar na página operacional e clicar no link cadastro de turma, para acessar a tela abaixo.

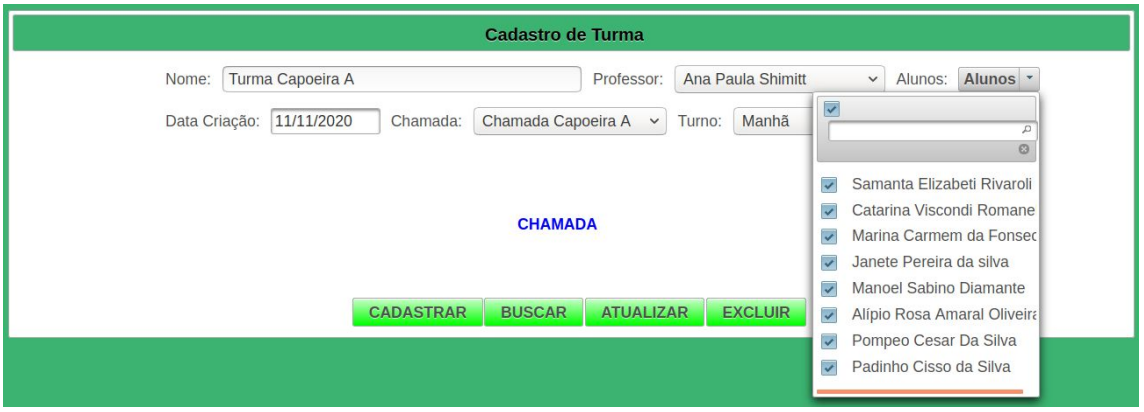


Figura 63: Tela cadastro turma.

Perceba que se ao cadastrar uma turma a chamada para esta turma não estiver cadastrada, existe um link para fazer o cadastro da chamada. O mesmo não acontece com o professor, porque o professor é na verdade um funcionário com o cargo de professor. Sendo assim, o cadastro do funcionário professor é feito pelo gerente. Para acessar a tela cadastro de chamada, basta clicar no link chamada na tela anterior.

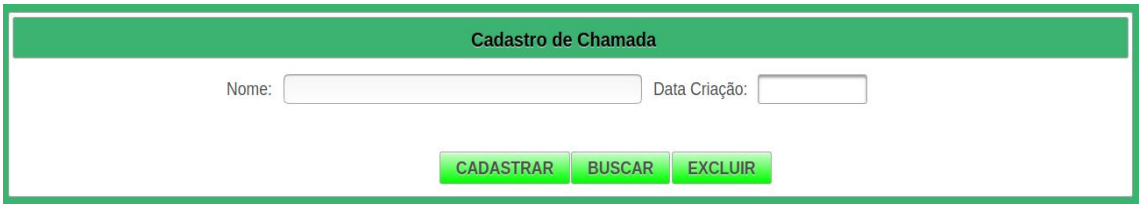


Figura 64: Tela cadastro chamada.

O próximo requisito abordado é o registrar frequência; nesse sentido vale ressaltar que, tanto o registro, quanto a atualização de uma frequência é feito na mesma página. Para registrar frequência, após acessar a tela abaixo, basta selecionar uma chamada e clicar em buscar, para alunos ausente basta apenas desmarcar o selectBooleanCheckbox, selecionar a data e clicar em cadastrar. Para atualizar, é necessário selecionar a chamada e selecionar a data que se deseja atualizar, depois clicar em buscar, após fazer a alteração é só clicar em atualizar. Por padrão, o selectBooleanCheckbox vem previamente preenchida como true, basta desmarcar para ficar falso.

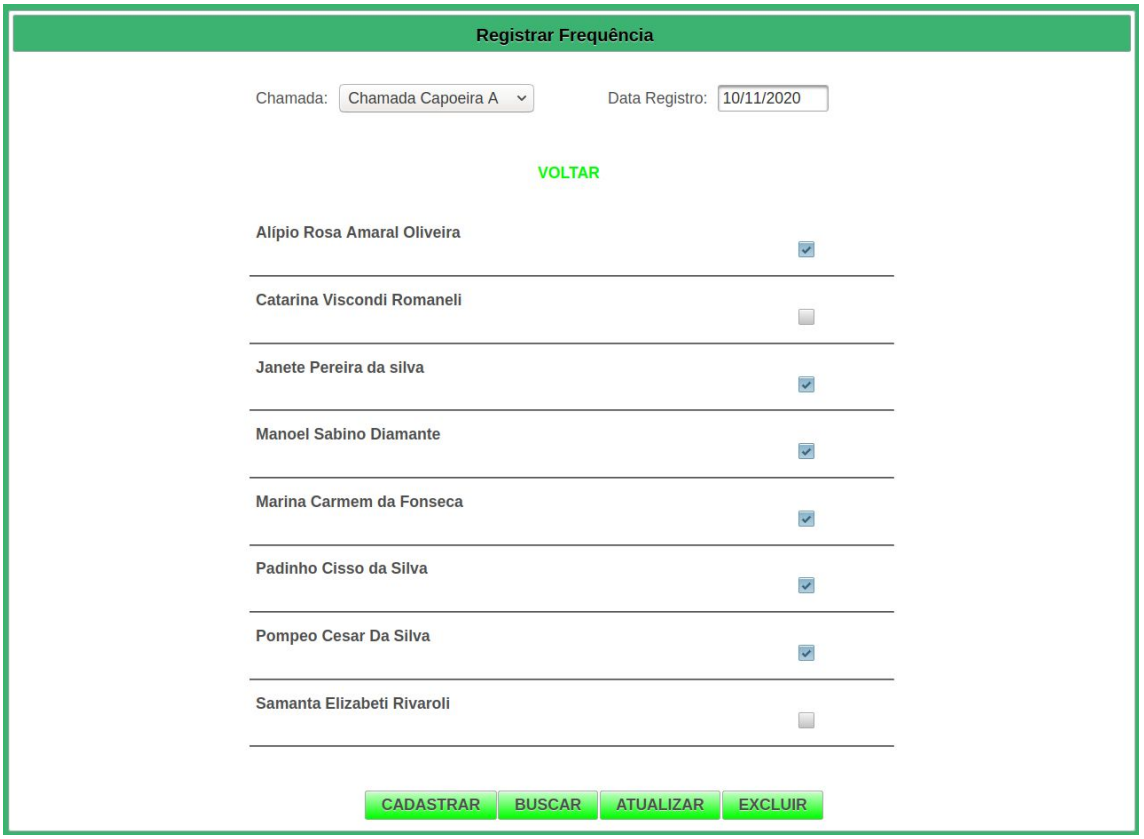
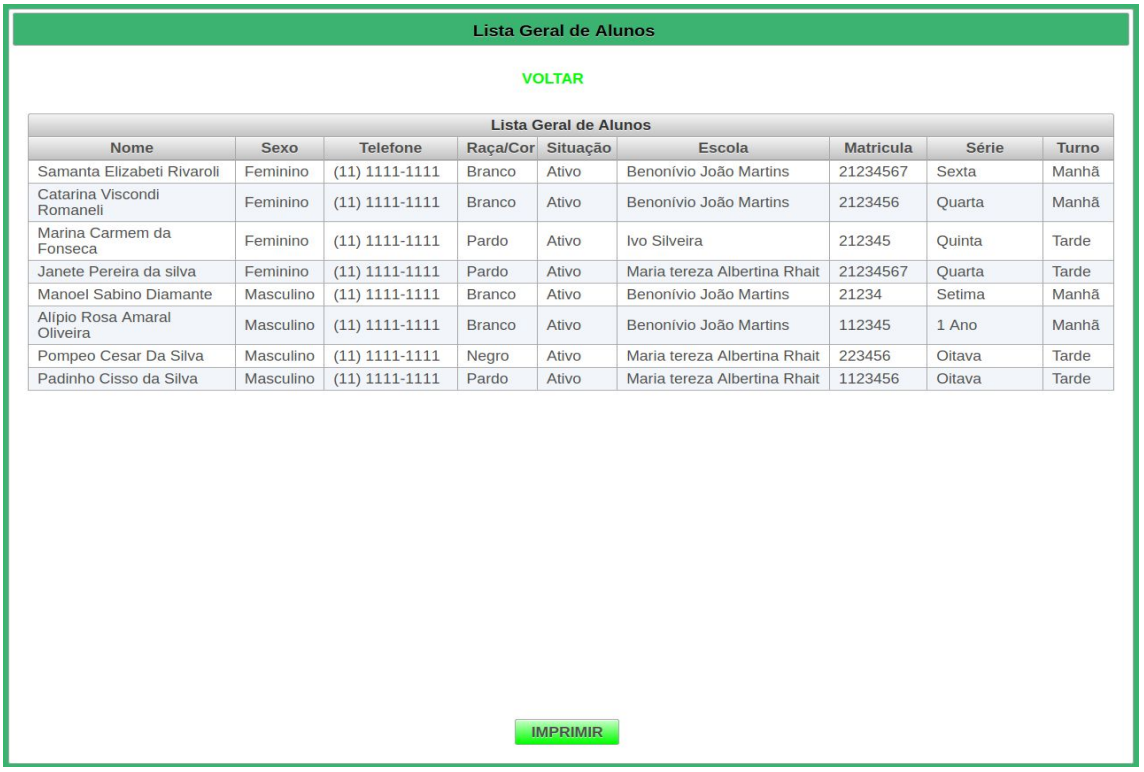


Figura 65: Tela registro e atualização frequência.

Após o cadastro dos alunos, uma série de relatórios estará disponíveis para o usuário do módulo operacional. Neste sentido, a partir de agora será abordado os requisitos referentes aos relatórios. Vale ressaltar que esses relatórios são solicitados pela prefeitura municipal de Palhoça, como objetivo de fiscalizar a lisura das atividades da associação, tendo em vista a parceria que existe entre as duas instituições. Sabendo que a maioria dos relatórios é gerado de forma automática, a partir do click no próprio link do relatório da página operacional. Após gerar os relatórios, é possível imprimir-los, para isso basta clicar no botão imprimir na própria página do relatório, fazendo com que seja gerado um PDF e disponibilizado pelo browser ao usuário. Iniciamos então a abordagem dos requisitos de relatórios da esquerda para a direita; sendo assim, estando na página operacional, basta clicar em relatório geral de aluno, para ter acesso a página de relatório abaixo.

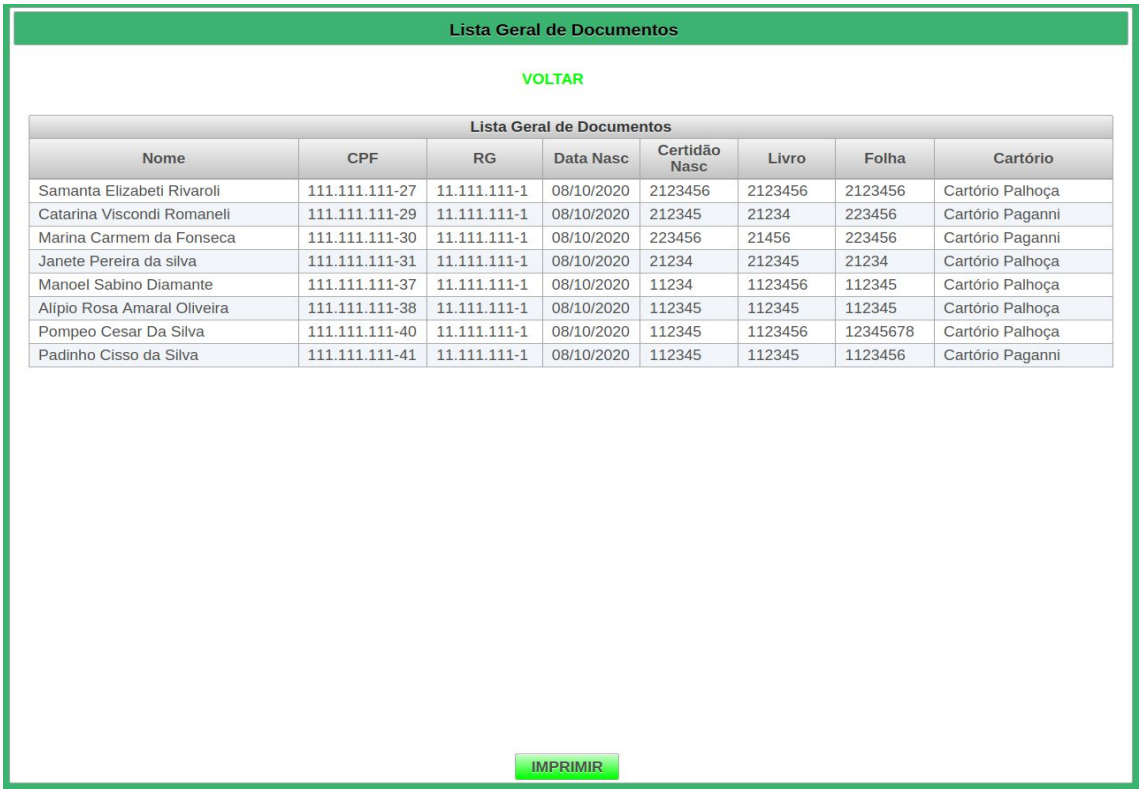


Lista Geral de Alunos

Nome	Sexo	Telefone	Raça/Cor	Situação	Escola	Matricula	Série	Turno
Samanta Elizabeti Rivaroli	Feminino	(11) 1111-1111	Branco	Ativo	Benonívio João Martins	21234567	Sexta	Manhã
Catarina Viscondi Romaneli	Feminino	(11) 1111-1111	Branco	Ativo	Benonívio João Martins	2123456	Quarta	Manhã
Marina Carmem da Fonseca	Feminino	(11) 1111-1111	Pardo	Ativo	Ivo Silveira	212345	Quinta	Tarde
Janete Pereira da silva	Feminino	(11) 1111-1111	Pardo	Ativo	Maria tereza Albertina Rhait	21234567	Quarta	Tarde
Manoel Sabino Diamante	Masculino	(11) 1111-1111	Branco	Ativo	Benonívio João Martins	21234	Setima	Manhã
Alípio Rosa Amaral Oliveira	Masculino	(11) 1111-1111	Branco	Ativo	Benonívio João Martins	112345	1 Ano	Manhã
Pompeo Cesar Da Silva	Masculino	(11) 1111-1111	Negro	Ativo	Maria tereza Albertina Rhait	223456	Oitava	Tarde
Padinho Cisso da Silva	Masculino	(11) 1111-1111	Pardo	Ativo	Maria tereza Albertina Rhait	1123456	Oitava	Tarde

Figura 66: Tela relatório geral alunos.

Semelhantemente para gerar o relatório geral de documentos, basta fazer o mesmo processo; na tela operacional, clicar em relatório geral de documentos.



Lista Geral de Documentos

Nome	CPF	RG	Data Nasc	Certidão Nasc	Livro	Folha	Cartório
Samanta Elizabeti Rivaroli	111.111.111-27	11.111.111-1	08/10/2020	2123456	2123456	2123456	Cartório Palhoça
Catarina Viscondi Romaneli	111.111.111-29	11.111.111-1	08/10/2020	212345	21234	223456	Cartório Paganni
Marina Carmem da Fonseca	111.111.111-30	11.111.111-1	08/10/2020	223456	21456	223456	Cartório Paganni
Janete Pereira da silva	111.111.111-31	11.111.111-1	08/10/2020	21234	212345	21234	Cartório Palhoça
Manoel Sabino Diamante	111.111.111-37	11.111.111-1	08/10/2020	11234	1123456	112345	Cartório Palhoça
Alípio Rosa Amaral Oliveira	111.111.111-38	11.111.111-1	08/10/2020	112345	112345	112345	Cartório Palhoça
Pompeo Cesar Da Silva	111.111.111-40	11.111.111-1	08/10/2020	112345	1123456	12345678	Cartório Palhoça
Padinho Cisso da Silva	111.111.111-41	11.111.111-1	08/10/2020	112345	112345	1123456	Cartório Paganni

Figura 67: Tela relatório geral documentos.

Semelhantemente para gerar o relatório de lista familiar, basta fazer o mesmo processo. Perceba que é possível que um mesmo pai e mãe possa ter mais de um filho na associação, e um aluno ter um ou mais irmão fazendo parte da associação, o que de fato é bem comum.

Figura 68: Tela relatório familiar.

Para gerar a lista de aluno em desistente, basta na página operacional clicar em lista de aluno desistente.

Figura 69: Tela relatório desistente por ano.

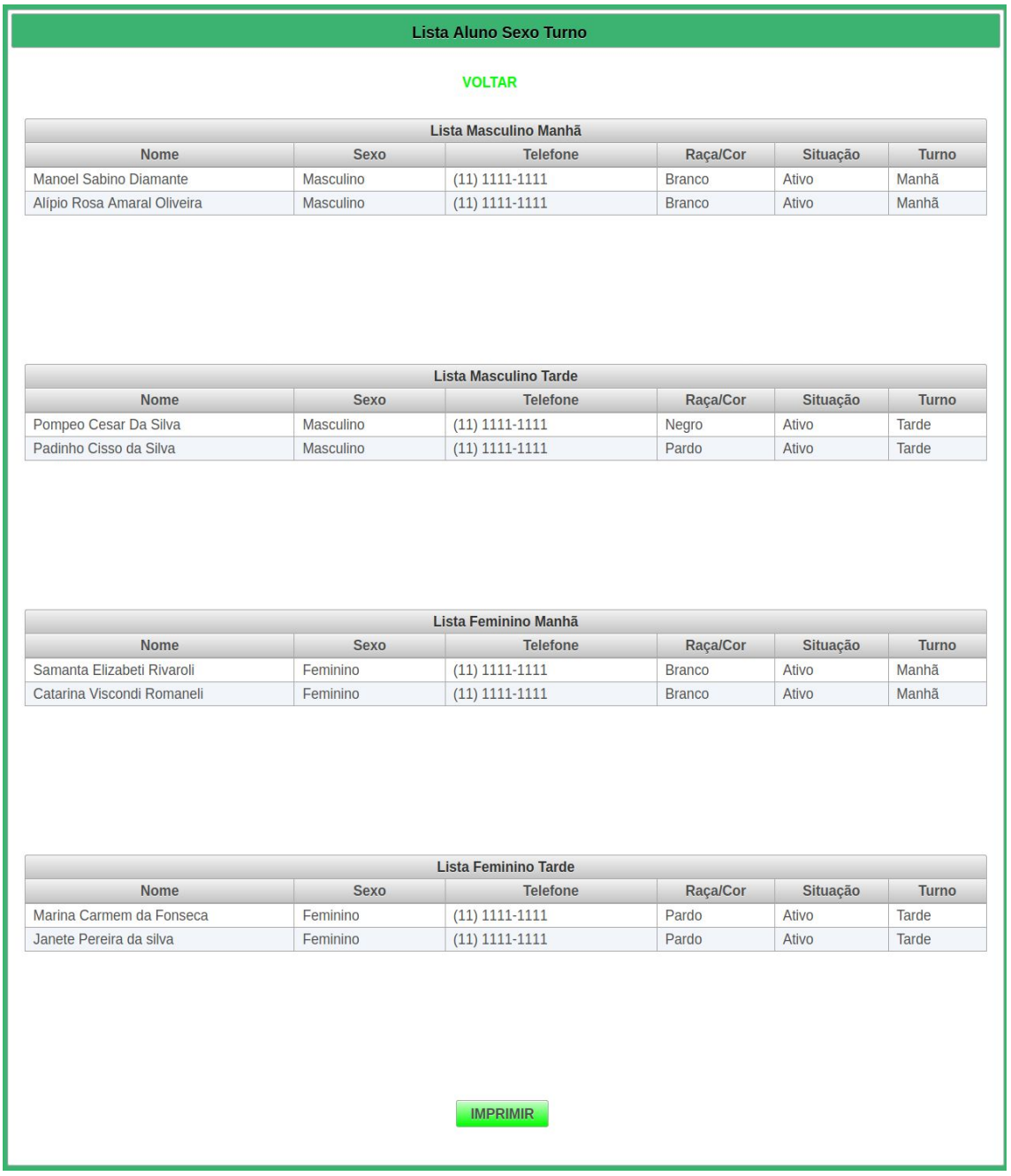
Para gerar a lista de aluno em espera, basta na página operacional clicar em lista de aluno em espera.

Figura 70: Tela relatório lista de espera.

Já para gerar o relatório mensal de frequência de aluno por turma, basta clicar no link mensal frequência de aluno por turma, na página operacional. Posteriormente é necessário selecionar a turma do aluno e o mês em que se quer pesquisar, então é só clicar no botão buscar para o relatório ser gerado, e caso queira imprimir, é só clicar no botão imprimir.

Figura 71: Tela relatório mensal frequência de aluno por turma.

Para gerar o relatório mensal de aluno por turno e sexo, basta clicar no link mensal de aluno por turno e sexo na página operacional. Para que o relatório seja gerado automaticamente, como mostrado na tela abaixo. Para processar esse relatório, utilizamos dois atributos essenciais, o turno do cadastro de turma, e o turno do cadastro de aluno.



Lista Aluno Sexo Turno

VOLTAR

Lista Masculino Manhã

Nome	Sexo	Telefone	Raça/Cor	Situação	Turno
Manoel Sabino Diamante	Masculino	(11) 1111-1111	Branco	Ativo	Manhã
Alípio Rosa Amaral Oliveira	Masculino	(11) 1111-1111	Branco	Ativo	Manhã

Lista Masculino Tarde

Nome	Sexo	Telefone	Raça/Cor	Situação	Turno
Pompeo Cesar Da Silva	Masculino	(11) 1111-1111	Negro	Ativo	Tarde
Padinho Cisso da Silva	Masculino	(11) 1111-1111	Pardo	Ativo	Tarde

Lista Feminino Manhã

Nome	Sexo	Telefone	Raça/Cor	Situação	Turno
Samanta Elizabeti Rivaroli	Feminino	(11) 1111-1111	Branco	Ativo	Manhã
Catarina Viscondi Romaneli	Feminino	(11) 1111-1111	Branco	Ativo	Manhã

Lista Feminino Tarde

Nome	Sexo	Telefone	Raça/Cor	Situação	Turno
Marina Carmem da Fonseca	Feminino	(11) 1111-1111	Pardo	Ativo	Tarde
Janete Pereira da silva	Feminino	(11) 1111-1111	Pardo	Ativo	Tarde

IMPRIMIR

Figura 72: Tela relatório mensal aluno por turno e sexo.

Já o relatório mensal de alunos por atividade, segue um rito parecido para acessar a página, no entanto, para que o relatório seja efetivamente concluído é necessário selecionar a atividade que se deseja obter o relatório em um checkbox, após selecionar a atividade o relatório é gerado de forma automática como visualizado na tela abaixo. Um atributo relevante listado neste relatório é sem sombra de dúvida, as turmas que este aluno participa nesta atividade. Pois uma atividade que é ministrada duas ou três vezes no mesmo turno, em dias diferente durante a semana, pode ser praticada pelo mesmo aluno, apenas alterasse a turma que ele está participando. Por exemplo, a atividade Capoeira, pode ser ministrada duas vezes por semana, em dias diferentes e no mesmo turno.

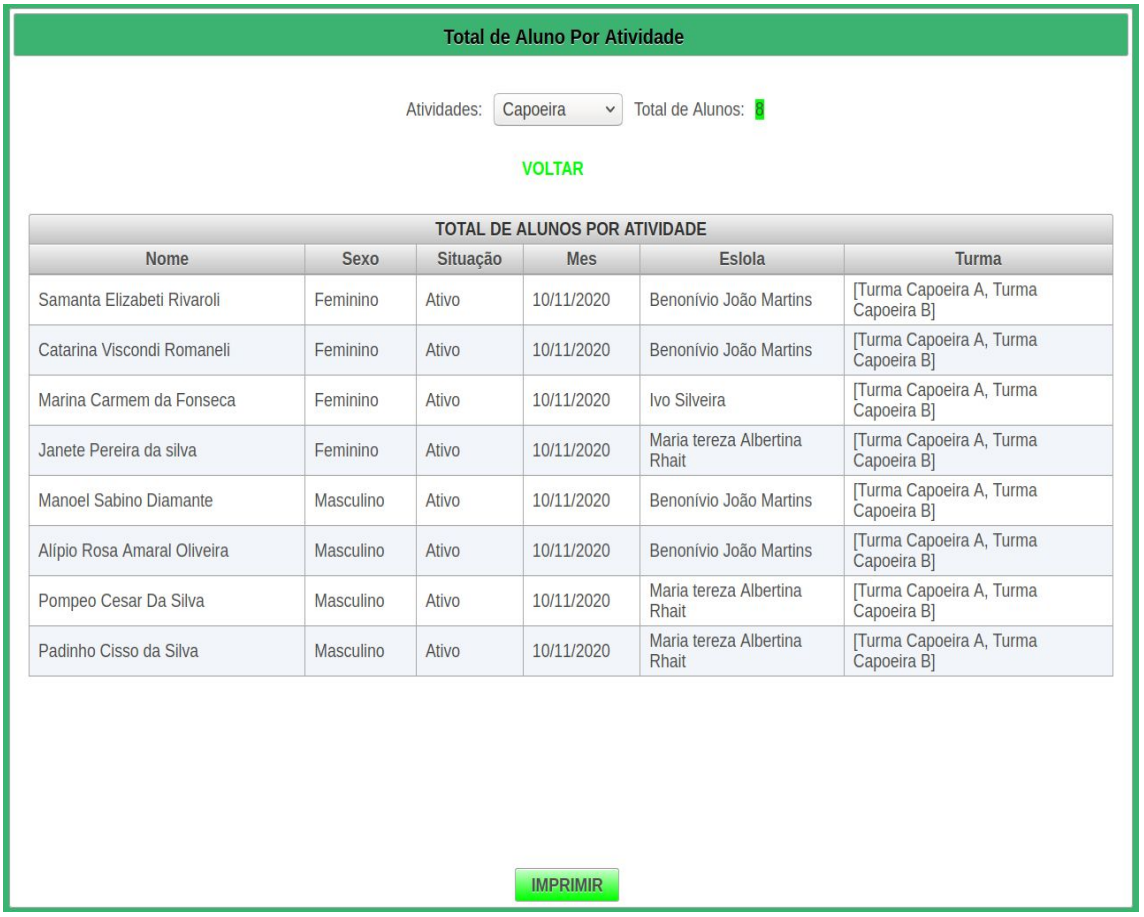


Figura 73: Tela relatório mensal alunos por atividade.

Já para gerar o relatório de funcionário por departamento, basta clicar no link funcionário por departamento, na página operacional. De forma automática é gerado o relatório como mostrado na tela abaixo.

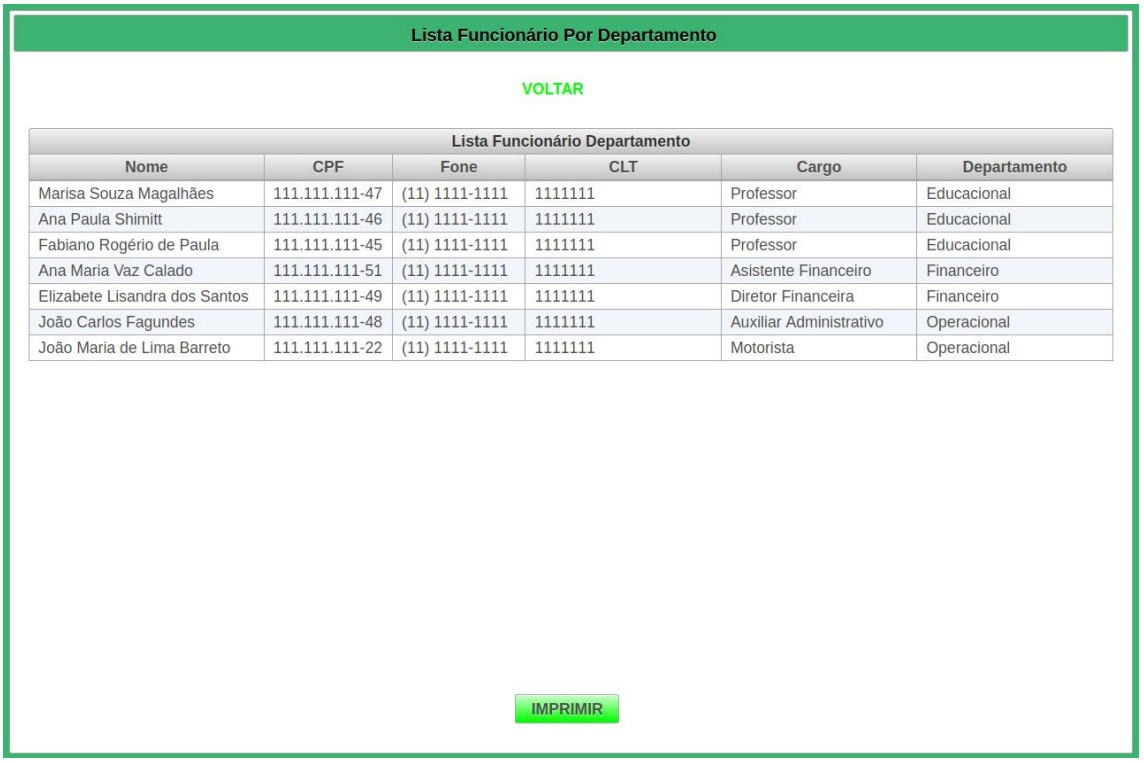


Figura 74: Tela relatório funcionário por departamento.

Dessa forma então foi concluída a implementação deste sistema que tem por objetivo, além de ser utilizado pela associação Pró Brejaru, definir o tema deste trabalho de conclusão de curso.

4.5 Modelagem da Base de Dados

A princípio a base de dados do sistema foi gerada pela convenção do padrão JEE, pela Java Persistence API, considerando a especificação JPA. Nesse contexto, não foi necessário fazer uma modelagem relacional para servir como diretriz para implementação da base de dados, tendo em vista que, na especificação jpa a implementação da base de dados acontece de forma automática baseado na classe entity e o mapeamento objeto relacional.

5 Conclusão

Diante da predisposição em realizar essa pesquisa segundo as características da associação Pró Brejaru, considerando o estado da arte e a definição de um sistema web, como solução para sistematizar alguns de seus processos, é possível afirmar que o projeto foi concluído com êxito. Sendo que quase todos os objetivos definidos previamente na engenharia de requisitos, na modelagem e arquitetura dos sistema foram concluídos com eficácia. Podemos caracterizar a conclusão deste sistema web, como uma primeira versão 1.0. De ínicio a perspectiva de abstração com a engenharia de requisitos, era apenas conhecer a demanda da associação como um todo, referente a um sistema que contemplasse todas as suas necessidades. Após começar o desenvolvimento percebeu-se que era possível definir o escopo do sistema, de forma que abrangesse todos os requisitos demandado pela engenharia de requisitos. Desta forma então a equipe definiu as estratégias para que o sistema fosse iniciado e concluído. Vale ressaltar que o sistema apesar de estar como um todo concluído, ele não foi implantado e nem validado pela associação Pró Brejaru. Pois configurações referentes a implantação, como por exemplo, configuração de ambiente de servidor e segurança criptográfica, foram abordado no modo desenvolvimento e não de implantação. No entanto, todos os teste de validação em modo de desenvolvimento obtiveram resultado satisfatório, passível de se confirmar.

Na escolha deste trabalho de conclusão de curso, alguns objetivos foram priorizados, entre eles conhecer as tecnologias que fazem parte do padrão de projeto JEE, e mensurar o nível do aprendizado abstraído durante os cursos ministrados na graduação, além obviamente, do objeto sistema web que será utilizado pela associação. Do ponto de vista da primeira parte a equipe se dá por satisfeita, pois aprendeu suficientemente as tecnologias para concluir o sistema web. Quanto a segunda parte, porém, vai estar a cargo dos membros da banca que avaliará este trabalho.

Do ponto de vista de todas as dificuldades, cabe aqui ressaltar duas, que se não foram as maiores, estão entre elas: a primeira foi a dificuldade de não falar inglês, melhor sabem vossas senhorias do que eu, o quanto é imprescindível essa língua para quem atua na área da tecnologia; a segunda, foi a perda do computador, o que evidentemente contribui completamente para o atraso do trabalho de conclusão de curso. No entanto, tanto uma, quanto a outra, não impediu que este trabalho fosse concluído, pelo contrário, serviram como estímulo ao aprendizado da língua inglesa, o que de fato aconteceu em início.

5.1 Trabalhos futuros.

No decorrer do desenvolvimento ficou mais nítido as decisões acertadas e equivocada desta primeira versão, e que poderão ser melhoradas de forma eficiente em versões futuras. Considerando a versão 1.0 deste sistema web desenvolvido, fica evidente que certos aspectos do sistema pode ser melhorado em oportunidades futuras, tais como as próprias interfaces do cliente deste sistema, além de alguns atributos que poderiam ter feito parte de certas entidades e não fizeram. Assim como, outros requisitos podem ser complementados, ou propriamente inicialmente implementados, como por exemplo, incrementar o requisito cadastro de doação, permitindo que empresas possam ser aceitas como doadoras, e não apenas pessoa física, como acontece nessa primeira versão.

Seria excelente também um novo trabalho de conclusão de curso que além da abordagem da implementação, pudesse pesquisar a abordagem necessária para a implantação deste sistema web. E até mesmo corrigir requisitos concluídos e que não atingirão as necessidades processuais de forma eficaz, a partir da validação completa deste sistema web pela associação Pró Brejaru. Mesmo pela velocidade da transformação nas relações sociais e institucionais causada pela avanço tecnológico, quanto pelo a natureza da evolução dos processos.

Referências Bibliográficas:

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA MICROS E PEQUENAS EMPRESAS. **Encontro de estudos sobre empreendorismo e gestão de pequenas empresas** - 2012 Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv1898.pdf> Acesso em: 17 mar. 2018.

UNICAMP. **Terceiro Setor cria novo paradigma social, diz especialista.** Disponível em: <https://www.unicamp.br/unicamp/noticias/2012/10/10/terceiro-setor-cria-novo-paradigma-social-e-se-expande-no-mundo-todo-diz> Acesso em: 21 fev. 2018.

MAXWELL. **Terceiro setor: uma breve abordagem.** Disponível em: <https://www.maxwell.vrac.puc-rio.br/18272/18272_3.PDF> Acesso em: 13 set. 2018.

EURICO de ANDRADE AZEVEDO: **Organização Social disponibilizado em**: <http://www.pge.sp.gov.br/centrodeestudos/revistaspge/revista5/5rev6.htm> Acesso em: 01 mar. 2019.

TRACTOR: **Quantas pessoas têm acesso à internet no mundo**? - disponível em: <https://www.tracto.com.br/quantas-pessoas-tem-acesso-a-internet-no-mundo/> Acesso em: 15 mai. 2018.

RENATO PEZZOTTI: **Quantas pessoas têm acesso à internet no mundo**? Disponível em:

<https://economia.uol.com.br/noticias/redacao/2019/04/01/com-39-bilhoes-de-usuarios-no-mundo-o-que-acontece-na-web-em-um-minuto.htm> Acesso em: 01 abr. 2018.

DEVMEDIA, **Principais padrões de projeto J2EE**, disponível em:

<https://www.devmedia.com.br/principais-padroes-j2ee-para-a-construcao-de-

aplicacoes-nao-distribuidas-parte-i/1812> Acesso em: 10 ago. 2019.

CONCEITOS: **Visão Geral do J2EE, (Java 2 Platform Enterprise Edition)** disponível em:

<http://walderson.com/IBM/RUP7/LargeProjects/tech.j2ee/guidances/concepts/java_2_platform_enterprise_edition_j2ee_overview_9A95BA45.html> Acesso em: 10 jan. 2018.

ORACLE. **Java Platform, Enterprise Edition (Java EE) 7.** Disponível em: <https://docs.oracle.com/javaee/7/index.html> Acesso em: 10 mar. 2018.

PRIMEFACES. **Primefaces User's Guides.** Disponível em: <https://www.primefaces.org/documentation/> Acesso em: 10 mar. 2019.

ECLIPSE FOUNDATION. **EclipseLink Documentation Center.** Disponível em: <https://www.eclipse.org/eclipselink/documentation/> Acesso em: 01 jun. 2019.

VISUAL PARADIGM. **Visual Paradigm User's Guide.** Disponível em:

<https://www.visual-paradigm.com/support/documents/> Acesso em: 01 set. 2019.

JAVA. **Java Server Faces Tecnologia. 2017.** Disponível em: https://www.oracle.com/java/technologies/javaserverfaces.html. Acesso em: 10 fev. 2019.

SILVA, I. P. **Desenvolvendo com Hibernate. 2014**. Disponível em:

<http://www.devmedia.com.br/artigo-java-magazine-73-desenvolvendo-com-

hibernate/14756>. Acesso em: 01 fev. 2019.

GLASSFISH. **Glassfish Server The Open Source Java EE Reference Implementation.** Disponível em: <https://javaee.github.io/glassfish/documentation> Acesso em: 02 fev. 2018

Apêndice A - Artigo

SISTEMA WEB PARA GERÊNCIA DE PROJETOS SOCIAIS

Universidade Federal de Santa Catarina - UFSC

Departamento de Informática e Estatística - INE

Curso de Sistemas de Informação - CSI

Caixa Postal 476 – 88040900 – Florianópolis – SC – Brazil

Petterson Sued Trindade

trindadepetterson@gmail.com

Abstract. Considering the need for organizations to systematize their processes, due to the changes in social and institutional relations caused by the advancement of technologies, this work proposed to know the processes of the association Pro Brejaru, with the objective of developing a system to systematize their processes. After knowing the processes, system development techniques were used, such as requirements analysis, modeling and systems architecture, to specify the system. Concepts, state of the art, design patterns and available technologies were also studied. Finally, the specification was used to implement a system on demand for this association

Resumo. Considerando a necessidade das organizações de sistematizar seus processos, devido às mudanças nas relações sociais e institucionais causada pelo avanço das tecnologias, este trabalho propôs conhecer os processos da associação Pró Brejaru, com o objetivo de desenvolver um sistema para sistematizar seus processos. Após conhecer os processos foi utilizado técnicas de desenvolvimento de sistemas como, análise de requisitos, modelagem e arquitetura de sistemas, para especificar o sistema. Também foram estudados conceitos, estado da arte, padrões de projeto e tecnologias disponíveis. Por fim, utilizou-se da especificação, para implementar um sistema sobre demanda para esta associação.

1. Introdução

Atualmente é imprescindível a utilização das Tecnologias da Informação para manter relações socioeconômica entre sociedade civil e instituições, seja ela privada ou pública, é o vem sendo observado por pesquisadores ao longo do tempo, como Alavi e Bergeron entre outros citados abaixo. No contexto atual, a Tecnologia da Informação desempenha tarefa crucial para o alcance dos objetivos das instituições (ALAVI & JOACHIMSTHALER, 1992; BERGERON, BATEU & RAYMOND, 1991). O avanço tecnológico tem exercido papel relevante nos diversos setores da economia de maneira que as organizações necessitam buscar mecanismos adequados diante da nova realidade.

É comum à maior parte dos trabalhadores atualmente o uso de tecnologias na execução de suas atividades e tarefas no âmbito das organizações, devido às mudanças nas relações sociais e institucionais causada pelo avanço da tecnologia. Administrar uma organização sem o auxílio de um sistema de informação é extremamente difícil e trabalhoso. Muitas vezes em pequenas organizações, que por algum motivo ainda não tem seus processos administrativo informatizados, muitos dos dados resultantes das atividades de seus processos são registrados em documentos de papel e armazenados em um conjunto de arquivos. Um dos motivos é caracterizado segundo SOUZA (2004, p 65) porque, os micros e pequenos empresários negligenciam as atividades de planejamento e controle dos seus negócios, considerando-as como uma burocracia desnecessária. Segundo estudos disponibilizados pelo VIIEGEPE(2012). Quanto aos recursos tecnológicos disponíveis, verificou-se que a maioria das empresas ainda não os utiliza, porém apresentam-se dispostas em adquirir tal recurso. Tal estudo afirma que em uma população de 90 entrevistados, 30% possuem computador, e só 9% dispõem de

sistemas de informação ou de programas de automação comercial. A falta de um sistema de informação para gerenciar uma organização, e executar seu processos, acaba gerando alguns problemas como por exemplo, perda de tempo, tanto para armazenar, quanto para pesquisar determinado documento neste conjunto de arquivos, perda de informações tendo em vista que muitos destes documentos podem em algum momento ser extraviado ou deteriorados pela ação do tempo. Além de existir à necessidade crescente de espaço, à medida que a organização e seu conjunto de dados e documentos crescem.

A importância de um sistema para gerenciar documentos e processos em uma pequena empresa, está diretamente ligado à sobrevivência da mesma no sistema ambiente. Como pode ser facilmente constatado através de pesquisas elaboradas pelo SEBRAE. Observa-se, portanto, que dos dois fatores condicionantes de sucesso apontado pelos empresários, ambos estão relacionados à informação e conhecimento, seja interna ou externamente. Em geral, as organizações não estão preparadas para o gerenciamento de seu ativo informacional, originando o acúmulo desnecessário, ou o descarte de informações importante. A execução das atividades cotidianas também é afetada pela deficiência informacional, não havendo uma metodologia ou modelo estabelecido, o funcionário age de forma empírica, ou seja, da forma que acredita estar correto, acarretando em perda de qualidade e gerando retrabalho (SEBRAE, 2010).

Sendo então de extrema importância a uma organização ainda que pequena, um sistema de informação para executar seus processos. De forma que venha garantir a qualidade dos seus dados e informações ao longo do tempo. Através do próprio

armazenamento destes dados e informações em uma base de dados ágil e segura. Permitindo a utilização destes dados e informações de forma muito mais eficiente, preservando a integridade e confiabilidade destes dados e informações, além de providenciar agilidade para executar os processos da organização, e que por consequência, seja mais sistemático e eficiente gerenciar as organizações.

2. Associação Pró Brejaru

Conhecendo a organização associação Pró Brejaru, foi possível perceber que apesar de seus processos estarem bem definidos e um tanto complexos, não era utilizado um sistema de software para execução ao menos da maior parte deles. Sendo que na execução dos seus processos, eram produzidos muitos dados e informações em papel e armazenados em pastas de arquivos, fazendo com que a integridade e a utilidade de muitos destes dados e informações estejam um tanto limitado ao longo do tempo. Além de exigir que uma quantidade de papel e espaço seja necessário para armazenar estes arquivos. Enquanto por outro lado, dados e informações importantes para a execução eficiente de alguns dos processos críticos da organização, estejam indisponíveis. Considerando tais aspectos, buscou-se no mercado um sistema de informação que pudesse solucionar estes problemas prévios. No entanto, as condições oportunas da organização em relação às necessidades financeira, restringiram a busca à algumas possibilidades de soluções.

Em um primeiro momento, foram consideradas as características dos sistemas disponíveis no mercado, com a real necessidade da organização. Neste aspecto, foi constatado que algumas das particularidades dos processos da organização não estavam contidos nas funcionalidades providas pelos sistemas. Em contrapartida, os sistemas disponibilizavam funcionalidades que fugiam o escopo dos processos da organização. Em um segundo momento, foi levado em consideração a limitação financeira da organização. Nesse aspecto, foi constatado efetivamente que todos os sistemas encontrados só eram possíveis de serem obtidos, através da compra ou licenciamento, o que o atual momento da organização tornaria inoportuno. Sendo

assim então, este trabalho visa buscar uma solução para esta organização, levando em consideração estes dois pontos de vista, e sobre demanda específica.

3. Proposta

A partir das pesquisas elaboradas durante o embasamento teórico e o estado da arte, foi possível perceber a relevância de um sistema de informação na gerência de uma organização. O quanto é importante para uma organização ter seus processos sistematizados e uma base de dados ágil e segura, que proporcione informação aos gestores e facilite a administração da organização e a execução de seus processos.

Nesta perspectiva, conhecendo a associação Pró Brejaru e suas necessidades de um sistema para a execução eficiente de seu processos, é que propomos especificar, modelar e construir um sistema, para ajudar a gerenciar esta associação e a executar seus processos de forma mais eficiente, e que principalmente atenda as necessidades específicas da organização, e que proporcione uma base de dados consistente confiável e informativa ao administrador.

3.1. Definição da arquitetura

A partir da engenharia de requisitos e da modelagem do sistema foi possível definir uma arquitetura para o sistema. A princípio poderia ser considerada para satisfazer a necessidade de solução para os problemas característicos de software da associação Pró Brejaru, um software para desktop. Porém, alguns dos requisitos pré-definidos nos remetem às características de um site. O que sendo analisado de forma mais específicas, percebeu-se que seria melhor propor um sistema WEB considerando a arquitetura WEB 2.0, cliente servidor. Alguns dos requisitos preveem páginas que de certa forma representam conteúdo “estático”, as quais são interfaces de comunicação da associação com a sociedade. E poderão ser acessadas por qualquer usuário da internet, pois essas páginas servirão em última análise de promoção do trabalho da associação. Tendo em vista porém que a quantidade de acesso às páginas, não é uma questão crítica, pois o número de acessos em determinado momento não tende a sobrecarregar o sistema, optou-se por utilizar o padrão de projeto java J2EE.

Os principais serviços disponibilizados pela plataforma J2EE destinam-se a suprir as necessidade de aplicações empresariais distribuídas, isto é, aquelas que necessitam da flexibilidade de disponibilizar acesso à sua lógica de negócios e dados para diferentes tipos de dispositivos cliente (navegadores, dispositivos móveis, aplicações desktop, etc.), ou para outras aplicações residentes na mesma empresa ou fora. A figura abaixo ilustra um ambiente J2EE típico.

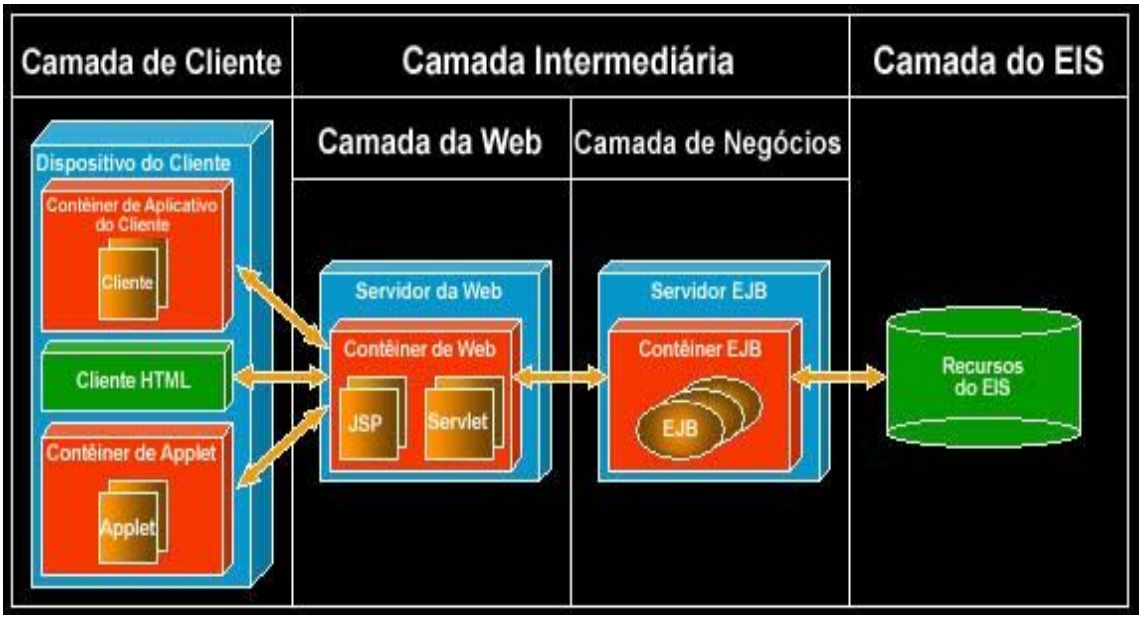


Figura 2 : Representa as camadas do ambiente J2EE.

Aplicações distribuídas são comumente compostas de uma camada cliente, que implementa uma interface com usuário, e uma ou mais camadas intermediárias, que processam a lógica do negócio e provêem serviços a camada cliente, e outra denominada de, Enterprise Information System ou EIS, formado por sistemas legados e banco de dados. A infraestrutura formada pela J2EE possibilita que estas camadas, possivelmente localizadas em máquinas diferentes, possam se comunicar remotamente e juntas compor uma aplicação (Padrões de Projeto, DEVMEDIA).

https://www.devmedia.com.br/principais-padroes-j2ee-para-a-construcao-de-aplicacoes-nao-distribuidas-parte-i/1812

Assim então dentro desta perspectiva de padrão de projeto J2EE, as tecnologias de ferramentas selecionadas para concluir o trabalho, foram as mais tradicionais convencionadas por essa plataforma. De forma que elas são conhecidas de todos os profissionais que atuam na área da TI, e seria desnecessário apresentar-las

completamente aqui. No entanto, para a banca melhor conhecer o projeto, é interessante explicitar-las: para ide foi utilizado o NetBeans 8.2, para as interfaces com clientes foi utilizado o framework o JSF 2.3 primefaces 6.1, para servidor o GlassFish 4.1.1, para os CRUDS JPA utilizando o framework Eclipselink 2.1, para banco de dados o Java DB derby 10.10.2.0, para a modelagem do sistema foi utilizado o Visual Paradigm 16.2.

3.2. Desenvolvimento

A implementação foi concluída considerando a engenharia de requisitos a modelagem e a arquitetura do sistema a priori definida. Utilizando as ferramentas e tecnologias de desenvolvimento se software do padrão JEE cliente servidor, para codificar o sistema. Cabendo ressaltar ou propriamente definir no entanto que o presente sistema deve ser classificado como uma versão 1.0. Dentro desta perspectiva como uma primeira versão o sistema objetivamente foi implementado da seguinte forma, um site contendo seis páginas: home, quem somos, o que fazemos, com quem fazemos, como fazemos e contatos. Além de um conjunto de CRUDs que contemplou, toda a sistematização dos processos da associação Pró Brejaru, segundo as características especificada na modelagem construída através da especificação. Após teste dos requisitos codificados em modo de desenvolvimento, é passível de afirmar que todos os requisitos foram concluídos aparentemente de forma eficiente. Pois todos os requisitos codificados foram submetidos a testes, de forma específica, como também compondo módulo e por último o sistema de forma completa.

3.3. Requisitos codificados

Devido ao tamanho do sistema e a quantidade de requisitos codificados, seria meio inconveniente explicitá los todos aqui. No entanto, a partir do requisito cadastro de aluno, que será mostrado aqui é possível subentender como o sistema como um todo foi implementado. O módulo operacional cujos requisitos serão abordado da esquerda para a direita, como procedemos nos módulos anteriores. Neste sentido, o usuário tendo efetuado o login com sucesso, terá acesso a página operacional

abaixo. Vale ressaltar que o cadastro de endereço é conhecido tal como foi abordado no módulo anterior e não se faz necessário explicitá lo aqui novamente.

Figura 1: Tela operacional.

Como observado na página acima o primeiro requisito é o cadastro de aluno e é passível de afirmar que, todo o módulo operacional, transcorre sobre este requisito, menos o relatório funcionário por departamento. Para que o cadastro de aluno seja concluído, é necessário que esteja previamente cadastrado, seu pai, mãe, irmãos, escola, cartório, problema de saúde, alergias, município de nascimento ou endereço; caso contrário, será necessário fazer estes cadastros primeiro, para depois efetivar o cadastro de aluno. Para isso é disponibilizado links, como mostrado na tela abaixo. Estando na página operacional, basta clicar no link aluno para o usuário acessar a tela de cadastro de aluno abaixo.

Figura 2: Tela cadastro aluno.

Perceba que é meio desnecessário mostrar todas as telas de cadastro representadas através dos links na tela anterior, cadastro de aluno. Ainda que, como mencionado, todas são utilizadas para cadastros prévios. Nesse sentido, será mostrada apenas o cadastro do pai. Caso o pai de um aluno não esteja previamente cadastrado, basta o usuário clicar no link cadastro de pai na tela anterior, para ter acesso a tela do cadastro de pai.

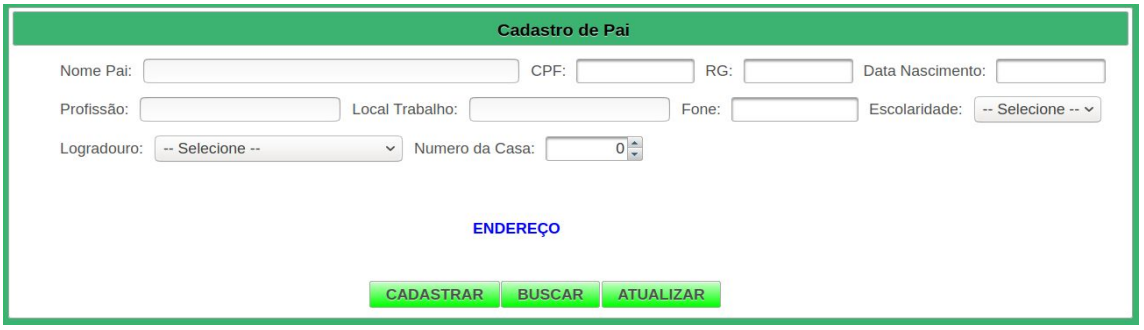


Figura 3: Tela cadastro pai.

Após os cadastros dos alunos forem efetuados com sucesso, uma série de relatórios estará disponíveis para o usuário do módulo operacional. Neste sentido, iremos apresentar apenas o relatório mensal de alunos por atividade. Este relatório segue um rito parecido para acessar a página, no entanto, para que o relatório seja efetivamente concluído é necessário selecionar a atividade que se deseja obter o

relatório em um checkbox, após selecionar a atividade o relatório é gerado de forma automática como visualizado na tela abaixo. Um atributo relevante listado neste relatório é sem sombra de dúvida, as turmas que este aluno participa nesta atividade. Pois uma atividade que é ministrada duas ou três vezes no mesmo turno, em dias diferente durante a semana, pode ser praticada pelo mesmo aluno, apenas alterasse a turma que ele está participando. Por exemplo, a atividade Capoeira, pode ser ministrada duas vezes por semana, em dias diferentes e no mesmo turno.

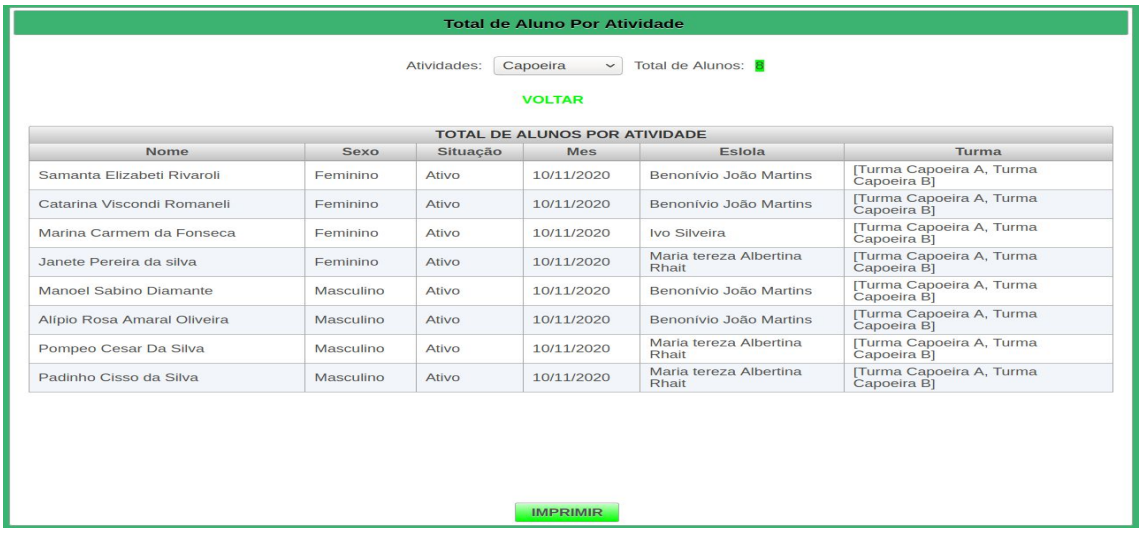


Figura 4: Tela relatório mensal aluno por atividade.

3.4. Testes de requisitos codificados

Após a codificação de cada requisito pré-definido na engenharia de requisitos e modelagem, testes unitários foram efetuados sobre cada requisito. Neste sentido, vamos analisar como foram feito os testes, segundo as características de cada um, inicializamos pelo cadastro de aluno. No geral a tecnologia de interface com cliente utilizado neste trabalho primefaces 6.1, disponibiliza uma bean validator que pode ser estendido para validar os seu componentes, caso o componente específico não seja previamente validado. Depois seguimos fazendo as validações padrões de variáveis de linguagem, minimizando as possibilidades de erros através de seleções em checkbox, e impossibilitando o cadastro com campo vazio. A imagem do código abaixo sintetiza e garante a impossibilidade o cadastro de campo vazio.

```
public String cadastrarAluno(){
 if(nome.equals("") || id > 1 || rg.equals("") || data_matricula == null || dataexpedicao == null || datanascimento == null
    || orgaoexpedidor == null || certidaonascimento <=0 || livro <= 0 || folha <=0 || cartorio == null || sexo == null
    || rua ==null || numero_casa <=0 || fone.equals("") || municipio_nascimento == null || raca ==null || situacao == null
    || list_alergias_selecionadas.size() <= 0 || list_problemas_saude_selecionados.size() <= 0
        || list_irmaos_selecionados.size() <= 0|| pai == null || mae == null || bolsafamilia == null || escola == null
        || numeromatricula <= 0 || serie == null || turno == null || cpf_aluno.equals("")){
    FacesContext ctx = FacesContext.getCurrentInstance();
    FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário inválido", "Usuário inválido");
    ctx.addMessage(null, msg);
    FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage("Preencha os Campos Corretamente!"))
        return "preencha os campos";
```

Caso tentarmos efetuar o cadastro de um aluno sem selecionar algum atributo, uma mensagem de erro é enviada ao usuário, como mostrado na tela de cadastro abaixo. Serve para o método de cadastrar, atualizar e excluir.

Figura 5: Tela de erro cadastro aluno.

Nas páginas de relatório também foram efetuada validações como por exemplo, caso o botão buscar seja clicado sem estar preenchido os atributos necessário, uma mensagem de erro é enviada ao usuário; o botão imprimir só é renderizado quando, o botão buscar é clicado, ou quando um selectOneMenu é selecionado; no caso abaixo, além de selecionar selectOneMenu também é necessário definir as datas, que referencia o mês da busca, para então fazer a busca da frequência dos alunos por turma.

Figura 6: Tela de erro frequência aluno por turma.

Selecionado a turma e as datas corretamente é efetuado a busca dos alunos que fazem parte da turma e a frequência do aluno nesta turma, e o botão imprimir é renderizado, como na tela abaixo.

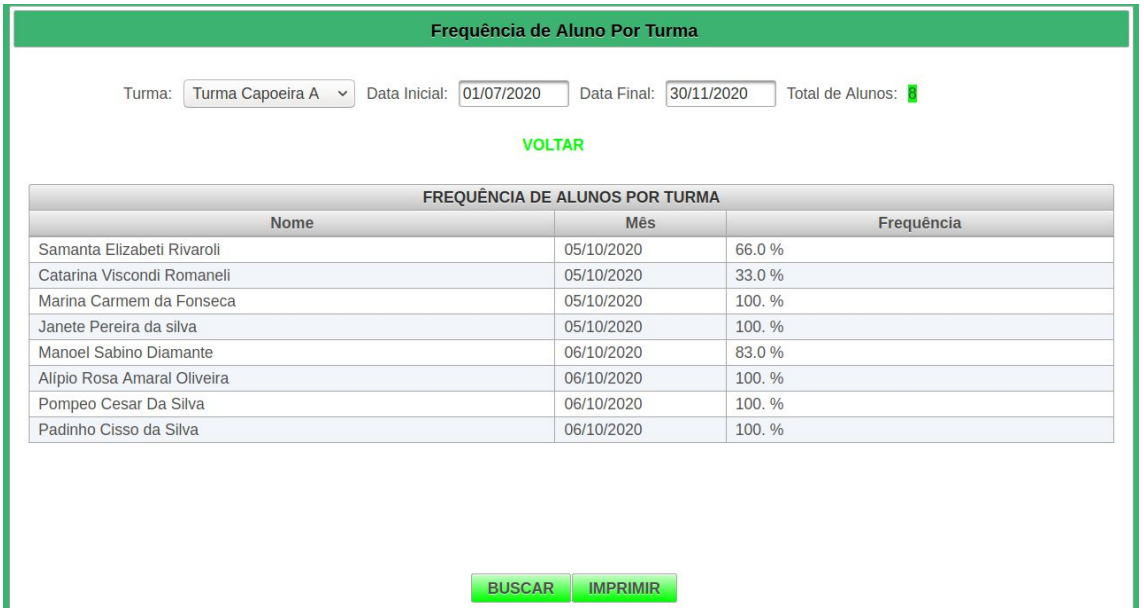


Figura 7: Tela relatório frequência aluno por turma

Aqui mostramos algumas das telas que validaram desde os cadastro de dados atualizações exclusões e a geração de relatórios, do sistema codificado. No geral as validações de todo o sistema seguiu basicamente estas características, variando de

acordo com a necessidade obviamente. Sendo então passível de afirmar que todos os requisitos previamente definidos, foram eficientemente codificados.

4. Conclusão

Diante da predisposição em realizar essa pesquisa segundo as características da associação Pró Brejaru, considerando o estado da arte e a definição de um sistema web, como solução para sistematizar alguns de seus processos, é possível afirmar que o projeto foi concluído com êxito. Sendo que quase todos os objetivos definidos previamente na engenharia de requisitos, na modelagem e arquitetura dos sistema foram concluídos com eficácia. Podemos caracterizar a conclusão deste sistema web, como uma primeira versão 1.0. De ínicio a perspectiva de abstração com a engenharia de requisitos, era apenas conhecer a demanda da associação como um todo, referente a um sistema que contemplasse todas as suas necessidades. Após começar o desenvolvimento percebeu-se que era possível definir o escopo do sistema, de forma que abrangesse todos os requisitos demandado pela engenharia de requisitos. Desta forma então a equipe definiu as estratégias para que o sistema fosse iniciado e concluído. Vale ressaltar que o sistema apesar de estar como um todo concluído, ele não foi implantado e nem validado pela associação Pró Brejaru. Pois configurações referentes a implantação, como por exemplo, configuração de ambiente de servidor e segurança criptográfica, foram abordado no modo desenvolvimento e não de implantação. No entanto, todos os teste de validação em modo de desenvolvimento obtiveram resultado satisfatório, passível de se confirmar.

Na escolha deste trabalho de conclusão de curso, alguns objetivos foram priorizados, entre eles conhecer as tecnologias que fazem parte do padrão de projeto JEE, e mensurar o nível do aprendizado abstraído durante os cursos ministrados na graduação, além obviamente, do objeto sistema web que será utilizado pela associação. Do ponto de vista da primeira parte a equipe se dá por satisfeita, pois aprendeu suficientemente as tecnologias para concluir o sistema web. Quanto a segunda parte, porém, vai estar a cargo dos membros da banca que avaliará este trabalho.

Do ponto de vista de todas as dificuldades, cabe aqui ressaltar duas, que se não foram as maiores, estão entre elas: a primeira foi a dificuldade de não falar inglês, melhor sabem vossas senhorias do que eu, o quanto é imprescindível essa língua para quem atua na área da tecnologia; a segunda, foi a perda do computador, o que evidentemente contribui completamente para o atraso do trabalho de conclusão de curso. No entanto, tanto uma, quanto a outra, não impediu que este trabalho fosse concluído, pelo contrário, serviram como estímulo ao aprendizado da língua inglesa, o que de fato aconteceu em início.

4.1. Trabalhos futuros

No decorrer do desenvolvimento ficou mais nítido as decisões acertadas e equivocada desta primeira versão, e que poderão ser melhoradas de forma eficiente em versões futuras. Considerando a versão 1.0 deste sistema web desenvolvido, fica evidente que certos aspectos do sistema pode ser melhorado em oportunidades futuras, tais como as próprias interfaces do cliente deste sistema, além de alguns atributos que poderiam ter feito parte de certas entidades e não fizeram. Assim como, outros requisitos podem ser complementados, ou propriamente inicialmente implementados, como por exemplo, incrementar o requisito cadastro de doação, permitindo que empresas possam ser aceitas como doadoras, e não apenas pessoa física, como acontece nessa primeira versão.

Seria excelente também um novo trabalho de conclusão de curso que além da abordagem da implementação, pudesse pesquisar a abordagem necessária para a implantação deste sistema web. E até mesmo corrigir requisitos concluídos e que não atingirão as necessidades processuais de forma eficaz, a partir da validação completa deste sistema web pela associação Pró Brejaru. Mesmo pela velocidade da transformação nas relações sociais e institucionais causada pela avanço tecnológico, quanto pelo a natureza da evolução dos processos.

Referências Bibliográficas

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA MICROS E PEQUENAS EMPRESAS. **Encontro de estudos sobre empreendorismo e gestão de pequenas**

empresas - 2012 Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv1898.pdf> Acesso em: 17 mar. 2018.

DEVMEDIA, **Principais padrões de projeto J2EE**, disponível em:

<https://www.devmedia.com.br/principais-padroes-j2ee-para-a-construcao-de-aplicacoes-nao-distribuidas-parte-i/1812> Acesso em: 10 ago. 2019.

CONCEITOS: **Visão Geral do J2EE, (Java 2 Platform Enterprise Edition)** disponível em:

<http://walderson.com/IBM/RUP7/LargeProjects/tech.j2ee/guidances/concepts/java_2_platform_enterprise_edition_j2ee_overview_9A95BA45.html> Acesso em: 10 jan. 2018.

ORACLE. **Java Platform, Enterprise Edition (Java EE) 7.** Disponível em: <https://docs.oracle.com/javaee/7/index.html> Acesso em: 10 mar. 2018.

JAVA. **Java Server Faces Tecnologia. 2017.** Disponível em: https://www.oracle.com/java/technologies/javaserverfaces.html. Acesso em: 10 fev. 2019.

SILVA, I. P. **Desenvolvendo com Hibernate. 2014**. Disponível em:

<http://www.devmedia.com.br/artigo-java-magazine-73-desenvolvendo-com-hibernate/14756>. Acesso em: 01 fev. 2019.

GLASSFISH. **Glassfish Server The Open Source Java EE Reference Implementation.** Disponível em: <https://javaee.github.io/glassfish/documentation> Acesso em: 02 fev. 2018

UNICAMP. **Terceiro Setor cria novo paradigma social, diz especialista.** Disponível em: <https://www.unicamp.br/unicamp/noticias/2012/10/10/terceiro-setor-cria-novo-paradigma-social-e-se-expande-no-mundo-todo-diz> Acesso em: 21 fev. 2018.

EURICO de ANDRADE AZEVEDO: **Organização Social disponibilizado em**: <http://www.pge.sp.gov.br/centrodeestudos/revistaspge/revista5/5rev6.htm> Acesso em: 01 mar. 2019.

TRACTOR: **Quantas pessoas têm acesso à internet no mundo**? - disponível em: <https://www.tracto.com.br/quantas-pessoas-tem-acesso-a-internet-no-mundo/> Acesso em: 15 mai. 2018.

Apêndice B - Código Fonte

A quantidade de código para colocar aqui é muito grande, por esse motivo, será colado aqui poucas classes, na sequência: AlunoJSFManagedBean, AlunoEntity, GeraBalancoJSFManagedBean, os três beans de sessão: Gerente, Financeiro e operacional.

Módulo operacional:

```
package bean_gerenciveis;

import beans_sessao.FinanceiroSessionBean;

import beans_sessao.GerenteSessionBean;

import beans_sessao.OperacionalSessionBean;

import entidades.AlergiaEntity;

import entidades.AlunoEntity;

import entidades.CartorioEntity;

import entidades.CidadeEntity;

import entidades.EnderecoEntity;

import entidades.EscolaEntity;

import entidades.IrmaoEntity;

import entidades.MaeEntity;

import entidades.PaiEntity;

import entidades.PessoaEntity;

import entidades.ProblemaSaudeEntity;

import javax.inject.Named;

import javax.enterprise.context.SessionScoped;

import java.io.Serializable;
```

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.ejb.EJB;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.inject.Inject;

/**
 *
 * @author petterson
 */
@Named(value = "alunoJSFManagedBean")
@SessionScoped
public class AlunoJSFManagedBean implements Serializable {

    @EJB
    OperacionalSessionBean ops;
    @EJB
    FinanceiroSessionBean fin;
    @EJB
    GerenteSessionBean geb;
    long id;
```

```
String nome;

String erro;

Date data_matricula;

String fone;

//@CPF (message="Cpf Inválido")

String cpf_aluno;

String rg;

String orgaoexpedidor;

Date dataexpedicao;

String sexo;

Date datanascimento;

int certidaonascimento;

int livro;

int folha;

String cartorio;

String rua;

int numero_casa;

String referencia;

String municipio_nascimento;

String raca;

String escola;

int numeromatricula;

String serie;
```

```
String turno;

String bolsafamilia;

String pai;

String mae;

String situacao;

ArrayList<String> list_problemas;

String problema_entity;

ArrayList<String> list_cartorios;

String cartorio_entity;

ArrayList<String> list_ruas;

String rua_entity;

ArrayList<String> list_municipio_nascimentos;

String cidade_entity;

ArrayList<String> list_escolas;

String escola_entity;

ArrayList<String> list_alergias;

String alergia_entity;

ArrayList<String> list_pais;

String pai_entity;

ArrayList<String> list_maes;

String mae_entity;

ArrayList<String> list_irmaos;

String irmao_entity;
```

```
ArrayList<String> list_irmaos_selecionados;

ArrayList<String> list_alergias_selecionadas;

ArrayList<String> list_problemas_saude_selecionados;

ArrayList<String> list_sexo;

ArrayList<String> list_racas;

ArrayList<String> list_series;

ArrayList<String> list_turnos;

ArrayList<String> list_bolsa_familias;

ArrayList<String> list_situacoes;

ArrayList<String> list_orgaos_espedidor;

PessoaEntity p;

@Inject

TurmaSFManagedBean tum;



public AlunoJSFManagedBean() {

this.list_cartorios = new ArrayList<>();

this.list_problemas = new ArrayList<>();

this.list_alergias = new ArrayList<>();

this.list_escolas = new ArrayList<>();

this.list_irmaos = new ArrayList<>();

this.list_maes = new ArrayList<>();
```

```
this.list_pais = new ArrayList<>();
this.list_municipio_nascimentos = new ArrayList<>();
this.list_ruas = new ArrayList<>();
this.list_orgaos_espedidor = new ArrayList<>();
this.list_orgaos_espedidor.add("Secretaria Segurança Publica SC");
this.list_orgaos_espedidor.add("Outro");
}

public long getId() {
return id;
}

public void setId(long id) {
this.id = id;
}

public String getNome() {
return nome;
}

public void setNome(String nome) {
this.nome = nome;
}
```

```
public String getErro() {
return erro;
}

public void setErro(String erro) {
this.erro = erro;
}

public Date getData_matricula() {
return data_matricula;
}

public void setData_matricula(Date data_matricula) {
this.data_matricula = data_matricula;
}

public String getFone() {
return fone;
}

public void setFone(String fone) {
this.fone = fone;
```

```
}

public String getCpf_aluno() {
return cpf_aluno;
}

public void setCpf_aluno(String cpf_aluno) {
this.cpf_aluno = cpf_aluno;
}

public String getRg() {
return rg;
}

public void setRg(String rg) {
this.rg = rg;
}

public String getOrgaoexpedidor() {
return orgaoexpedidor;
}

public void setOrgaoexpedidor(String orgaoexpedidor) {
```

```
this.orgaoexpedidor = orgaoexpedidor;
}

public Date getDataexpedicao() {
return dataexpedicao;
}

public void setDataexpedicao(Date dataexpedicao) {
this.dataexpedicao = dataexpedicao;
}

public String getSexo() {
return sexo;
}

public void setSexo(String sexo) {
this.sexo = sexo;
}

public Date getDatanascimento() {
return datanascimento;
}
```

```
public void setDatanascimento(Date datanascimento) {
this.datanascimento = datanascimento;
}

public int getCertidaonascimento() {
return certidaonascimento;
}

public void setCertidaonascimento(int certidaonascimento) {
this.certidaonascimento = certidaonascimento;
}

public int getLivro() {
return livro;
}

public void setLivro(int livro) {
this.livro = livro;
}

public int getFolha() {
return folha;
}
```

```
public void setFolha(int folha) {
this.folha = folha;
}

public String getCartorio() {
return cartorio;
}

public void setCartorio(String cartorio) {
this.cartorio = cartorio;
}

public String getRua() {
return rua;
}

public void setRua(String rua) {
this.rua = rua;
}

public int getNumero_casa() {
return numero_casa;
```

```
}

public void setNumero_casa(int numero_casa) {
this.numero_casa = numero_casa;
}

public String getReferencia() {
return referencia;
}

public void setReferencia(String referencia) {
this.referencia = referencia;
}

public String getMunicipio_nascimento() {
return municipio_nascimento;
}

public void setMunicipio_nascimento(String municipio_nascimento) {
this.municipio_nascimento = municipio_nascimento;
}

public String getRaca() {
```

```
return raca;
}

public void setRaca(String raca) {
this.raca = raca;
}

public String getEscola() {
return escola;
}

public void setEscola(String escola) {
this.escola = escola;
}

public int getNumeromatricula() {
return numeromatricula;
}

public void setNumeromatricula(int numeromatricula) {
this.numeromatricula = numeromatricula;
}
```

```
public String getSerie() {
return serie;
}

public void setSerie(String serie) {
this.serie = serie;
}

public String getTurno() {
return turno;
}

public void setTurno(String turno) {
this.turno = turno;
}

public ArrayList<String> getList_alergias_selecionadas() {
return list_alergias_selecionadas;
}

public void setList_alergias_selecionadas(ArrayList<String>
list_alergias_selecionadas) {
this.list_alergias_selecionadas = list_alergias_selecionadas;
}
```

```
	public ArrayList<String> getList_problemas_saude_selecionados() {
	return list_problemas_saude_selecionados;
	}

	public void setList_problemas_saude_selecionados(ArrayList<String>
list_problemas_saude_selecionados) {
	this.list_problemas_saude_selecionados =
list_problemas_saude_selecionados;
	}

	public String getBolsafamilia() {
	return bolsafamilia;
	}

	public void setBolsafamilia(String bolsafamilia) {
	this.bolsafamilia = bolsafamilia;
	}

	public String getPai() {
	return pai;
	}

	public void setPai(String pai) {
```

```
this.pai = pai;
}

public String getMae() {
return mae;
}

public void setMae(String mae) {
this.mae = mae;
}

public String getSituacao() {
return situacao;
}

public void setSituacao(String situacao) {
this.situacao = situacao;
}

public List<String> getList_problemas() {
return ops.selecioneProblemaSaude();
}
```

```
public void setList_problemas(ArrayList<String> list_problemas) {
this.list_problemas = list_problemas;
}

public List<String> getList_cartorios() {
return ops.selecioneCartorios();
}

public void setList_cartorios(ArrayList<String> list_cartorios) {
this.list_cartorios = list_cartorios;
}

public List<String> getList_ruas() {
return ops.selecioneRuas();
}

public void setList_ruas(ArrayList<String> list_ruas) {
this.list_ruas = list_ruas;
}

public List<String> getList_municipio_nascimentos() {
return ops.selecioneCidades();
}
```

```
public void setList_municipio_nascimentos(ArrayList<String>
list_municipio_nascimentos) {
this.list_municipio_nascimentos = list_municipio_nascimentos;
}

public List<String> getList_escolas() {
return ops.selecioneEscolas();
}

public void setList_escolas(ArrayList<String> list_escolas) {
this.list_escolas = list_escolas;
}

public List<String> getList_alergias() {
return ops.selecioneAlergias();
}

public void setList_alergias(ArrayList<String> list_alergias) {
this.list_alergias = list_alergias;
}

public List<String> getList_pais() {
return ops.selecionePais();
```

```
}

public void setList_pais(ArrayList<String> list_pais) {
this.list_pais = list_pais;
}

public List<String> getList_maes() {
return ops.selecioneMaes();
}

public void setList_maes(ArrayList<String> list_maes) {
this.list_maes = list_maes;
}

public List<String> getList_irmaos() {
return ops.selecioneIrmaos();
}

public void setList_irmaos(ArrayList<String> list_irmaos) {
this.list_irmaos = list_irmaos;
}

public ArrayList<String> getList_irmaos_selecionados() {
```

```
        return list_irmaos_selecionados;
        }

        public void setList_irmaos_selecionados(ArrayList<String>
list_irmaos_selecionados) {
        this.list_irmaos_selecionados = list_irmaos_selecionados;
        }

        public String getCartorio_entity() {
        return cartorio_entity;
        }

        public void setCartorio_entity(String cartorio_entity) {
        this.list_cartorios.add(cartorio_entity);
        this.cartorio_entity = cartorio_entity;
        }

        public String getRua_entity() {
        return rua_entity;
        }

        public void setRua_entity(String rua_entity) {
        this.list_ruas.add(rua_entity);
        this.rua_entity = rua_entity;
```

```
}

public String getCidade_entity() {
return cidade_entity;
}

public void setCidade_entity(String cidade_entity) {
this.list_municipio_nascimentos.add(cidade_entity);
this.cidade_entity = cidade_entity;
}

public String getEscola_entity() {
return escola_entity;
}

public void setEscola_entity(String escola_entity) {
this.list_escolas.add(escola_entity);
this.escola_entity = escola_entity;
}

public String getAlergia_entity() {
return alergia_entity;
}
```

```
public void setAlergia_entity(String alergia_entity) {
this.list_alergias.add(alergia_entity);
this.alergia_entity = alergia_entity;
}

public String getPai_entity() {
return pai_entity;
}

public void setPai_entity(String pai_entity) {
this.pai_entity = pai_entity;
}

public String getMae_entity() {
return mae_entity;
}

public void setMae_entity(String mae_entity) {
this.list_maes.add(mae_entity);
this.mae_entity = mae_entity;
}
```

```
public String getIrmao_entity() {
return irmao_entity;
}

public void setIrmao_entity(String irmao_entity) {
this.irmao_entity = irmao_entity;
}

public String getProblema_entity() {
return problema_entity;
}

public void setProblema_entity(String problema_entity) {
this.list_problemas.add(problema_entity);
this.problema_entity = problema_entity;
}

public ArrayList<String> getList_sexo() {
return list_sexo;
}

public void setList_sexo(ArrayList<String> list_sexo) {
this.list_sexo = list_sexo;
```

```
}

public ArrayList<String> getList_racas() {
return list_racas;
}

public void setList_racas(ArrayList<String> list_racas) {
this.list_racas = list_racas;
}

public ArrayList<String> getList_series() {
return list_series;
}

public void setList_series(ArrayList<String> list_series) {
this.list_series = list_series;
}

public ArrayList<String> getList_turnos() {
return list_turnos;
}

public void setList_turnos(ArrayList<String> list_turnos) {
```

```
this.list_turnos = list_turnos;
}

public ArrayList<String> getList_bolsa_familias() {
return list_bolsa_familias;
}

public void setList_bolsa_familias(ArrayList<String> list_bolsa_familias) {
this.list_bolsa_familias = list_bolsa_familias;
}

public ArrayList<String> getList_situacoes() {
return list_situacoes;
}

public void setList_situacoes(ArrayList<String> list_situacoes) {
this.list_situacoes = list_situacoes;
}

public ArrayList<String> getList_orgaos_espedidor() {
return list_orgaos_espedidor;
}
```

```
public PessoaEntity getP() {
return p;
}

public void setP(PessoaEntity p) {
this.p = p;
}

public void setList_orgaos_espedidor(ArrayList<String> list_orgaos_espedidor) {
this.list_orgaos_espedidor = list_orgaos_espedidor;
}

public String cadastrarAluno(){
if(nome.equals("") || id > 1 || rg.equals("") || data_matricula == null || dataexpedicao == null || datanascimento == null
|| orgaoexpedidor == null || certidaonascimento <=0 || livro <= 0 || folha <=0 || cartorio == null || sexo == null
|| rua ==null || numero_casa <=0 || fone.equals("") || municipio_nascimento == null || raca ==null || situacao == null
|| list_alergias_selecionadas.size() <= 0 || list_problemas_saude_selecionados.size() <= 0
|| list_irmaos_selecionados.size() <= 0|| pai == null || mae == null || bolsafamilia == null || escola == null
|| numeromatricula <= 0 || serie == null || turno == null || cpf_aluno.equals("")){
```

```
        FacesContext ctx = FacesContext.getCurrentInstance();
        FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário inválido", "Usuário inválido");
        ctx.addMessage(null, msg);
        FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage("Preencha os Campos Corretamente!"));
        return "preencha os campos";
  }else{
        PessoaEntity pe = new PessoaEntity();
        pe.setId(id);
        pe.setNome(nome);
        pe.setCpf(cpf_aluno);
        pe.setData_nasc(data_matricula);
        if(geb.cadastrePessoa(pe)){
                this.p = geb.selecionePessoaAtualiza(nome);
        CartorioEntity c = ops.selecioneCartorioAtualiza(cartorio);
        EnderecoEntity e = fin.selecioneRuasPorNome(rua);
        EscolaEntity s = ops.selecioneEscolaAtualiza(escola);
        CidadeEntity ci = ops.selecioneCidadeAtualiza(municipio_nascimento);
        List<AlergiaEntity> l_a = ops.selecioneListAlergias(list_alergias_selecionadas);
        List<IrmaoEntity> l_i = ops.selecioneListaIrmaos(list_irmaos_selecionados);
        List<ProblemaSaudeEntity> p_s = ops.selecioneListaProblemas(list_problemas_saude_selecionados);
```

```
PaiEntity p = ops.selecionePaiAtualiza(pai);

MaeEntity m = ops.selecioneMaeAtualiza(mae);

AlunoEntity alu = new AlunoEntity();

alu.setId(id);

alu.setData_matricula(data_matricula);

alu.setFone_aluno(fone);

alu.setRg(rg);

alu.setOrgao_expedidor(orgaoexpedidor);

alu.setData_expedicao(dataexpedicao);

alu.setSexo(sexo);

alu.setNumero_certidao(certidaonascimento);

alu.setLivro_certidao(livro);

alu.setFolha_certidao(folha);

alu.setCartorio(c);

alu.setRua_aluno(e);

alu.setNumero_casa(numero_casa);

alu.setPonto_referencia(referencia);

alu.setCidade_nascimento(ci);

alu.setCor_raca(raca);

alu.setEscola(s);

alu.setMatricula_escola(numeromatricula);

alu.setSerie(serie);

alu.setTurno(turno);
```

```
alu.setAlergia(l_a);
alu.setProblema_saude(p_s);
alu.setBolsa_familia(bolsafamilia);
alu.setIrmaos(l_i);
alu.setPai(p);
alu.setMae(m);
alu.setPessoa(this.p);
alu.setSituacao(situacao);
        s.getList_aluno().add(alu);
        c.getList_aluno().add(alu);
        ci.getList_aluno().add(alu);
        p.getList_aluno().add(alu);
        m.getList_aluno().add(alu);
        e.getList_aluno().add(alu);
if(ops.cadastreAluno(alu)){
ops.atualizaEscola(s);
ops.atualizaCartorio(c);
ops.atualizaCidade(ci);
ops.atualizaPai(p);
ops.atualizaMae(m);
ops.atualizaEndereco(e);
for(int i=0; i< l_a.size(); i++){
AlergiaEntity a = l_a.get(i);
```

```
a.getList_aluno().add(alu);
ops.atualizaAlergia(a);
}
for(int i=0; i<l_i.size(); i++){
IrmaoEntity ir = l_i.get(i);
ir.getList_aluno().add(alu);
ops.atualizaIrmao(ir);
}
for(int i=0; i<p_s.size(); i++){
ProblemaSaudeEntity pr = p_s.get(i);
pr.getList_aluno().add(alu);
ops.atualizaProblema(pr);
}
FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new
FacesMessage("Inserido!"));
tum.setNome_aluno_entity(this.p.getNome());
this.apagaObjeto();
}
}else{
AlunoEntity alu = new AlunoEntity();
        alu = ops.selecioneAlunoAtualiza(nome);
        if(alu.getFone_aluno().equals("")){
        this.p = geb.selecionePessoaAtualiza(nome);
        if(this.p.getNome().equals("")){
```

```
FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new
FacesMessage("O CPPF está errado!"));
                    return nome+" :  CPF está errado!";
                }else{
                CartorioEntity c = ops.selecioneCartorioAtualiza(cartorio);
        EnderecoEntity e = fin.selecioneRuasPorNome(rua);
        EscolaEntity s = ops.selecioneEscolaAtualiza(escola);
        CidadeEntity ci = ops.selecioneCidadeAtualiza(municipio_nascimento);
        List<AlergiaEntity> l_a =
ops.selecioneListAlergias(list_alergias_selecionadas);
        List<IrmaoEntity> l_i = ops.selecioneListaIrmaos(list_irmaos_selecionados);
        List<ProblemaSaudeEntity> p_s =
ops.selecioneListaProblemas(list_problemas_saude_selecionados);
        PaiEntity p = ops.selecionePaiAtualiza(pai);
        MaeEntity m = ops.selecioneMaeAtualiza(mae);
        //AlunoEntity alu = new AlunoEntity();
        alu.setId(id);
        alu.setData_matricula(data_matricula);
        alu.setFone_aluno(fone);
        alu.setRg(rg);
        alu.setOrgao_expedidor(orgaoexpedidor);
        alu.setData_expedicao(dataexpedicao);
        alu.setSexo(sexo);
```

```
alu.setNumero_certidao(certidaonascimento);
alu.setLivro_certidao(livro);
alu.setFolha_certidao(folha);
alu.setCartorio(c);
alu.setRua_aluno(e);
alu.setNumero_casa(numero_casa);
alu.setPonto_referencia(referencia);
alu.setCidade_nascimento(ci);
alu.setCor_raca(raca);
alu.setEscola(s);
alu.setMatricula_escola(numeromatricula);
alu.setSerie(serie);
alu.setTurno(turno);
alu.setAlergia(l_a);
alu.setProblema_saude(p_s);
alu.setBolsa_familia(bolsafamilia);
alu.setIrmaos(l_i);
alu.setPai(p);
alu.setMae(m);
alu.setPessoa(this.p);
alu.setSituacao(situacao);
        s.getList_aluno().add(alu);
        c.getList_aluno().add(alu);
```

```
            ci.getList_aluno().add(alu);
            p.getList_aluno().add(alu);
            m.getList_aluno().add(alu);
            e.getList_aluno().add(alu);
if(ops.cadastreAluno(alu)){
ops.atualizaEscola(s);
ops.atualizaCartorio(c);
ops.atualizaCidade(ci);
ops.atualizaPai(p);
ops.atualizaMae(m);
ops.atualizaEndereco(e);
for(int i=0; i< l_a.size(); i++){
AlergiaEntity a = l_a.get(i);
a.getList_aluno().add(alu);
ops.atualizaAlergia(a);
}
for(int i=0; i<l_i.size(); i++){
IrmaoEntity ir = l_i.get(i);
ir.getList_aluno().add(alu);
ops.atualizaIrmao(ir);
}
for(int i=0; i<p_s.size(); i++){
ProblemaSaudeEntity pr = p_s.get(i);
```

```
pr.getList_aluno().add(alu);
ops.atualizaProblema(pr);
}
FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage("Inserido!"));
tum.setNome_aluno_entity(this.p.getNome());
this.apagaObjeto();
}
        }
}else{
        FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage(nome+" :  Já é Aluno!"));
        this.apagaObjeto();
        return nome+" :  Já é Aluno!";
        }
}
}
return "operacional?faces-redirect=true";
}

public void buscarAluno(){
if(nome.equals("")){
FacesContext ctx = FacesContext.getCurrentInstance();
```

```
FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário inválido", "Digite um Nome Válido");

ctx.addMessage(null, msg);

}else{

AlunoEntity a = new AlunoEntity();

a = ops.selecioneAlunoAtualiza(nome);

if(a.getFone_aluno().equals("")){

FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage("Digite o Nome Correto"));

}else{

this.p = a.getPessoa();

id = a.getId();

nome = this.p.getNome();

data_matricula = a.getData_matricula();

fone = a.getFone_aluno();

cpf_aluno = this.p.getCpf();

rg = a.getRg();

orgaoexpedidor = a.getOrgao_expedidor();

dataexpedicao = a.getData_expedicao();

sexo = a.getSexo();

datanascimento = this.p.getData_nasc();

certidaonascimento = a.getNumero_certidao();

livro = a.getLivro_certidao();
```

```
folha = a.getFolha_certidao();
cartorio = a.getCartorio().getNome_cartorio();
rua = a.getRua_aluno().getLogradouro();
numero_casa = a.getNumero_casa();
referencia = a.getPonto_referencia();
municipio_nascimento = a.getCidade_nascimento().getNome_cidade();
raca = a.getCor_raca();
escola = a.getEscola().getNome_escola();
numeromatricula = a.getMatricula_escola();
serie = a.getSerie();
turno = a.getTurno();
list_alergias_selecionadas = this.preencheNomeAlergia(a.getAlergia());
list_problemas_saude_selecionados = this.preencheNomeProblema(a.getProblema_saude());
bolsafamilia = a.getBolsa_familia();
list_irmaos_selecionados = this.preencheNomeIrmao(a.getIrmaos());
pai = a.getPai().getPessoa().getNome();
mae = a.getMae().getPessoa().getNome();
situacao = a.getSituacao();
FacesContext.getCurrentInstance().addMessage("formPermissoes: xxx", new FacesMessage("Encontrado!"));
}
}
}
```

```
public void atualizarAluno(){

if(nome.equals("") || id < 1 || rg.equals("") || data_matricula == null || dataexpedicao == null || datanascimento == null

|| orgaoexpedidor == null || certidaonascimento <=0 || livro <= 0 || folha <=0 || cartorio == null || sexo == null

|| rua ==null || numero_casa <=0 || fone.equals("") || municipio_nascimento == null || raca ==null || situacao == null

|| list_alergias_selecionadas.size() <= 0 || list_problemas_saude_selecionados.size() <= 0 || list_irmaos_selecionados.size() <= 0

|| pai == null || mae == null || bolsafamilia == null || escola == null || numeromatricula <= 0 || serie == null

|| turno == null || cpf_aluno.equals("")){

FacesContext ctx = FacesContext.getCurrentInstance();

FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário inválido", "Campo Nulo");

ctx.addMessage(null, msg);

// return "preencha os campos";

}else {

CartorioEntity c = ops.selecioneCartorioAtualiza(cartorio);

EnderecoEntity e = fin.selecioneRuasPorNome(rua);

EscolaEntity s = ops.selecioneEscolaAtualiza(escola);

CidadeEntity ci = ops.selecioneCidadeAtualiza(municipio_nascimento);
```

```
        List<AlergiaEntity> l_a = ops.selecioneListAlergias(list_alergias_selecionadas);
        List<IrmaoEntity> l_i = ops.selecioneListaIrmaos(list_irmaos_selecionados);
        List<ProblemaSaudeEntity> p_s = ops.selecioneListaProblemas(list_problemas_saude_selecionados);
        PaiEntity p = ops.selecionePaiAtualiza(pai);
        MaeEntity m = ops.selecioneMaeAtualiza(mae);
        AlunoEntity a = new AlunoEntity();
        a.setId(id);
        a.setData_matricula(data_matricula);
        a.setFone_aluno(fone);
        a.setRg(rg);
        a.setOrgao_expedidor(orgaoexpedidor);
        a.setData_expedicao(dataexpedicao);
        a.setSexo(sexo);
        a.setNumero_certidao(certidaonascimento);
        a.setLivro_certidao(livro);
        a.setFolha_certidao(folha);
        a.setCartorio(c);
        a.setRua_aluno(e);
        a.setNumero_casa(numero_casa);
        a.setPonto_referencia(referencia);
        a.setCidade_nascimento(ci);
        a.setCor_raca(raca);
```

```
a.setEscola(s);
a.setMatricula_escola(numeromatricula);
a.setSerie(serie);
a.setTurno(turno);
a.setAlergia(l_a);
a.setProblema_saude(p_s);
a.setBolsa_familia(bolsafamilia);
a.setIrmaos(l_i);
a.setPai(p);
a.setMae(m);
a.setSituacao(situacao);
a.setPessoa(this.p);
s.getList_aluno().add(a);
        c.getList_aluno().add(a);
        ci.getList_aluno().add(a);
        p.getList_aluno().add(a);
        m.getList_aluno().add(a);
        e.getList_aluno().add(a);
if(ops.atualizaAluno(a)){
        ops.atualizaEscola(s);
ops.atualizaCartorio(c);
ops.atualizaCidade(ci);
ops.atualizaPai(p);
```

```
ops.atualizaMae(m);
ops.atualizaEndereco(e);
for(int i=0; i< l_a.size(); i++){
AlergiaEntity ale = l_a.get(i);
ale.getList_aluno().add(a);
ops.atualizaAlergia(ale);
}
for(int i=0; i<l_i.size(); i++){
IrmaoEntity ir = l_i.get(i);
ir.getList_aluno().add(a);
ops.atualizaIrmao(ir);
}
for(int i=0; i<p_s.size(); i++){
ProblemaSaudeEntity pr = p_s.get(i);
pr.getList_aluno().add(a);
ops.atualizaProblema(pr);
}
        FacesContext.getCurrentInstance().addMessage("formPermissoes:
xxx", new FacesMessage("Atualizado!"));
        this.apagaObjeto();
}else{
        FacesContext.getCurrentInstance().addMessage("formPermissoes:
xxx", new FacesMessage("Não Atualizado!"));
}
```

```
}

}


public void excluirAluno(){

if(nome.equals("") || id < 1 || rg.equals("") || data_matricula == null || dataexpedicao == null || datanascimento == null

|| orgaoexpedidor == null || certidaonascimento <=0 || livro <= 0 || folha <=0 || cartorio == null || sexo == null

|| rua ==null || numero_casa <=0 || fone.equals("") || municipio_nascimento == null || raca ==null || situacao == null

|| list_alergias_selecionadas.size() <= 0 || list_problemas_saude_selecionados.size() <= 0 || list_irmaos_selecionados.size() <= 0

|| pai == null || mae == null || bolsafamilia == null || escola == null || numeromatricula <= 0 || serie == null

|| turno == null || cpf_aluno.equals("")){

FacesContext ctx = FacesContext.getCurrentInstance();

FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário inválido", "Campo Nulo");

ctx.addMessage(null, msg);

}else{

CartorioEntity c = ops.selecioneCartorioAtualiza(cartorio);

EnderecoEntity e = fin.selecioneRuasPorNome(rua);

EscolaEntity s = ops.selecioneEscolaAtualiza(escola);
```

```
        CidadeEntity ci = ops.selecioneCidadeAtualiza(municipio_nascimento);
        List<AlergiaEntity> l_a = ops.selecioneListAlergias(list_alergias_selecionadas);
        List<IrmaoEntity> l_i = ops.selecioneListaIrmaos(list_irmaos_selecionados);
        List<ProblemaSaudeEntity> p_s = ops.selecioneListaProblemas(list_problemas_saude_selecionados);
        PaiEntity p = ops.selecionePaiAtualiza(pai);
        MaeEntity m = ops.selecioneMaeAtualiza(mae);
                AlunoEntity a = new AlunoEntity();
                a.setId(id);
                a.setData_matricula(data_matricula);
                a.setFone_aluno(fone);
                a.setRg(rg);
                a.setOrgao_expedidor(orgaoexpedidor);
                a.setData_expedicao(dataexpedicao);
                a.setSexo(sexo);
                a.setNumero_certidao(certidaonascimento);
                a.setLivro_certidao(livro);
                a.setFolha_certidao(folha);
                a.setCartorio(c);
                a.setRua_aluno(e);
                a.setNumero_casa(numero_casa);
                a.setPonto_referencia(referencia);
                a.setCidade_nascimento(ci);
```

```
                a.setCor_raca(raca);
                a.setEscola(s);
                a.setMatricula_escola(numeromatricula);
                a.setSerie(serie);
                a.setTurno(turno);
                a.setAlergia(l_a);
                a.setProblema_saude(p_s);
                a.setBolsa_familia(bolsafamilia);
                a.setIrmaos(l_i);
                a.setPai(p);
                a.setMae(m);
                a.setSituacao(situacao);
                a.setPessoa(this.p);
                if(ops.removeAluno(a))
                FacesContext.getCurrentInstance().addMessage("formPermissoes:
xxx", new FacesMessage("Removido!"));
        else
                FacesContext.getCurrentInstance().addMessage("formPermissoes:
xxx", new FacesMessage("ID não válido!"));
        }
        this.apagaObjeto();
        }

        private ArrayList<String> preencheNomeIrmao(List<IrmaoEntity> m){
```

```
        ArrayList<String> al = new ArrayList<>();
        for(int i=0; i<m.size(); i++){
        String s = m.get(i).getPessoa().getNome();
        al.add(s);
        }
        return al;
        }

        private                                                    ArrayList<String>
preencheNomeProblema(List<ProblemaSaudeEntity> p){
        ArrayList<String> al = new ArrayList<>();
        for(int i=0; i<p.size(); i++){
        String s = p.get(i).getProblema_saude();
        al.add(s);
        }
        return al;
        }

        private ArrayList<String> preencheNomeAlergia(List<AlergiaEntity> a){
        ArrayList<String> al = new ArrayList<>();
        for(int i=0; i<a.size(); i++){
        String s = a.get(i).getNome_alergia();
        al.add(s);
        }
```

```
return al;

}


private void apagaObjeto(){

id = 0;

nome = "";

cpf_aluno = "";

rg ="";

data_matricula = null;

fone ="";

orgaoexpedidor ="";

dataexpedicao = null;

sexo = "";

datanascimento = null;

certidaonascimento = 0;

livro = 0;

folha = 0;

cartorio = "";

rua = "";

numero_casa = 0;

referencia = "";

municipio_nascimento ="";

raca = "";
```

```
        escola = "";
        numeromatricula = 0;
        serie = "";
        turno = "";
        list_alergias_selecionadas.clear();
        list_problemas_saude_selecionados.clear();
        bolsafamilia = "";
        list_irmaos_selecionados.clear();
        pai = "";
        mae = "";
        situacao ="";
        this.p = null;
        }

        public Date dataAtual(){

        return new Date();
        }
}
```

Classe AlunoEntity:

```
package entidades;

import java.io.Serializable;
```

```
import java.util.Date;
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;

/**
 *
 * @author petterson
 */
@Entity
public class AlunoEntity implements Serializable {

        private static final long serialVersionUID = 1L;

        @Id
```

```
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;

@Temporal(javax.persistence.TemporalType.DATE)
private Date data_matricula;
private String fone_aluno;
private String rg;
private String orgao_expedidor;
@Temporal(javax.persistence.TemporalType.DATE)
private Date data_expedicao;
private String sexo;
@OneToOne
private PessoaEntity pessoa;
private int numero_certidao;
private int livro_certidao;
private int folha_certidao;
@OneToOne(cascade=CascadeType.REMOVE)
private CartorioEntity cartorio;
@OneToOne(cascade=CascadeType.REMOVE)
private EnderecoEntity rua_aluno;
private int numero_casa;
private String ponto_referencia;
@OneToOne(cascade=CascadeType.REMOVE)
```

```
private CidadeEntity cidade_nascimento;

private String cor_raca;

@OneToOne(cascade=CascadeType.REMOVE)

private EscolaEntity escola;

private int matricula_escola;

private String serie;

private String turno;

@ManyToMany

private List<AlergiaEntity> alergia;

@ManyToMany

private List<ProblemaSaudeEntity> problema_saude;

@ManyToMany(fetch = FetchType.EAGER)

private List<IrmaoEntity> irmaos;

private String bolsa_familia;

@OneToOne(cascade=CascadeType.REMOVE)

private PaiEntity pai;

@OneToOne(cascade=CascadeType.REMOVE)

private MaeEntity mae;

private String situacao;



public AlunoEntity(){}
```

```
public Long getId() {
return id;
}

public void setId(Long id) {
this.id = id;
}

public Date getData_matricula() {
return data_matricula;
}

public void setData_matricula(Date data_matricula) {
this.data_matricula = data_matricula;
}

public String getFone_aluno() {
return fone_aluno;
}

public void setFone_aluno(String fone_aluno) {
this.fone_aluno = fone_aluno;
}
```

```
public String getRg() {
return rg;
}

public void setRg(String rg) {
this.rg = rg;
}

public String getOrgao_expedidor() {
return orgao_expedidor;
}

public void setOrgao_expedidor(String orgao_expedidor) {
this.orgao_expedidor = orgao_expedidor;
}

public Date getData_expedicao() {
return data_expedicao;
}
```

```
public void setData_expedicao(Date data_expedicao) {
this.data_expedicao = data_expedicao;
}

public String getSexo() {
return sexo;
}

public void setSexo(String sexo) {
this.sexo = sexo;
}

public int getNumero_certidao() {
return numero_certidao;
}

public void setNumero_certidao(int numero_certidao) {
this.numero_certidao = numero_certidao;
}

public int getLivro_certidao() {
return livro_certidao;
}
```

```
public void setLivro_certidao(int livro_certidao) {
this.livro_certidao = livro_certidao;
}

public int getFolha_certidao() {
return folha_certidao;
}

public void setFolha_certidao(int folha_certidao) {
this.folha_certidao = folha_certidao;
}

public int getNumero_casa() {
return numero_casa;
}

public void setNumero_casa(int numero_casa) {
this.numero_casa = numero_casa;
}

public String getPonto_referencia() {
return ponto_referencia;
```

```
}

public void setPonto_referencia(String ponto_referencia) {
this.ponto_referencia = ponto_referencia;
}

public String getCor_raca() {
return cor_raca;
}

public void setCor_raca(String cor_raca) {
this.cor_raca = cor_raca;
}

public int getMatricula_escola() {
return matricula_escola;
}

public void setMatricula_escola(int matricula_escola) {
this.matricula_escola = matricula_escola;
}

public String getSerie() {
```

```
return serie;

}


public void setSerie(String serie) {

this.serie = serie;

}


public String getTurno() {

return turno;

}


public void setTurno(String turno) {

this.turno = turno;

}


public String getBolsa_familia() {

return bolsa_familia;

}


public void setBolsa_familia(String bolsa_familia) {

this.bolsa_familia = bolsa_familia;

}
```

```
public String getSituacao() {
return situacao;
}

public void setSituacao(String situacao) {
this.situacao = situacao;
}

public PessoaEntity getPessoa() {
return pessoa;
}

public void setPessoa(PessoaEntity pessoa) {
this.pessoa = pessoa;
}

public CartorioEntity getCartorio() {
return cartorio;
}

public void setCartorio(CartorioEntity cartorio) {
this.cartorio = cartorio;
}
```

```
public EnderecoEntity getRua_aluno() {
return rua_aluno;
}

public void setRua_aluno(EnderecoEntity rua_aluno) {
this.rua_aluno = rua_aluno;
}

public CidadeEntity getCidade_nascimento() {
return cidade_nascimento;
}

public void setCidade_nascimento(CidadeEntity cidade_nascimento) {
this.cidade_nascimento = cidade_nascimento;
}

public EscolaEntity getEscola() {
return escola;
}

public void setEscola(EscolaEntity escola) {
this.escola = escola;
```

```
        }

        public List<AlergiaEntity> getAlergia() {
        return alergia;
        }

        public void setAlergia(List<AlergiaEntity> alergia) {
        this.alergia = alergia;
        }

        public List<ProblemaSaudeEntity> getProblema_saude() {
        return problema_saude;
        }

        public void setProblema_saude(List<ProblemaSaudeEntity> problema_saude)
{
        this.problema_saude = problema_saude;
        }

        public List<IrmaoEntity> getIrmaos() {
        return irmaos;
        }

        public void setIrmaos(List<IrmaoEntity> irmaos) {
```

```
this.irmaos = irmaos;
}

public PaiEntity getPai() {
return pai;
}

public void setPai(PaiEntity pai) {
this.pai = pai;
}

public MaeEntity getMae() {
return mae;
}

public void setMae(MaeEntity mae) {
this.mae = mae;
}

@Override
public int hashCode() {
int hash = 0;
hash += (id != null ? id.hashCode() : 0);
```

```
        return hash;
        }

        @Override
        public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof AlunoEntity)) {
        return false;
        }
        AlunoEntity other = (AlunoEntity) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
        }
        return true;
        }

        @Override
        public String toString() {
        return "entidades.AlunoEntity[ id=" + id + " ]";
        }

}
```

Classe relatório GerarBalanço:

```
package bean_gerenciveis;


import beans_sessao.FinanceiroSessionBean;
import beans_sessao.GerenteSessionBean;
import com.itextpdf.io.font.constants.StandardFonts;
import com.itextpdf.io.image.ImageData;
import com.itextpdf.io.image.ImageDataFactory;
import com.itextpdf.kernel.colors.ColorConstants;
import com.itextpdf.kernel.font.PdfFont;
import com.itextpdf.kernel.font.PdfFontFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.kernel.colors.DeviceGray;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.borders.Border;
import com.itextpdf.layout.element.Cell;
import com.itextpdf.layout.element.Image;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;
import com.itextpdf.layout.property.HorizontalAlignment;
import com.itextpdf.layout.property.TextAlignment;
import com.itextpdf.layout.property.UnitValue;
```

```
import entidades.BalancoEntity;

import entidades.EntradaEntity;

import java.awt.Color;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import javax.inject.Named;

import java.io.Serializable;

import java.net.MalformedURLException;

import java.nio.file.Files;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.stream.Collectors;

import javax.ejb.EJB;

import javax.faces.application.FacesMessage;

import javax.faces.context.FacesContext;

import javax.faces.view.ViewScoped;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServletResponse;
```

```
/**
 *
 * @author petterson
 */
@Named(value = "gerarBalanceteJSFManagedBean")
@ViewScoped
public class GerarBalanceteJSFManagedBean implements Serializable {

        @EJB
        FinanceiroSessionBean fin;
        @EJB
        GerenteSessionBean geb;
        Date data_inicial;
        Date data_final;
        double valor_percentual;
        String nome_total;
        String percentual_total;
        boolean imprimir;
        List<String> nome_descricoes;
        List<BalancoEntity> balancos;
        List<BalancoEntity> governof;
        List<BalancoEntity> governom;
```

```
List<BalancoEntity> governoe;

List<BalancoEntity> empresas;

List<BalancoEntity> eventos;

List<BalancoEntity> receitas;

List<BalancoEntity> institutos_fundacoes;

List<BalancoEntity> resultado;



public GerarBalanceteJSFManagedBean() {

this.nome_descricoes = new ArrayList<>();

this.balancos = new ArrayList<>();

this.governoe = new ArrayList<>();

this.governof = new ArrayList<>();

this.governom = new ArrayList<>();

this.empresas = new ArrayList<>();

this.eventos = new ArrayList<>();

this.receitas = new ArrayList<>();

this.institutos_fundacoes = new ArrayList<>();

this.resultado = new ArrayList<>();

this.imprimir = false;

}


public Date getData_inicial() {
```

```
return data_inicial;
}

public void setData_inicial(Date data_inicial) {
this.data_inicial = data_inicial;
}

public Date getData_final() {
return data_final;
}

public void setData_final(Date data_final) {
this.data_final = data_final;
}

public List<String> getNome_descricoes() {
return nome_descricoes;
}

public void setNome_descricoes(List<String> nome_descricoes) {
this.nome_descricoes = nome_descricoes;
}
```

```
public List<BalancoEntity> getBalancos() {
return balancos;
}

public void setBalancos(List<BalancoEntity> balancos) {
this.balancos = balancos;
}

public double getValor_percentual() {
return valor_percentual;
}

public void setValor_percentual(double valor_percentual) {
this.valor_percentual = valor_percentual;
}

public List<BalancoEntity> getGovernof() {
return governof;
}

public void setGovernof(List<BalancoEntity> governof) {
this.governof = governof;
}
```

```
public List<BalancoEntity> getGovernom() {
return governom;
}

public void setGovernom(List<BalancoEntity> governom) {
this.governom = governom;
}

public List<BalancoEntity> getGovernoe() {
return governoe;
}

public void setGovernoe(List<BalancoEntity> governoe) {
this.governoe = governoe;
}

public List<BalancoEntity> getEmpresas() {
return empresas;
}

public void setEmpresas(List<BalancoEntity> empresas) {
this.empresas = empresas;
```

```
}

public List<BalancoEntity> getEventos() {
return eventos;
}

public void setEventos(List<BalancoEntity> eventos) {
this.eventos = eventos;
}

public List<BalancoEntity> getReceitas() {
return receitas;
}

public void setReceitas(List<BalancoEntity> receitas) {
this.receitas = receitas;
}

public List<BalancoEntity> getInstitutos_fundacoes() {
return institutos_fundacoes;
}

public List<BalancoEntity> getResultado() {
```

```
return resultado;
}

public void setResultado(List<BalancoEntity> resultado) {
this.resultado = resultado;
}

public void setInstitutos_fundacoes(List<BalancoEntity> institutos_fundacoes)
{
this.institutos_fundacoes = institutos_fundacoes;
}

public String getNome_total() {
return "TOTAL";
}

public void setNome_total(String nome_total) {
this.nome_total = nome_total;
}

public String getPercentual_total() {
return "100%";
}
```

```
public void setPercentual_total(String percentual_total) {
this.percentual_total = percentual_total;
}

public boolean isImprimir() {
return imprimir;
}

public void setImprimir(boolean imprimir) {
this.imprimir = imprimir;
}

public void buscar(){
if(data_inicial == null || data_final == null){
FacesContext ctx = FacesContext.getCurrentInstance();
FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Selecione as Datas", "Selecione as Datas");
ctx.addMessage(null, msg);
//return "preencha os campos";
}else{
    this.imprimir = true;
    List<BalancoEntity> bl = new ArrayList<>();
```

```
List<String> nomes = fin.selecioneNomeDescricao(data_inicial,
data_final);
this.nome_descricoes =
nomes.stream().distinct().collect(Collectors.toList());
for(int i=0; i<this.nome_descricoes.size(); i++){
List<EntradaEntity> l = fin.selecioneBalancoEntrada(data_inicial,
data_final, this.nome_descricoes.get(i));
BalancoEntity b = new BalancoEntity();
b.setNome(this.nome_descricoes.get(i));
b.setValor(this.calculaValor(l));
b.setValor_sttring(this.convertDoubleValor(this.calculaValor(l)));
this.valor_percentual += b.getValor();
bl.add(b);
}
List<BalancoEntity> bala = new ArrayList<>();
BalancoEntity balanco = new BalancoEntity();
balanco.setNome(this.getNome_total());
balanco.setPercentual(this.getPercentual_total());
balanco.setValor(this.valor_percentual);

balanco.setValor_sttring(this.convertDoubleValor(this.valor_percentual));
this.resultado.add(balanco);
bala = this.calculaPercentual(bl, this.valor_percentual);
this.defineTabela(bala);
```

```
        }
        }

        public void imprimirLista() throws FileNotFoundException,
MalformedURLException, IOException{
        PdfDocument pdf = new PdfDocument(new
PdfWriter("/home/sueder/Documentos/balanco.pdf"));
        pdf.addNewPage();
        Document document = new Document(pdf);
        String imageFile = "/home/sueder/Imagens/Abdulbaha.jpg";
        ImageData data = ImageDataFactory.create(imageFile);
        Image img = new Image(data);
        img.setMaxHeight(60);
        img.setMaxWidth(60);
        Paragraph header = new Paragraph()
        .add(img)
        .add("                                Balanço Anual")
        .setMarginBottom(20)
        .setFont(PdfFontFactory.createFont(StandardFonts.HELVETICA))
        .setFontSize(14)
        .setFontColor(ColorConstants.RED);

        ///////////////GOVERNO MUNICIPAL//////////////////////////////
```

```
        float[] columnWidths = {100F, 100F, 100F};
        Table t0 = new Table(UnitValue.createPercentArray(columnWidths));
        t0.setMarginBottom(20);
        t0.setHorizontalAlignment(HorizontalAlignment.CENTER);
        PdfFont f = PdfFontFactory.createFont(StandardFonts.HELVETICA);
        Cell cell = new Cell(1, 3)
                .add(new Paragraph("Governo Municipal")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t0.addHeaderCell(cell);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
```

```
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
        };
        for (Cell hfCell : headerFooter) {
                if (i == 0) {
                t0.addHeaderCell(hfCell);
                } else {
                t0.addFooterCell(hfCell);
                }
        }
        }

        for (int j = 0; j < this.governom.size(); j++) {
        if(j==0){
                t0.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getNome()).setFontSize(10)));
                t0.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getValor_sttring()).setFontSize(10)));
                t0.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getPercentual()).setFontSize(10)));
        }else{
```

```
                t0.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getNome()).setFontSize(10)));
        t0.addCell(new      Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getValor_sttring()).setFontSize(10)));
        t0.addCell(new      Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governom.get(j).getPercentual()).setFontSize(10)));
        }
        }

        ///////////////GOVERNO ESTADUAL////////////////////////////////

        Table t1 = new Table(UnitValue.createPercentArray(columnWidths));
        t1.setMarginBottom(20);
        t1.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell1 = new Cell(1, 3)
                .add(new Paragraph("Governo Estadual")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
```

```
t1.addHeaderCell(cell1);
for (int i = 0; i < 1; i++) {
Cell[] headerFooter = new Cell[]{
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
};
for (Cell hfCell : headerFooter) {
if (i == 0) {
t1.addHeaderCell(hfCell);
} else {
t1.addFooterCell(hfCell);
}
}
}

for (int j = 0; j < this.governoe.size(); j++) {
if(j==0){
```

```
                t1.addCell(new                    Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getNome()).setFontSize(10)));

                t1.addCell(new                    Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getValor_sttring()).setFontSize(10)));

                t1.addCell(new                    Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getPercentual()).setFontSize(10)));

        }else{

                t1.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getNome()).setFontSize(10)));

                t1.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getValor_sttring()).setFontSize(10)));

                t1.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governoe.get(j).getPercentual()).setFontSize(10)));

        }

        }


        ///////////////GOVERNO FEDERAL//////////////////////////////


        Table t2 = new Table(UnitValue.createPercentArray(columnWidths));

        t2.setMarginBottom(20);

        t2.setHorizontalAlignment(HorizontalAlignment.CENTER);
```

```
        Cell cell2 = new Cell(1, 3)
                .add(new Paragraph("Governo Federal")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t2.addHeaderCell(cell2);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
        };
        for (Cell hfCell : headerFooter) {
                if (i == 0) {
                t2.addHeaderCell(hfCell);
```

```
                } else {
                t2.addFooterCell(hfCell);
                }
        }
        }

        for (int j = 0; j < this.governof.size(); j++) {
        if(j==0){
                t2.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getNome()).setFontSize(10)));
                t2.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getValor_sttring()).setFontSize(10)));
                t2.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getPercentual()).setFontSize(10)));
        }else{
                t2.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getNome()).setFontSize(10)));
                t2.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getValor_sttring()).setFontSize(10)));
                t2.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.governof.get(j).getPercentual()).setFontSize(10)));
```

```
        }
        }

        ////////////////INSTITUTOS E FUNDAÇÕES//////////////////////////////

        Table t3 = new Table(UnitValue.createPercentArray(columnWidths));
        t3.setMarginBottom(20);
        t3.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell3 = new Cell(1, 3)
                .add(new Paragraph("Institutos e Fundações")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t3.addHeaderCell(cell3);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
```

```
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
        };
        for (Cell hfCell : headerFooter) {
                if (i == 0) {
                t3.addHeaderCell(hfCell);
                } else {
                t3.addFooterCell(hfCell);
                }
        }
        }

        for (int j = 0; j < this.institutos_fundacoes.size(); j++) {
        if(j==0){
                t3.addCell(new                    Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getNome()).setFontSize(10)));
                t3.addCell(new                    Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getValor_sttring()).setFontSize(10)));
```

```
                t3.addCell(new                              Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getPercentual()).setFontSize(10)));
        }else{
                t3.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getNome()).setFontSize(10)));
                t3.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getValor_sttring()).setFontSize(10)));
                t3.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.institutos_fundacoes.get(j).getPercentual()).setFontSize(10)));
        }
        }

        ///////////////EMPRESAS PRIVADAS//////////////////////////////


        Table t4 = new Table(UnitValue.createPercentArray(columnWidths));
        t4.setMarginBottom(20);
        t4.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell4 = new Cell(1, 3)
                .add(new Paragraph("Empresas Privadas")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
```

```
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t4.addHeaderCell(cell4);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
        };
        for (Cell hfCell : headerFooter) {
                if (i == 0) {
                t4.addHeaderCell(hfCell);
                } else {
                t4.addFooterCell(hfCell);
                }
        }
```

```
        }


        for (int j = 0; j < this.empresas.size(); j++) {
        if(j==0){
                t4.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getNome()).setFontSize(10)));
                t4.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getValor_sttring()).setFontSize(10)));
                t4.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getPercentual()).setFontSize(10)));
        }else{
                t4.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getNome()).setFontSize(10)));
                t4.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getValor_sttring()).setFontSize(10)));
                t4.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.empresas.get(j).getPercentual()).setFontSize(10)));
        }
        }


        ///////////////RECEITAS FINANCEIRAS//////////////////////////////
```

```
        Table t5 = new Table(UnitValue.createPercentArray(columnWidths));
        t5.setMarginBottom(20);
        t5.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell5 = new Cell(1, 3)
                .add(new Paragraph("Receitas Financeiras")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t5.addHeaderCell(cell5);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
```

```
};
for (Cell hfCell : headerFooter) {
if (i == 0) {
t5.addHeaderCell(hfCell);
} else {
t5.addFooterCell(hfCell);
}
}
}

for (int j = 0; j < this.receitas.size(); j++) {
if(j==0){
t5.addCell(new Cell().setBackgroundColor(new DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new Paragraph(this.receitas.get(j).getNome()).setFontSize(10)));
t5.addCell(new Cell().setBackgroundColor(new DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new Paragraph(this.receitas.get(j).getValor_sttring()).setFontSize(10)));
t5.addCell(new Cell().setBackgroundColor(new DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new Paragraph(this.receitas.get(j).getPercentual()).setFontSize(10)));
}else{
t5.addCell(new Cell().setTextAlignment(TextAlignment.CENTER).add(new Paragraph(this.receitas.get(j).getNome()).setFontSize(10)));
```

```
                t5.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.receitas.get(j).getValor_sttring()).setFontSize(10)));
                t5.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.receitas.get(j).getPercentual()).setFontSize(10)));
        }
        }

        ///////////////BAZAR E EVENTOS////////////////////////////////

        Table t6 = new Table(UnitValue.createPercentArray(columnWidths));
        t6.setMarginBottom(20);
        t6.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell6 = new Cell(1, 3)
                .add(new Paragraph("Bazar e Eventos")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t6.addHeaderCell(cell6);
```

```
for (int i = 0; i < 1; i++) {
Cell[] headerFooter = new Cell[]{
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
};
for (Cell hfCell : headerFooter) {
if (i == 0) {
t6.addHeaderCell(hfCell);
} else {
t6.addFooterCell(hfCell);
}
}
}

for (int j = 0; j < this.eventos.size(); j++) {
if(j==0){
```

```
                t6.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getNome()).setFontSize(10)));

                t6.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getValor_sttring()).setFontSize(10)));

                t6.addCell(new                          Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getPercentual()).setFontSize(10)));

        }else{

                t6.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getNome()).setFontSize(10)));

                t6.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getValor_sttring()).setFontSize(10)));

                t6.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.eventos.get(j).getPercentual()).setFontSize(10)));

        }

        }


        ///////////////PESSOAS FÍSICAS//////////////////////////////


        Table t7 = new Table(UnitValue.createPercentArray(columnWidths));

        t7.setMarginBottom(20);

        t7.setHorizontalAlignment(HorizontalAlignment.CENTER);
```

```
        Cell cell7 = new Cell(1, 3)
                .add(new Paragraph("Pessoas Físicas")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t7.addHeaderCell(cell7);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valores")),
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual")),
        };
        for (Cell hfCell : headerFooter) {
                if (i == 0) {
                t7.addHeaderCell(hfCell);
```

```
                } else {
                t7.addFooterCell(hfCell);
                }
        }
        }

        for (int j = 0; j < this.balancos.size(); j++) {
        if(j==0){
                t7.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getNome()).setFontSize(10)));
                t7.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getValor_sttring()).setFontSize(10)));
                t7.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getPercentual()).setFontSize(10)));
        }else{
                t7.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getNome()).setFontSize(10)));
                t7.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getValor_sttring()).setFontSize(10)));
                t7.addCell(new
Cell().setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.balancos.get(j).getPercentual()).setFontSize(10)));
```

```
        }
        }

        ///////////////TOTAL////////////////////////////////

        Table t8 = new Table(UnitValue.createPercentArray(columnWidths));
        t8.setMarginBottom(20);
        t8.setHorizontalAlignment(HorizontalAlignment.CENTER);
        Cell cell8 = new Cell(1, 3)
                .add(new Paragraph("Total")
                .setBackgroundColor(new DeviceGray(0.25f))
                .setBorder(Border.NO_BORDER))
                .setFont(f)
                .setFontSize(13)
                .setFontColor(DeviceGray.WHITE)
                .setBackgroundColor(DeviceGray.BLACK)
                .setTextAlignment(TextAlignment.CENTER);
        t8.addHeaderCell(cell8);
        for (int i = 0; i < 1; i++) {
        Cell[] headerFooter = new Cell[]{
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Nome")),
```

```
                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Valor Total")),

                new
Cell().setTextAlignment(TextAlignment.CENTER).setBackgroundColor(new
DeviceGray(0.75f)).add(new Paragraph("Percentual Completo")),

        };

        for (Cell hfCell : headerFooter) {

                if (i == 0) {

                t8.addHeaderCell(hfCell);

                } else {

                t8.addFooterCell(hfCell);

                }

        }

        }


        for (int j = 0; j < this.resultado.size(); j++) {

                t8.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.resultado.get(j).getNome()).setFontSize(10)));

                t8.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.resultado.get(j).getValor_sttring()).setFontSize(10)));

                t8.addCell(new                  Cell().setBackgroundColor(new
DeviceGray(0.90f)).setTextAlignment(TextAlignment.CENTER).add(new
Paragraph(this.resultado.get(j).getPercentual()).setFontSize(10)));
```

```
        }


        //document.add(img);
        document.add(header);
        document.add(t0);
        document.add(t1);
        document.add(t2);
        document.add(t3);
        document.add(t4);
        document.add(t5);
        document.add(t6);
        document.add(t7);
        document.add(t8);
        document.close();

        ////////////////////////////////ENVIA PARA O BROWSER//////////////////////

        HttpServletResponse response = (HttpServletResponse) FacesContext.getCurrentInstance().getExternalContext().getResponse();
        File arquivo = new File("/home/sueder/Documentos/balanco.pdf");
        int tamanho = (int) arquivo.length();
        response.setContentType("application/pdf"); // tipo do conteúdo na resposta
        response.setContentLength(tamanho); // opcional. ajuda na barra de progresso
```

```
        response.setHeader("Content-Disposition",                    "attachment;
filename=balanco.pdf");
        ServletOutputStream output = response.getOutputStream();
        Files.copy(arquivo.toPath(), output); // escreve bytes no fluxo de saída
        FacesContext.getCurrentInstance().responseComplete();
        }

        private void defineTabela(List<BalancoEntity> b){
        List<String> nomes = new ArrayList<>();
        for(int i=0; i<b.size(); i++){
                BalancoEntity bb = b.get(i);
                if(bb.getNome().equals("Bingo")                                    ||
bb.getNome().equals("Bazar/Brecho")){
                this.eventos.add(bb);
                nomes.add("EVENTOS");
                }else{
                String n = geb.getTipoInstituicao(bb.getNome());
                if(n.equals("nao")){
                        this.balancos.add(bb);
                        nomes.add("PESSOAS FISICAS");
                }else if(n.equals("GOVERNO FEDERAL")){
                        this.governof.add(bb);
                        nomes.add("GOVERNO FEDERAL");
                }else if(n.equals("GOVERNO MUNICIPAL")){
```

```
                this.governom.add(bb);
                nomes.add("GOVERNO MUNICIPAL");
            }else if(n.equals("GOVERNO ESTADUAL")){
                this.governoe.add(bb);
                nomes.add("GOVERNO ESTADUAL");
            }else if(n.equals("INSTITUTO FUNDAÇÕES")){
                this.institutos_fundacoes.add(bb);
                nomes.add("INSTITUTO FUNDAÇÕES");
            }else if(n.equals("EMPRESAS PRIVADAS")){
                this.empresas.add(bb);
                nomes.add("EMPRESAS PRIVADAS");
            }else if(n.equals("RECEITAS FINANCEIRAS")){
                this.receitas.add(bb);
                nomes.add("RECEITAS FINANCEIRAS");
            }
            }
        }
        List<BalancoEntity> balan = new ArrayList<>();
        this.nome_descricoes = nomes.stream().distinct().collect(Collectors.toList());
        for(int i=0; i< this.nome_descricoes.size(); i++){
            if(this.nome_descricoes.get(i).equals("EVENTOS")){
            BalancoEntity e = this.defineCabecario(this.eventos,
this.nome_descricoes.get(i));
            balan.add(e);
```

```
                balan.addAll(this.eventos);
                this.eventos.clear();
                this.eventos.addAll(balan);
                balan.clear();
                } else if(this.nome_descricoes.get(i).equals("PESSOAS FISICAS")){
                BalancoEntity e = this.defineCabecario(this.balancos,
this.nome_descricoes.get(i));
                balan.add(e);
                balan.addAll(this.balancos);
                this.balancos.clear();
                this.balancos.addAll(balan);
                balan.clear();
                } else if(this.nome_descricoes.get(i).equals("GOVERNO FEDERAL")){
                BalancoEntity e = this.defineCabecario(this.governof,
this.nome_descricoes.get(i));
                balan.add(e);
                balan.addAll(this.governof);
                this.governof.clear();
                this.governof.addAll(balan);
                balan.clear();
                } else if(this.nome_descricoes.get(i).equals("GOVERNO ESTADUAL")){
                BalancoEntity e = this.defineCabecario(this.governoe,
this.nome_descricoes.get(i));
                balan.add(e);
```

```
balan.addAll(this.governoe);
this.governoe.clear();
this.governoe.addAll(balan);
balan.clear();
} else if(this.nome_descricoes.get(i).equals("GOVERNO MUNICIPAL")){
BalancoEntity e = this.defineCabecario(this.governom, this.nome_descricoes.get(i));
balan.add(e);
balan.addAll(this.governom);
this.governom.clear();
this.governom.addAll(balan);
balan.clear();
} else if(this.nome_descricoes.get(i).equals("EMPRESAS PRIVADAS")){
BalancoEntity e = this.defineCabecario(this.empresas, this.nome_descricoes.get(i));
balan.add(e);
balan.addAll(this.empresas);
this.empresas.clear();
this.empresas.addAll(balan);
balan.clear();
} else if(this.nome_descricoes.get(i).equals("INSTITUTO FUNDAÇÕES")){
```

```
                BalancoEntity e = this.defineCabecario(this.institutos_fundacoes,
this.nome_descricoes.get(i));
                balan.add(e);
                balan.addAll(this.institutos_fundacoes);
                this.institutos_fundacoes.clear();
                this.institutos_fundacoes.addAll(balan);
                balan.clear();
                } else if(this.nome_descricoes.get(i).equals("RECEITAS
FINANCEIRAS")){
                BalancoEntity e = this.defineCabecario(this.receitas,
this.nome_descricoes.get(i));
                balan.add(e);
                balan.addAll(this.receitas);
                this.receitas.clear();
                this.receitas.addAll(balan);
                balan.clear();
                }
        }
        }

        private BalancoEntity defineCabecario(List<BalancoEntity> b, String nome){
        double valor =0;
        String percentualgeral="";
        BalancoEntity ba = new BalancoEntity();
```

```
        ba.setNome(nome);
        for(int i=0; i<b.size(); i++){
                valor += b.get(i).getValor();
                double v = (valor * 100)/this.valor_percentual;
                percentualgeral = this.convertDouble(v);
        }
        ba.setPercentual(percentualgeral);
        ba.setValor_sttring(this.convertDoubleValor(valor));
        return ba;
        }

        private double calculaValor(List<EntradaEntity> entradas){
        double valor=0;
        for(int i=0; i<entradas.size(); i++){
                EntradaEntity e = entradas.get(i);
                valor += e.getValor_entrada();
        }
        return valor;
        }

        private List<BalancoEntity> calculaPercentual(List<BalancoEntity> bas,
double valorTotal){
        List<BalancoEntity> b = new ArrayList<>();
        for(int i=0; i<bas.size(); i++){
```

```
            BalancoEntity ba = bas.get(i);
            double v = (ba.getValor() * 100)/valorTotal;
            ba.setPercentual(this.convertDouble(v));
            b.add(ba);
    }
    return b;
    }

    private String convertDouble(double va){
    String p = " %";
    String v = String.valueOf(va);
    //System.out.println(v);
    if(v.length() < 4){
            v += v +00;
    }
    String s = v.substring(0, 4);
    String f = s+p;
    return f;
    }

    private String convertDoubleValor(double va){
    String p = "R$ ";
    String v = String.valueOf(va);
```

```
        String s = "0";
        String f = p+v+s;
        return f;
        }
}
```

Abaixo os beans de sessão: Financeiro, Gerente, Operacional:

FinanceiroSessionBean:

```
package beans_sessao;

import entidades.AgenciaEntity;
import entidades.BalanceteEntity;
import entidades.BalancoEntity;
import entidades.ChequeEntity;
import entidades.CidadeEntity;
import entidades.ContaEntity;
import entidades.DescricaoEntity;
import entidades.DescricaoSaidaEntity;
import entidades.DistritoEntity;
import entidades.EnderecoEntity;
import entidades.EntradaEntity;
import entidades.FornecedorEntity;
import entidades.PrestadorFisicoEntity;
import entidades.PrestadoraServicoEntity;
```

```
import entidades.SaidaEntity;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import javax.ejb.EJBException;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import javax.persistence.TemporalType;

/**
 *
 * @author petterson
 */
@Stateless
@LocalBean
public class FinanceiroSessionBean {

        @PersistenceContext(unitName="jami-ejbPU")
```

```
EntityManager em ;



////////            PRESTADORA



public boolean cadastrePrestadoraServico(PrestadoraServicoEntity prestadora){
boolean ret=true;
Query query = em.createQuery("Select p.nome_prestadora FROM PrestadoraServicoEntity p");
List<String> nome_prestadoras = query.getResultList();
for(int j=0; j< nome_prestadoras.size(); j++){
String sa = nome_prestadoras.get(j);
if(sa.equals(prestadora.getNome_prestadora()))
        ret = false;
}
if(ret == true)
        em.persist(prestadora);
return ret;
}


public List<String> selecionePrestadoraServicos() {
```

```
Query query = em.createQuery("Select p.nome_prestadora FROM PrestadoraServicoEntity p");
List<String> ufs = query.getResultList();
return ufs;
}

public PrestadoraServicoEntity selecionePrestadoraServicoAtualiza(String s) {
PrestadoraServicoEntity p = new PrestadoraServicoEntity();
Query query = em.createQuery("Select p FROM PrestadoraServicoEntity p WHERE p.nome_prestadora = :s");
query.setParameter("s", s);
try{
p = (PrestadoraServicoEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setNome_prestadora("");
}
p.getId();
p.getCnpj_prestadora();
p.getNome_prestadora();
p.getFone_prestadora();
p.getEmail_prestadora();
return p;
}
```

```
public boolean removePrestadoraServico(PrestadoraServicoEntity p){
boolean ret=true;
PrestadoraServicoEntity pr = new PrestadoraServicoEntity();
try{
pr = em.find(PrestadoraServicoEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaPrestadoraServico(PrestadoraServicoEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
return ret;
}

////////            PRESTADOR FISICO

public boolean cadastrePrestadorFisico(PrestadorFisicoEntity prestador){
        em.persist(prestador);
return true;
}

public List<String> selecionePrestadoraFisicos() {
Query query = em.createQuery("Select a.nome FROM PrestadorFisicoEntity p, p.pessoa a");
List<String> ufs = query.getResultList();
return ufs;
}

public PrestadorFisicoEntity selecionePrestadorFisicoAtualiza(String s) {
PrestadorFisicoEntity p = new PrestadorFisicoEntity();
```

```
Query query = em.createQuery("Select p FROM PrestadorFisicoEntity p, p.pessoa a WHERE a.nome = :s");
query.setParameter("s", s);
try{
p = (PrestadorFisicoEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setFone_fisico("");
}
p.getId();
p.getPessoa();
p.getFone_fisico();
p.getEmail_fisico();
return p;
}

public boolean removePrestadorFisico(PrestadorFisicoEntity p){
boolean ret=true;
PrestadorFisicoEntity pr = new PrestadorFisicoEntity();
try{
pr = em.find(PrestadorFisicoEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(pr.getId() == null)
        return false;
```

```
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaPrestadorFisico(PrestadorFisicoEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

////////            FORNECEDOR

public boolean cadastreFornecedor(FornecedorEntity fornecedor){
```

```
boolean ret=true;
Query query = em.createQuery("Select f.nome_fornecedor FROM FornecedorEntity f");
List<String> nome_fornecedores = query.getResultList();
for(int j=0; j< nome_fornecedores.size(); j++){
String sa = nome_fornecedores.get(j);
if(sa.equals(fornecedor.getNome_fornecedor()))
        ret = false;
}
if(ret == true)
        em.persist(fornecedor);
return ret;
}

public List<String> selecioneFornecedores() {
Query query = em.createQuery("Select f.nome_fornecedor FROM FornecedorEntity f");
List<String> ufs = query.getResultList();
return ufs;
}

public FornecedorEntity selecioneFornecedorAtualiza(String s) {
FornecedorEntity f = new FornecedorEntity();
```

```
        Query query = em.createQuery("Select f FROM FornecedorEntity f WHERE
f.nome_fornecedor = :s");
        query.setParameter("s", s);
        try{
        f = (FornecedorEntity) query.getSingleResult();
        }catch(NoResultException e){
                f.setNome_fornecedor("");
        }
        f.getId();
        f.getCnpj_forneceedor();
        f.getNome_fornecedor();
        f.getFone_fornecedor();
        f.getEmail_fornecedor();
        f.getData_cadastro();
        return f;
        }

        public boolean removeFornecedor(FornecedorEntity f){
        boolean ret=true;
        FornecedorEntity fo = new FornecedorEntity();
        try{
        fo = em.find(FornecedorEntity.class, f.getId());
        }catch(java.lang.IllegalArgumentException e){
        e.getMessage();
```

```
        if(fo.getId() == null)
        return false;
}
try{
em.remove(fo);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;

}

public boolean atualizaFornecedor(FornecedorEntity f){
boolean ret=true;
try{
em.merge(f);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
////////            AGÊNCIA


        public boolean cadastreAgencia(AgenciaEntity agencia){
        boolean ret=true;
        Query query = em.createQuery("Select a.nome_agencia FROM AgenciaEntity
a");
        List<String> nome_agencias = query.getResultList();
        for(int j=0; j< nome_agencias.size(); j++){
        String sa = nome_agencias.get(j);
        if(sa.equals(agencia.getNome_agencia()))
                ret = false;
        }
        if(ret == true)
                em.persist(agencia);
        return ret;
        }
```

```
public List<String> selecioneAgencias() {
Query query = em.createQuery("Select a.nome_agencia FROM AgenciaEntity a");
List<String> ufs = query.getResultList();
return ufs;
}

public List<String> selecioneNumeroAgencias() {
Query query = em.createQuery("Select a.numero_agencia FROM AgenciaEntity a");
List<String> ufs = query.getResultList();
return ufs;
}

public AgenciaEntity selecioneAgenciaAtualiza(String s) {
AgenciaEntity a = new AgenciaEntity();
Query query = em.createQuery("Select a FROM AgenciaEntity a WHERE a.nome_agencia = :s");
query.setParameter("s", s);
try{
a = (AgenciaEntity) query.getSingleResult();
}catch(NoResultException e){
        a.setNome_agencia("");
```

```
        }
        a.getId();
        a.getNumero_agencia();
        a.getNome_agencia();
        a.getFone_agencia();
        a.getEmail_agencia();
        a.getRua_agencia();
        a.getNumero_rua_agencia();
        return a;
        }

        public AgenciaEntity selecioneAgenciaConta(String s) {
        AgenciaEntity a = new AgenciaEntity();
        Query query = em.createQuery("Select a FROM AgenciaEntity a WHERE
a.numero_agencia = :s");
        query.setParameter("s", s);
        try{
        a = (AgenciaEntity) query.getSingleResult();
        }catch(NoResultException e){
                a.setNome_agencia("");
        }
        a.getId();
        a.getNumero_agencia();
        a.getNome_agencia();
```

```
a.getFone_agencia();

a.getEmail_agencia();

a.getRua_agencia();

a.getNumero_rua_agencia();

return a;

}


public boolean removeAgencia(AgenciaEntity a){

boolean ret=true;

AgenciaEntity ag = new AgenciaEntity();

try{

ag = em.find(AgenciaEntity.class, a.getId());

}catch(java.lang.IllegalArgumentException e){

e.getMessage();

        if(ag.getId() == null)

        return false;

}

try{

em.remove(ag);

}catch(java.lang.IllegalArgumentException e){

return false;

}

return ret;
```

```
}


public boolean atualizaAgencia(AgenciaEntity a){
boolean ret=true;
try{
em.merge(a);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
////////            CONTA


public boolean cadastreConta(ContaEntity conta){
boolean ret=true;
Query query = em.createQuery("Select c.nome_conta FROM ContaEntity c");
List<String> nome_contas = query.getResultList();
for(int j=0; j< nome_contas.size(); j++){
String sa = nome_contas.get(j);
if(sa.equals(conta.getNome_conta()))
        ret = false;
}
if(ret == true)
        em.persist(conta);
return ret;
}


public List<String> selecioneContas() {
Query query = em.createQuery("Select c.nome_conta FROM ContaEntity c");
List<String> numero_contas = query.getResultList();
return numero_contas;
```

```
        }

        public List<String> selecioneNumeroContas() {
        Query query = em.createQuery("Select c.numero_conta FROM ContaEntity
c");
        List<String> numero_contas = query.getResultList();
        return numero_contas;
        }

        public String selecioneNomeConta(ContaEntity c) {
        String n = c.getNome_conta();
        String numero="";
        Query query = em.createQuery("Select c.nome_conta FROM ContaEntity c
WHERE c.nome_conta =:n");
        query.setParameter("n", n);
        //String numero = (String) query.getSingleResult();
        try{
                numero = (String) query.getSingleResult();
        }catch(NoResultException e){
                c.setNome_conta("");
        }
        return numero;
        }
```

```
	public ContaEntity selecioneContaId(long id) {
	ContaEntity c = new ContaEntity();
	Query query = em.createQuery("Select c FROM ContaEntity c WHERE c.id =
:id");
	query.setParameter("id", id);
	try{
	c = (ContaEntity) query.getSingleResult();
	}catch(NoResultException e){
		c.setNome_conta("");
	}
	c.getId();
	c.getNumero_conta();
	c.getNome_conta();
	c.getAgencia();
	c.getTitular_conta();
	c.getData_criacao();
	c.getSaldo();
	c.getBalancetes();
	return c;
	}

	public ContaEntity selecioneContaAtualiza(String s) {
	ContaEntity c = new ContaEntity();
```

```
Query query = em.createQuery("Select c FROM ContaEntity c WHERE c.numero_conta = :s");
query.setParameter("s", s);
try{
c = (ContaEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNome_conta("");
}
c.getId();
c.getNumero_conta();
c.getNome_conta();
c.getAgencia();
c.getTitular_conta();
c.getData_criacao();
c.getSaldo();
c.getBalancetes();
return c;
}

public boolean removeConta(ContaEntity c){
boolean ret=true;
ContaEntity co = new ContaEntity();
try{
co = em.find(ContaEntity.class,c.getId());
```

```
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(co.getId() == null)
        return false;
}
try{
em.remove(co);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;

}

public boolean atualizaConta(ContaEntity c){
boolean ret=true;
try{
em.merge(c);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
public boolean atualizaContaValor(ContaEntity c){
boolean ret=true;
ContaEntity co = new ContaEntity();
String n =c.getNumero_conta();
long l=0;
Query quer = em.createQuery("Select c.id FROM ContaEntity c WHERE c.numero_conta = :n");
quer.setParameter("n", n);
try{
l =  (long) quer.getSingleResult();
}catch(NoResultException e){
        return false;
}
try{
co = em.find(ContaEntity.class, l);
co.getId();
co.setNumero_conta(c.getNumero_conta());
co.getNome_conta();
co.getAgencia();
co.getTitular_conta();
co.getData_criacao();
co.setSaldo(c.getSaldo());
co.getBalancetes();
```

```
}catch(NoResultException e){
        return false;
}
try{
em.merge(co);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

////////        BALANCETE


public boolean cadastreBalancete(BalanceteEntity balancete){
boolean ret=true;
Query query = em.createQuery("Select b.descricao_balancete FROM BalanceteEntity b");
List<String> nome_balancetes = query.getResultList();
for(int j=0; j< nome_balancetes.size(); j++){
String sa = nome_balancetes.get(j);
if(sa.equals(balancete.getDescricao_balancete()))
        ret = false;
}
```

```
if(ret == true)
        em.persist(balancete);
return ret;
}

public List<String> selecioneNomeBalancetes() {
Query query = em.createQuery("Select b.descricao_balancete FROM BalanceteEntity b");
List<String> nome_balancetes = query.getResultList();
return nome_balancetes;
}

public List<BalanceteEntity> selecioneBalancetes() {
Query query = em.createQuery("Select b FROM BalanceteEntity b");
List<BalanceteEntity> nome_balancetes = query.getResultList();
return nome_balancetes;
}

public BalanceteEntity selecioneBalanceteAtualiza(String s) {
BalanceteEntity b = new BalanceteEntity();
Query query = em.createQuery("Select b FROM BalanceteEntity b WHERE b.descricao_balancete = :s");
query.setParameter("s", s);
try{
```

```
b = (BalanceteEntity) query.getSingleResult();
}catch(NoResultException e){
        b.setDescricao_balancete("");
}
b.getId();
b.getNumero_conta();
b.getData_inicial();
b.getData_final();
b.getDescricao_balancete();
b.getSaldo();
return b;
}

public boolean removeBalancete(BalanceteEntity b){
boolean ret=true;
BalanceteEntity ba = new BalanceteEntity();
String nc = b.getNumero_conta();
ContaEntity c = new ContaEntity();
Query query = em.createQuery("Select c FROM ContaEntity c WHERE c.numero_conta = :nc");
query.setParameter("nc", nc);
try{
c = (ContaEntity) query.getSingleResult();
if(c.getBalancetes().size() > 1){
```

```
c.getBalancetes().remove(b);
em.merge(c);
try{
        ba = em.find(BalanceteEntity.class, b.getId());
}catch(java.lang.IllegalArgumentException e){
        return false;
}
try{
        em.remove(ba);
}catch(java.lang.IllegalArgumentException e){
return false;
}
}else{
        c.getBalancetes().remove(b);
        em.merge(c);
        try{
        ba = em.find(BalanceteEntity.class, b.getId());
}catch(java.lang.IllegalArgumentException e){
        return false;
}
try{
        em.remove(ba);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
}
}catch(java.lang.IllegalArgumentException e){
        return false;
}

return ret;
}

public boolean atualizaBalancete(BalanceteEntity b){
boolean ret=true;
try{
em.merge(b);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public long buscaId(String balancete){
long l=0;
Query quer = em.createQuery("Select b.id FROM BalanceteEntity b WHERE
b.descricao_balancete = :balancete");
```

```
        quer.setParameter("balancete", balancete);
        try{
        l =  (long) quer.getSingleResult();
        }catch(NoResultException e){
                return 0;
        }
        return l;
}
        ////////          CHEQUE


        public boolean cadastreCheque(ChequeEntity cheque){
        boolean ret=true;
        Query query = em.createQuery("Select c.numero_cheque FROM
ChequeEntity c");
        List<String> nome_cheques = query.getResultList();
        for(int j=0; j< nome_cheques.size(); j++){
        String sa = nome_cheques.get(j);
        if(sa.equals(cheque.getNumero_cheque()))
                ret = false;
        }
        if(ret == true)
                em.persist(cheque);
        return ret;
```

```
}

public List<String> selecioneCheques() {
Query query = em.createQuery("Select c.numero_cheque FROM ChequeEntity c");
List<String> ufs = query.getResultList();
return ufs;
}

public ChequeEntity selecioneChequeAtualiza(String s) {
ChequeEntity c = new ChequeEntity();
Query query = em.createQuery("Select c FROM ChequeEntity c WHERE c.numero_cheque = :s");
query.setParameter("s", s);
try{
c = (ChequeEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNumero_cheque("");
}
c.getId();
c.getNumero_cheque();
return c;
}
```

```
public boolean removeCheque(ChequeEntity c){
boolean ret=true;
ChequeEntity ce = new ChequeEntity();
try{
ce = em.find(ChequeEntity.class, c.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ce.getId() == null)
        return false;
}
try{
em.remove(ce);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaCheque(ChequeEntity c){
boolean ret=true;
try{
em.merge(c);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
return ret;
}

////////            DESCRIÇAO


public boolean cadastreDescricao(DescricaoEntity descricao){
boolean ret=true;
Query query = em.createQuery("Select d.descricao FROM DescricaoEntity
d");
List<String> nome_descricoes = query.getResultList();
for(int j=0; j< nome_descricoes.size(); j++){
String sa = nome_descricoes.get(j);
if(sa.equals(descricao.getDescricao()))
        ret = false;
}
if(ret == true)
        em.persist(descricao);
return ret;
}

public List<String> selecioneDescricoes() {
```

```
Query query = em.createQuery("Select d.descricao FROM DescricaoEntity
d");

List<String> ufs = query.getResultList();

Query query2 = em.createQuery("Select i.nome_instituicao FROM
InstituicaoEntity i");

List<String> ufs2 = query2.getResultList();

Query query3 = em.createQuery("Select p.nome FROM ApoiadorEntity a,
a.pessoa p");

List<String> ufs3 = query3.getResultList();

ufs.addAll(ufs2);

ufs.addAll(ufs3);

return ufs;

}


public DescricaoEntity selecioneDescricaoAtualiza(String s) {

DescricaoEntity d = new DescricaoEntity();

Query query = em.createQuery("Select d FROM DescricaoEntity d WHERE
d.descricao = :s");

query.setParameter("s", s);

try{

d = (DescricaoEntity) query.getSingleResult();

}catch(NoResultException e){

        d.setDescricao("");

}
```

```
d.getId();
d.getDescricao();
return d;
}

public boolean removeDescricao(DescricaoEntity d){
boolean ret=true;
DescricaoEntity de = new DescricaoEntity();
try{
de = em.find(DescricaoEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(de.getId() == null)
        return false;
}
try{
em.remove(de);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
public boolean atualizaDescricao(DescricaoEntity d){
boolean ret=true;
try{
em.merge(d);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

////////            DESCRIÇAO SAIDA


public boolean cadastreDescricaoSaida(DescricaoSaidaEntity descricaoSaida){
boolean ret=true;
Query query = em.createQuery("Select s.descricao_saida FROM DescricaoSaidaEntity s");
List<String> nome_descricoes_saidas = query.getResultList();
for(int j=0; j< nome_descricoes_saidas.size(); j++){
String sa = nome_descricoes_saidas.get(j);
if(sa.equals(descricaoSaida.getDescricao_saida()))
        ret = false;
}
```

```
if(ret == true)
        em.persist(descricaoSaida);
return ret;
}

public List<String> selecioneDescricoesSaida() {
Query query = em.createQuery("Select s.descricao_saida FROM DescricaoSaidaEntity s");
List<String> ufs = query.getResultList();
Query query2 = em.createQuery("Select f.nome_fornecedor FROM FornecedorEntity f");
List<String> ufs2 = query2.getResultList();
Query query3 = em.createQuery("Select a.nome FROM PrestadorFisicoEntity p, p.pessoa a");
List<String> ufs3 = query3.getResultList();
Query query4 = em.createQuery("Select t.nome_prestadora FROM PrestadoraServicoEntity t");
List<String> ufs4 = query4.getResultList();
ufs.addAll(ufs2);
ufs.addAll(ufs3);
ufs.addAll(ufs4);
return ufs;
}
```

```
public DescricaoSaidaEntity selecioneDescricaoSaidaAtualiza(String s) {
DescricaoSaidaEntity d = new DescricaoSaidaEntity();
Query query = em.createQuery("Select d FROM DescricaoSaidaEntity d WHERE d.descricao_saida = :s");
query.setParameter("s", s);
try{
d = (DescricaoSaidaEntity) query.getSingleResult();
}catch(NoResultException e){
        d.setDescricao_saida("");
}
d.getId();
d.getDescricao_saida();
return d;
}

public boolean removeDescricaoSaida(DescricaoSaidaEntity d){
boolean ret=true;
DescricaoSaidaEntity de = new DescricaoSaidaEntity();
try{
de = em.find(DescricaoSaidaEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
        return false;
}
try{
```

```
em.remove(de);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaDescricaoSaida(DescricaoSaidaEntity d){
boolean ret=true;
try{
em.merge(d);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

////////            SAIDA

public boolean cadastreSaida(SaidaEntity saida){
boolean ret=true;
if(saida == null)
return false;
```

```
        else
        em.persist(saida);
        return ret;
        }

        public List<SaidaEntity> selecioneSaidasDate(Date d1, Date d2) {
        Query query = em.createQuery("SELECT s FROM SaidaEntity s WHERE
s.data_emissao >= :d1 AND s.data_compensacao <= :d2");
        query.setParameter("d1",d1 , TemporalType.DATE);
        query.setParameter("d2",d2 , TemporalType.DATE);
        List<SaidaEntity> ufs = query.getResultList();
        return ufs;
        }

        public List<SaidaEntity> selecioneSaidaMes(){
        Query query = em.createQuery("SELECT s FROM SaidaEntity s");
        List<SaidaEntity> ufs = query.getResultList();
        return ufs;
        }

        public boolean removeSaida(Long id_remova){
        boolean ret=true;
        SaidaEntity sa = new SaidaEntity();
        try{
```

```
sa = em.find(SaidaEntity.class, id_remova);
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(sa.getId() == null)
        return false;
}
try{
        em.remove(sa);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
return ret;
}

////////        BALANÇO

public List<String> selecioneNomeDescricao(Date d1, Date d2){
Query query = em.createQuery("SELECT e.nome_descricao FROM EntradaEntity e WHERE e.data_emissao >= :d1 AND e.data_conpensacao <= :d2");
query.setParameter("d1",d1 , TemporalType.DATE);
query.setParameter("d2",d2 , TemporalType.DATE);
List<String> ufs = query.getResultList();
return ufs;
}
```

```
public List<EntradaEntity> selecioneBalancoEntrada(Date d1, Date d2, String nome) {

Query query = em.createQuery("SELECT e FROM EntradaEntity e WHERE e.data_emissao >= :d1 AND e.data_conpensacao <= :d2 AND e.nome_descricao = :nome");

query.setParameter("d1",d1 , TemporalType.DATE);

query.setParameter("d2",d2 , TemporalType.DATE);

query.setParameter("nome",nome);

List<EntradaEntity> ufs = query.getResultList();

return ufs;

}



////////            ENTRADA



public boolean cadastreEntrada(EntradaEntity entrada){

boolean ret=true;

if(entrada == null)

return false;

else

em.persist(entrada);

return ret;
```

```
}

public List<EntradaEntity> selecioneEntradaDate(Date d1, Date d2) {
Query query = em.createQuery("SELECT e FROM EntradaEntity e WHERE e.data_emissao >= :d1 AND e.data_conpensacao <= :d2");
query.setParameter("d1",d1 , TemporalType.DATE);
query.setParameter("d2",d2 , TemporalType.DATE);
List<EntradaEntity> ufs = query.getResultList();
return ufs;
}

public List<Long> getIdesEntrada(){

Query query = em.createQuery("Select e FROM EntradaEntity e");
List<EntradaEntity> ufs = query.getResultList();
ArrayList<Long> ides_entradas = new ArrayList<>();
for(int i=0;i<ufs.size();i++){
        ides_entradas.add(ufs.get(i).getId());
}

return ides_entradas;
}

public List<EntradaEntity> selecioneEntradaMes(){
```

```
Query query = em.createQuery("SELECT s FROM EntradaEntity s");
List<EntradaEntity> ufs = query.getResultList();
return ufs;
}

public boolean removeEntrada(long id_remova){
boolean ret=true;
EntradaEntity en = new EntradaEntity();
try{
en = em.find(EntradaEntity.class, id_remova);
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(en.getId() == null)
        return false;
}
try{
        em.remove(en);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
return ret;
}
```

```
////////// CIDADES

public boolean cadastreCidade(CidadeEntity cidade){
boolean ret=true;

em.persist(cidade);
return ret;

}

public List<String> selecioneCidades() {
Query query = em.createQuery("Select c.nome_cidade FROM CidadeEntity c");
List<String> nome_cidades = query.getResultList();
return nome_cidades;
}

public CidadeEntity selecioneCidadePorNome(String nome_cid) {
CidadeEntity c = new CidadeEntity();
Query query = em.createQuery("Select c FROM CidadeEntity c WHERE c.nome_cidade = :nome_cid");
query.setParameter("nome_cid", nome_cid);
```

```
try{
c = (CidadeEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNome_cidade("");
}
c.getId();
c.getNome_cidade();
c.getUf_cidade();
return c;
}


///////////////       DISTRITO


public boolean cadastreDistrito(DistritoEntity distrito){
boolean ret=true;

em.persist(distrito);
return ret;
}

public List<String> selecioneDistritos() {
```

```
        Query query = em.createQuery("Select d.nome_distrito FROM DistritoEntity
d");
        List<String> nome_distritos = query.getResultList();


        return nome_distritos;
        }


        public DistritoEntity selecioneDistritoPorNome(String nome_dis) {
        DistritoEntity d = new DistritoEntity();
        Query query = em.createQuery("Select d FROM DistritoEntity d WHERE
d.nome_distrito = :nome_dis");
        query.setParameter("nome_dis", nome_dis);
        try{
        d = (DistritoEntity) query.getSingleResult();
        }catch(NoResultException e){
                d.setNome_distrito("");
        }
        d.getId();
        d.getNome_distrito();
        d.getCidade();
        return d;
        }
```

```
//////////          RUAS

public boolean cadastreEndereco(EnderecoEntity endereco){
boolean ret=true;

em.persist(endereco);
return ret;

}

public List<String> selecioneRuas() {
Query query = em.createQuery("Select e.logradouro FROM EnderecoEntity e");
List<String> nome_ruas = query.getResultList();
return nome_ruas;
}

public EnderecoEntity selecioneRuasPorNome(String nome_rua) {
EnderecoEntity rua = new EnderecoEntity();
Query query = em.createQuery("Select e FROM EnderecoEntity e WHERE e.logradouro = :nome_rua");
query.setParameter("nome_rua", nome_rua);
try{
```

```
        rua = (EnderecoEntity) query.getSingleResult();
        }catch(NoResultException e){
                rua.setLogradouro("");
        }
        rua.getId();
        rua.getLogradouro();
        rua.getCep();
        //rua.getDistrito_endereco();
        return rua;
        }


}
```

OperacionalSessioBean:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package beans_sessao;

import entidades.AlergiaEntity;
```

```
import entidades.AlunoEntity;
import entidades.CartorioEntity;
import entidades.ChamadaEntity;
import entidades.CidadeEntity;
import entidades.DistritoEntity;
import entidades.EnderecoEntity;
import entidades.EscolaEntity;
import entidades.EstuChaEntity;
import entidades.FrequenciaEntity;
import entidades.IrmaoEntity;
import entidades.MaeEntity;
import entidades.PaiEntity;
import entidades.ProblemaSaudeEntity;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
```

```
/**
 *
 * @author petterson
 */
@Stateless
@LocalBean
public class OperacionalSessionBean {



        @PersistenceContext(unitName="jami-ejbPU")
        EntityManager em ;



        ////////////          ALUNOS



        public boolean cadastreAluno(AlunoEntity aluno){
                em.persist(aluno);
        return true;
        }
```

```
public List<String> selecioneNomeAlunos() {

String s = "Ativo";

Query query = em.createQuery("Select p.nome FROM AlunoEntity a, a.pessoa p WHERE a.situacao = :s");

query.setParameter("s", s);

List<String> ufs = query.getResultList();

return ufs;

}


public List<AlunoEntity> selecioneAlunos(List<String> nome_alunos) {

String s = "Ativo";

Query query = em.createQuery("Select a FROM AlunoEntity a, a.pessoa p WHERE p.nome in :nome_alunos AND a.situacao = :s");

query.setParameter("s", s);

query.setParameter("nome_alunos", nome_alunos);

List<AlunoEntity> ufs = query.getResultList();

return ufs;

}


public List<AlunoEntity> selecioneAlunosDocumentos() {

String s= "Ativo";

Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s");

query.setParameter("s", s);
```

```
List<AlunoEntity> ufs = query.getResultList();
return ufs;
}

public List<AlunoEntity> selecioneTodosAlunos() {
Query query = em.createQuery("Select a FROM AlunoEntity a");
List<AlunoEntity> ufs = query.getResultList();
return ufs;
}

public List<AlunoEntity> selecioneAlunosEmEspera() {
String s= "Espera";
Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s");
query.setParameter("s", s);
List<AlunoEntity> ufs = query.getResultList();
return ufs;
}

public List<AlunoEntity> selecioneAlunosDesistentes() {
String s = "Desistente";
Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s");
query.setParameter("s", s);
```

```
        List<AlunoEntity> ufs = query.getResultList();
        return ufs;
        }

        public List<AlunoEntity> selecioneAlunosPorTurnoSexo(String sexo, String turno) {
        String s = "Ativo";
        Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s AND a.sexo = :sexo AND a.turno = :turno");
        query.setParameter("s", s);
        query.setParameter("sexo", sexo);
        query.setParameter("turno", turno);
        List<AlunoEntity> ufs = query.getResultList();
        return ufs;
        }

        public List<AlunoEntity> selecioneAlunoMasculinoManha() {
        String s = "Ativo";
        String sexo = "Masculino";
        String turno ="Manhã";
        Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s AND a.sexo = :sexo AND a.turno = :turno");
        query.setParameter("s", s);
        query.setParameter("sexo", sexo);
```

```
query.setParameter("turno", turno);
List<AlunoEntity> ufs = query.getResultList();
return ufs;
}

public List<AlunoEntity> selecioneAlunoMasculinoTarde() {
String s = "Ativo";
String sexo = "Masculino";
String turno ="Tarde";
Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s AND a.sexo = :sexo AND a.turno = :turno");
query.setParameter("s", s);
query.setParameter("sexo", sexo);
query.setParameter("turno", turno);
List<AlunoEntity> ufs = query.getResultList();
return ufs;
}

public List<AlunoEntity> selecioneAlunoFemininoManha() {
String s = "Ativo";
String sexo = "Feminino";
String turno ="Manhã";
Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s AND a.sexo = :sexo AND a.turno = :turno");
```

```
query.setParameter("s", s);

query.setParameter("sexo", sexo);

query.setParameter("turno", turno);

List<AlunoEntity> ufs = query.getResultList();

return ufs;

}


public List<AlunoEntity> selecioneAlunoFemininoTarde() {

String s = "Ativo";

String sexo = "Feminino";

String turno ="Tarde";

Query query = em.createQuery("Select a FROM AlunoEntity a WHERE a.situacao = :s AND a.sexo = :sexo AND a.turno = :turno");

query.setParameter("s", s);

query.setParameter("sexo", sexo);

query.setParameter("turno", turno);

List<AlunoEntity> ufs = query.getResultList();

return ufs;

}


public AlunoEntity selecioneAlunosPorNome(String nome_aluno) {

String ativo = "Ativo";

AlunoEntity ufs = new AlunoEntity();
```

```
Query query = em.createQuery("Select a FROM AlunoEntity a, a.pessoa p WHERE p.nome = :nome_aluno AND a.situacao = :Ativo");
query.setParameter("nome_aluno", nome_aluno);
query.setParameter("Ativo", ativo);
try{
        ufs = (AlunoEntity) query.getSingleResult();
}catch(NoResultException e){
        ufs.setFone_aluno("");
}
return ufs;
}

public AlunoEntity selecioneAlunoAtualiza(String s) {
AlunoEntity a = new AlunoEntity();
Query query = em.createQuery("Select a FROM AlunoEntity a, a.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
try{
a = (AlunoEntity) query.getSingleResult();
}catch(NoResultException e){
        a.setFone_aluno("");
}
a.getId();
a.getFone_aluno();
```

```
a.getRg();
a.getOrgao_expedidor();
a.getData_expedicao();
a.getSexo();
a.getData_matricula();
a.getNumero_certidao();
a.getLivro_certidao();
a.getFolha_certidao();
a.getCartorio();
a.getRua_aluno();
a.getNumero_casa();
a.getPonto_referencia();
a.getCidade_nascimento();
a.getCor_raca();
a.getEscola();
a.getMatricula_escola();
a.getSerie();
a.getTurno();
a.getAlergia();
a.getProblema_saude();
a.getBolsa_familia();
a.getIrmaos();
a.getPai();
```

```
a.getMae();
a.getSituacao();
a.getPessoa();
return a;
}

public boolean removeAluno(AlunoEntity a){
AlunoEntity al = new AlunoEntity();
CartorioEntity c = a.getCartorio();
EscolaEntity es = a.getEscola();
CidadeEntity ci = a.getCidade_nascimento();
EnderecoEntity en = a.getRua_aluno();
PaiEntity p = a.getPai();
MaeEntity m = a.getMae();
List<ProblemaSaudeEntity> list_pro = a.getProblema_saude();
List<AlergiaEntity> list_aler = a.getAlergia();
List<IrmaoEntity> list_irm = a.getIrmaos();
try{
al = em.find(AlunoEntity.class, a.getId());
//System.out.println("aluno"+al.getNome_aluno());
}catch(java.lang.IllegalArgumentException e){
        System.out.println("erro aluno");
        return false;
```

```
        }
        try{
        for(int i=0; i< list_irm.size(); i++){
                IrmaoEntity ir = list_irm.get(i);
                if(ir.getList_aluno().size() > 1){
                ir.getList_aluno().remove(al);
                al.getIrmaos().remove(ir);
                try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
                try{em.merge(ir);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro ir");return false;}
                }else{try{em.remove(ir);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro ir");return false;};}
        }
        for(int i=0; i< list_aler.size(); i++){
                AlergiaEntity aler = list_aler.get(i);
                if(aler.getList_aluno().size() > 1){
                aler.getList_aluno().remove(al);
                al.getAlergia().remove(aler);
                try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
                try{em.merge(aler);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro aler");return false;}
                }else{try{em.remove(aler);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro aler");return false;};}
```

```
        }
        for(int i=0; i< list_pro.size(); i++){
                ProblemaSaudeEntity proble = list_pro.get(i);
                if(proble.getList_aluno().size() > 1){
                proble.getList_aluno().remove(al);
                al.getProblema_saude().remove(proble);
                try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
                try{em.merge(proble);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro proble");return false;}
                }else{try{em.remove(proble);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro proble");return false;};}
        }
        if(c.getList_aluno().size() >1){
                c.getList_aluno().remove(al);
                al.setCartorio(null);
                try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
                try{em.merge(c);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro c");return false;}
        }else{try{em.remove(c);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro c");return false;};}
        if(es.getList_aluno().size() > 1){
        es.getList_aluno().remove(al);
        al.setEscola(null);
```

```
        try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}

        try{em.merge(es);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro es");return false;}

        }else{try{em.remove(es);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro es");return false;};}

        if(ci.getList_aluno().size() > 1){

        ci.getList_aluno().remove(al);

        al.setCidade_nascimento(null);

        try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}

        try{em.merge(ci);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro ci");return false;}

        }else{try{em.remove(ci);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro ci");return false;};}

        if(en.getList_aluno().size() >1){

        en.getList_aluno().remove(al);

        al.setRua_aluno(null);

        try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}

        try{em.merge(en);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro en");return false;}

        }else{try{em.remove(en);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro en");return false;};}

        if(m.getList_aluno().size() > 1){

        m.getList_aluno().remove(al);
```

```
        al.setMae(null);
        try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
        try{em.merge(m);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro m");return false;}
        }else{try{em.remove(m);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro m");return false;};}
        System.out.println(p.getList_aluno().size());
        if(p.getList_aluno().size() > 1){
        p.getList_aluno().remove(al);
        al.setPai(null);
        try{em.merge(al);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro pai");return false;}
        try{em.merge(p);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro p a");return false;}
        }else{try{em.remove(p);}catch(java.lang.IllegalArgumentException
e){System.out.println("erro p r");return false;};}
        em.remove(al);
        }catch(java.lang.IllegalArgumentException e){
        System.out.println("erro aluno");
        return false;
        }
        return true;
        }
```

```
public boolean atualizaAluno(AlunoEntity a){
boolean ret=true;
try{
em.merge(a);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
//////////          EstuChamad

public boolean cadastreEstuChamada(EstuChaEntity a){
boolean ret=true;
```

```
            em.persist(a);
        return ret;
        }

        public List<EstuChaEntity> selecioneEstuCha(List<String> nomes, String
data_hoje , String nome_chamada){
        Query query = em.createQuery("Select e FROM EstuChaEntity e WHERE
e.data_cadastro = :data_hoje AND "
                + "e.nome_chamada = :nome_chamada AND e.nome_estunte in
:nomes");
        query.setParameter("data_hoje", data_hoje);
        query.setParameter("nome_chamada", nome_chamada);
        query.setParameter("nomes", nomes);
        List<EstuChaEntity> ufs = query.getResultList();
        return ufs;
        }

        public boolean atualizaEstuCha(EstuChaEntity es){
        boolean ret=true;
        try{
        em.merge(es);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
```

```
return ret;
}

public boolean removeEstuCha(EstuChaEntity es){
boolean ret=true;
EstuChaEntity est = new EstuChaEntity();
try{
est = em.find(EstuChaEntity.class, es.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(est.getId() == null)
        return false;
}
try{
em.remove(est);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}


//////////          ALERGIAS
```

```
public boolean cadastreAlergia(AlergiaEntity alergia){
boolean ret=true;
Query query = em.createQuery("Select a.nome_alergia FROM AlergiaEntity
a");
List<String> nome_alergias = query.getResultList();
for(int j=0; j< nome_alergias.size(); j++){
String sa = nome_alergias.get(j);
if(sa.equals(alergia.getNome_alergia()))
        ret = false;
}
if(ret == true)
        em.persist(alergia);
return ret;
}

public List<String> selecioneAlergias() {
Query query = em.createQuery("Select a.nome_alergia FROM AlergiaEntity
a");
List<String> ufs = query.getResultList();
return ufs;
}
```

```
public List<AlergiaEntity> selecioneListAlergias(ArrayList<String> nome_alergias) {
Query query = em.createQuery("Select a FROM AlergiaEntity a WHERE a.nome_alergia in :nome_alergias");
query.setParameter("nome_alergias", nome_alergias);
List<AlergiaEntity> ufs = query.getResultList();
return ufs;
}

public AlergiaEntity selecioneAlergiaAtualiza(String s) {
AlergiaEntity a = new AlergiaEntity();
Query query = em.createQuery("Select p FROM AlergiaEntity p WHERE p.nome_alergia = :s");
query.setParameter("s", s);
try{
a = (AlergiaEntity) query.getSingleResult();
}catch(NoResultException e){
        a.setNome_alergia("");
}
a.getId();
a.getNome_alergia();
a.getGrau_perigo_alergia();
return a;
}
```

```
public boolean removeAlergia(AlergiaEntity a){
boolean ret=true;
AlergiaEntity al = new AlergiaEntity();
try{
al = em.find(AlergiaEntity.class, a.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(al.getId() == null)
        return false;
}
try{
em.remove(al);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaAlergia(AlergiaEntity a){
boolean ret=true;
try{
em.merge(a);
```

```
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

// PROBLEMA SAUDE

public boolean cadastreProblemaSaude(ProblemaSaudeEntity problema){
boolean ret=true;
Query query = em.createQuery("Select p.problema_saude FROM ProblemaSaudeEntity p");
List<String> nome_problemas = query.getResultList();
for(int j=0; j< nome_problemas.size(); j++){
String sa = nome_problemas.get(j);
if(sa.equals(problema.getProblema_saude()))
        ret = false;
}
if(ret == true)
```

```
            em.persist(problema);
        return ret;
        }

  public List<String> selecioneProblemaSaude() {
        Query query = em.createQuery("Select p.problema_saude FROM
ProblemaSaudeEntity p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

         public List<ProblemaSaudeEntity> selecioneListaProblemas(List<String>
nome_problemas) {
        Query query = em.createQuery("Select p FROM ProblemaSaudeEntity p
WHERE p.problema_saude in :nome_problemas");
        query.setParameter("nome_problemas", nome_problemas);
        List<ProblemaSaudeEntity> ufs = query.getResultList();
        return ufs;
        }

        public ProblemaSaudeEntity selecioneProblemaAtualiza(String s) {
        ProblemaSaudeEntity p = new ProblemaSaudeEntity();
        Query query = em.createQuery("Select p FROM ProblemaSaudeEntity p
WHERE p.problema_saude = :s");
```

```
query.setParameter("s", s);
try{
p = (ProblemaSaudeEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setProblema_saude("");
}
p.getId();
p.getProblema_saude();
p.getGrau_perigo();
return p;
}

public boolean removeProblema(ProblemaSaudeEntity p){
boolean ret=true;
ProblemaSaudeEntity po = new ProblemaSaudeEntity();
try{
po = em.find(ProblemaSaudeEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(po.getId() == null)
        return false;
}
try{
```

```
em.remove(po);
}catch(java.lang.IllegalArgumentException e){
        return false;
}
return ret;
}

public boolean atualizaProblema(ProblemaSaudeEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
        // CIDADES


        public boolean cadastreCidade(CidadeEntity cidade){
        boolean ret=true;
        Query query = em.createQuery("Select c.nome_cidade FROM CidadeEntity
c");
        List<String> nome_cidades = query.getResultList();
        for(int j=0; j< nome_cidades.size(); j++){
        String ci = nome_cidades.get(j);
        if(ci.equals(cidade.getNome_cidade()))
                ret = false;
        }
        if(ret == true)
                em.persist(cidade);
        return ret;


        }


        public List<String> selecioneCidades() {
        Query query = em.createQuery("Select c.nome_cidade FROM CidadeEntity
c");
        List<String> nome_cidades = query.getResultList();
        return nome_cidades;
        }
```

```
public CidadeEntity selecioneCidadeAtualiza(String s) {
CidadeEntity c = new CidadeEntity();
Query query = em.createQuery("Select c FROM CidadeEntity c WHERE c.nome_cidade = :s");
query.setParameter("s", s);
try{
c = (CidadeEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNome_cidade("");
}
c.getId();
c.getNome_cidade();
c.getUf_cidade();
return c;
}

public boolean removeAlunoAtualizaCidade(AlunoEntity a){
CidadeEntity c=a.getCidade_nascimento();
String nome = c.getNome_cidade();
List<String> n_a = new ArrayList<>();
Query query = em.createQuery("Select a.nome_aluno FROM AlunoEntity a JOIN a.cidade_nascimento c WHERE c.nome_cidade = :nome");
query.setParameter("nome", nome);
```

```
try{
n_a = query.getResultList();
if(n_a.size() > 1){
        a.setCidade_nascimento(null);
        try{
        em.merge(a);
        }catch(java.lang.IllegalArgumentException e){
        System.out.println("Atualiza Aluno");
                return true;
        }
}else{
        return true;
}
}catch(NoResultException e){
        return false;
}
return true;
}

/*public boolean removeCidadeAtualizaDistrito(CidadeEntity c){
boolean res=true;
String nome = c.getNome_cidade();
CidadeEntity ci = new CidadeEntity();
```

```
List<DistritoEntity> l_d = new ArrayList<>();
Query query = em.createQuery("Select d FROM DistritoEntity d JOIN d.cidade
c WHERE c.nome_cidade = :nome");
query.setParameter("nome", nome);
try{
l_d = query.getResultList();
if(l_d.size() >= 1){
for(int i=0; i<l_d.size(); i++){
this.removeDistrito(l_d.get(i));
}
}else{
try{
ci = em.find(CidadeEntity.class, c.getId());
}catch(java.lang.IllegalArgumentException e){
System.out.println("busca cidade");
return false;
}
try{
em.remove(ci);
}catch(java.lang.IllegalArgumentException e){
System.out.println("remove");
return false;
}
}
```

```
}catch(java.lang.IllegalArgumentException e){
        System.out.println("busca distrito");
                return true;
}
return res;
}*/

public boolean removeDistritoAtualizaCidade(DistritoEntity d){
CidadeEntity di = d.getCidade();
DistritoEntity dis = new DistritoEntity();
        if(di.getList_distrito().size() > 1){
        di.getList_distrito().remove(d);
        try{
        em.merge(di);
        }catch(java.lang.IllegalArgumentException e){
        System.out.println("erro cidade");
                return false;
        }
        try{
                dis = em.find(DistritoEntity.class, d.getId());
                dis.setCidade(null);
        }catch(java.lang.IllegalArgumentException e){
                System.out.println("busca cidade");
```

```
                return false;
        }
        try{
                em.remove(dis);
        }catch(java.lang.IllegalArgumentException e){
                System.out.println("remove");
                return false;
        }
        }else{
        try{
                em.remove(di);
        }catch(java.lang.IllegalArgumentException e){
                System.out.println("remove");
                return false;
        }
        }
return true;
}

public boolean removeCidade(CidadeEntity c){
boolean ret=true;
CidadeEntity ci = new CidadeEntity();
try{
```

```
ci = em.find(CidadeEntity.class, c.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ci.getId() == null)
        return false;
}
try{
em.remove(ci);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaCidade(CidadeEntity c){
boolean ret=true;
try{
em.merge(c);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
        //      DISTRITO


        public boolean cadastreDistrito(DistritoEntity distrito){
        boolean ret=true;
        Query query = em.createQuery("Select d.nome_distrito FROM DistritoEntity
d");
        List<String> nome_distritos = query.getResultList();
        for(int j=0; j< nome_distritos.size(); j++){
        String ci = nome_distritos.get(j);
        if(ci.equals(distrito.getNome_distrito()))
                ret = false;
        }
        if(ret == true)
```

```
            em.persist(distrito);
        return ret;
        }

        public List<String> selecioneDistritos() {
        Query query = em.createQuery("Select d.nome_distrito FROM DistritoEntity
d");
        List<String> nome_distritos = query.getResultList();
        return nome_distritos;
        }

        public DistritoEntity selecioneDistritoAtualiza(String s) {
        DistritoEntity d = new DistritoEntity();
        Query query = em.createQuery("Select d FROM DistritoEntity d WHERE
d.nome_distrito = :s");
        query.setParameter("s", s);
        try{
        d = (DistritoEntity) query.getSingleResult();
        }catch(NoResultException e){
                d.setNome_distrito("");
        }
        d.getId();
        d.getNome_distrito();
        d.getCidade();
```

```
return d;
}

public boolean removeDistrito(DistritoEntity d){
boolean ret=true;
DistritoEntity di = new DistritoEntity();
try{
di = em.find(DistritoEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(di.getId() == null)
        return false;
}
try{
em.remove(di);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaDistrito(DistritoEntity d){
boolean ret=true;
```

```
try{
em.merge(d);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}




//                  RUAS


public boolean cadastreEndereco(EnderecoEntity endereco){
boolean ret=true;
Query query = em.createQuery("Select e.logradouro FROM EnderecoEntity
e");
List<String> nome_ruas = query.getResultList();
for(int j=0; j< nome_ruas.size(); j++){
String ci = nome_ruas.get(j);
if(ci.equals(endereco.getLogradouro()))
        ret = false;
```

```
        }
        if(ret == true)
                em.persist(endereco);
        return ret;


        }


        public List<String> selecioneRuas() {
        Query query = em.createQuery("Select e.logradouro FROM EnderecoEntity e");
        List<String> ufs = query.getResultList();
        return ufs;
        }


        public EnderecoEntity selecioneEnderecoAtualiza(String s) {
        EnderecoEntity n = new EnderecoEntity();
        Query query = em.createQuery("Select e FROM EnderecoEntity e WHERE e.logradouro = :s");
        query.setParameter("s", s);
        try{
        n = (EnderecoEntity) query.getSingleResult();
        }catch(NoResultException e){
                n.setLogradouro("");
        }
```

```
n.getId();
n.getLogradouro();
n.getCep();
n.getDistrito_endereco();
return n;
}

public boolean removeAlunoAtualizaRua(AlunoEntity a){
EnderecoEntity en=a.getRua_aluno();
String nome = en.getLogradouro();
List<String> n_a = new ArrayList<>();
Query query = em.createQuery("Select a.nome_aluno FROM AlunoEntity a JOIN a.rua_aluno r WHERE r.logradouro = :nome");
query.setParameter("nome", nome);
try{
n_a = query.getResultList();
if(n_a.size() > 1){
        a.setRua_aluno(null);
        try{
        em.merge(a);
        }catch(java.lang.IllegalArgumentException e){
        System.out.println("Atualiza Aluno");
                return true;
        }
```

```
}else{
        //System.out.println(a.getPai().getNome_pai()+"return");
        return true;
}
}catch(NoResultException e){
        return false;
}
return true;
}

public boolean removeEndereco(EnderecoEntity n){
boolean ret=true;
EnderecoEntity en = new EnderecoEntity();
try{
en = em.find(EnderecoEntity.class, n.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(en.getId() == null)
        return false;
}
try{
em.remove(en);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
return ret;
}

public boolean atualizaEndereco(EnderecoEntity n){
boolean ret=true;
try{
em.merge(n);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

//        ESCOLAS


public boolean cadastreEscola(EscolaEntity escola){
boolean ret=true;
Query query = em.createQuery("Select e.nome_escola FROM EscolaEntity
e");
List<String> nome_escolas = query.getResultList();
for(int j=0; j< nome_escolas.size(); j++){
```

```
String sa = nome_escolas.get(j);
if(sa.equals(escola.getNome_escola()))
        ret = false;
}
if(ret == true)
        em.persist(escola);
return ret;
}

public List<String> selecioneEscolas() {
Query query = em.createQuery("Select e.nome_escola FROM EscolaEntity e");
List<String> ufs = query.getResultList();
return ufs;
}

public EscolaEntity selecioneEscolaAtualiza(String s) {
EscolaEntity e = new EscolaEntity();
Query query = em.createQuery("Select e FROM EscolaEntity e WHERE e.nome_escola = :s");
query.setParameter("s", s);
try{
e = (EscolaEntity) query.getSingleResult();
}catch(NoResultException ex){
```

```
            e.setNome_escola("");
}
e.getId();
e.getNome_escola();
e.getRua_escola();
e.getNumero_escola();
return e;
}

public boolean removeEscola(EscolaEntity e){
boolean ret=true;
EscolaEntity es = new EscolaEntity();
try{
es = em.find(EscolaEntity.class, e.getId());
}catch(java.lang.IllegalArgumentException ex){
            if(es.getId() == null)
            return false;
}
try{
em.remove(es);
}catch(java.lang.IllegalArgumentException ex){
return false;
}
```

```
return ret;

}


public boolean atualizaEscola(EscolaEntity e){

boolean ret=true;

try{

em.merge(e);

}catch(java.lang.IllegalArgumentException xe){

return false;

}

return ret;

}
```

//// CARTORIOS

```
public boolean cadastreCartorio(CartorioEntity cartorio){
boolean ret=true;
Query query = em.createQuery("Select c.nome_cartorio FROM CartorioEntity
c");
List<String> nome_cartorios = query.getResultList();
for(int j=0; j< nome_cartorios.size(); j++){
String sa = nome_cartorios.get(j);
if(sa.equals(cartorio.getNome_cartorio()))
        ret = false;
}
if(ret == true)
        em.persist(cartorio);
return ret;
}

public List<String> selecioneCartorios() {
Query query = em.createQuery("Select c.nome_cartorio FROM CartorioEntity
c");
List<String> ufs = query.getResultList();
return ufs;
}
```

```
public CartorioEntity selecioneCartorioAtualiza(String s) {
CartorioEntity c = new CartorioEntity();
Query query = em.createQuery("Select c FROM CartorioEntity c WHERE c.nome_cartorio = :s");
query.setParameter("s", s);
try{
c = (CartorioEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNome_cartorio("");
}
c.getId();
c.getNome_cartorio();
c.getRua_cartorio();
c.getNumero_caetorio();
return c;
}

public boolean removeCartorio(CartorioEntity c){
boolean ret=true;
if(c != null){
CartorioEntity ca = new CartorioEntity();
EnderecoEntity en = c.getRua_cartorio();
try{
ca = em.find(CartorioEntity.class, c.getId());
```

```
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ca.getId() == null)
        return false;
}
try{
em.remove(ca);
}catch(java.lang.IllegalArgumentException e){
return false;
}
}else{
        return true;
}
return ret;
}

public boolean atualizaCartorio(CartorioEntity c){
boolean ret=true;
try{
em.merge(c);
}catch(java.lang.IllegalArgumentException e){
return false;
}
```

```
return ret;
}

//////////////        MAE

public boolean cadastreMae(MaeEntity mae){
        em.persist(mae);
return true;
}

public List<String> selecioneMaes() {
Query query = em.createQuery("Select p.nome FROM MaeEntity a, a.pessoa p");
List<String> ufs = query.getResultList();
return ufs;
}

public MaeEntity selecioneMaeAtualiza(String s) {
MaeEntity m = new MaeEntity();
Query query = em.createQuery("Select m FROM MaeEntity m, m.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
```

```
try{
m = (MaeEntity) query.getSingleResult();
}catch(NoResultException e){
        m.setFone_mae("");
}
m.getId();
m.getRua_mae();
m.getProfissao_mae();
m.getLocal_trabalho_mae();
m.getFone_mae();
m.getEscolaridade_mae();
m.getRua_mae();
m.getNumero_casa_mae();
m.getPessoa();
m.getList_aluno();
return m;
}

public boolean atualizaMae(MaeEntity m){
boolean ret=true;
try{
em.merge(m);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
return ret;
}

public boolean removeMae(MaeEntity p){
boolean ret=true;
MaeEntity pi = new MaeEntity();
EnderecoEntity pa = p.getRua_mae();
try{
pi = em.find(MaeEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(pi.getFone_mae().equals(""))
        return false;
}
try{

if(pa.getList_aluno().size() > 1){
        pa.setLogradouro(null);
        this.atualizaEndereco(pa);
}else{this.removeEndereco(pa);}
em.remove(pi);
```

```
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}



////      IRMÃOS



public boolean cadastreIrmao(IrmaoEntity irmao){
        em.persist(irmao);
return true;
}

public List<String> selecioneIrmaos() {
Query query = em.createQuery("Select p.nome FROM IrmaoEntity a, a.pessoa p");
List<String> ufs = query.getResultList();
return ufs;
}

public List<IrmaoEntity> selecioneListaIrmaos(List<String> nome_irmaos) {
```

```
Query query = em.createQuery("Select i FROM IrmaoEntity i, i.pessoa p WHERE p.nome in :nome_irmaos");
query.setParameter("nome_irmaos", nome_irmaos);
List<IrmaoEntity> ufs = query.getResultList();
return ufs;
}

public IrmaoEntity selecioneIrmaoAtualiza(String s) {
IrmaoEntity i = new IrmaoEntity();
Query query = em.createQuery("Select i FROM IrmaoEntity i, i.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
try{
i = (IrmaoEntity) query.getSingleResult();
}catch(NoResultException e){
        i.setFone("");
}
i.getId();
i.getPessoa();
i.getList_aluno();
i.getFone();
return i;
}
```

```
public boolean removeIrmao(IrmaoEntity i){
boolean ret=true;
IrmaoEntity ir = new IrmaoEntity();
try{
ir = em.find(IrmaoEntity.class, i.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ir.getId() == null)
        return false;
}
try{
em.remove(ir);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaIrmao(IrmaoEntity i){
boolean ret=true;
try{
em.merge(i);
}catch(java.lang.IllegalArgumentException e){
```

```
return false;
}
return ret;
}



////////// PAI

public boolean cadastrePai(PaiEntity pai){
        em.persist(pai);
return true;
}

public List<String> selecionePais() {
Query query = em.createQuery("Select a.nome FROM PaiEntity p, p.pessoa
a");
List<String> ufs = query.getResultList();
return ufs;
}

public PaiEntity selecionePaiAtualiza(String s) {
PaiEntity p = new PaiEntity();
```

```
Query query = em.createQuery("Select p FROM PaiEntity p, p.pessoa a WHERE a.nome = :s");
query.setParameter("s", s);
try{
p = (PaiEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setFone_pai("");
}
p.getId();
p.getRg_pai();
p.getFone_pai();
p.getProfissao_pai();
p.getLocal_trabalho_pai();
p.getEscolaridade_pai();
p.getRua_pai();
p.getNumero_casa_pai();
p.getPessoa();
p.getList_aluno();
return p;
}

public boolean removePai(PaiEntity p){
boolean ret=true;
PaiEntity pi = new PaiEntity();
```

```
EnderecoEntity pa = p.getRua_pai();
try{
pi = em.find(PaiEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(pi.getFone_pai().equals(""))
        return false;
}
try{

if(pa.getList_aluno().size() > 1){
        pa.setLogradouro(null);
        this.atualizaEndereco(pa);
}else{this.removeEndereco(pa);}
em.remove(pi);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaPai(PaiEntity p){
boolean ret=true;
```

```
        try{
        em.merge(p);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret
    }


        //////////////      CHAMADA


        public boolean cadastreChamada(ChamadaEntity chamada){
        boolean ret=true;
        Query query = em.createQuery("Select c.nome_chamada FROM
ChamadaEntity c");
        List<String> nome_chamadas = query.getResultList();
        for(int j=0; j< nome_chamadas.size(); j++){
        String sa = nome_chamadas.get(j);
        if(sa.equals(chamada.getNome_chamada()))
                ret = false;
        }
        if(ret == true)
                em.persist(chamada);
        return ret;
```

```
}

public List<String> selecioneChamadas() {
Query query = em.createQuery("Select c.nome_chamada FROM ChamadaEntity c");
List<String> ufs = query.getResultList();
return ufs;
}

public List<String> selecioneChamadasComTurmas() {
Query query = em.createQuery("Select t.chamada.nome_chamada FROM TurmaEntity t");
List<String> ufs = query.getResultList();
return ufs;
}

public boolean removeChamada(ChamadaEntity c){
boolean ret=true;
ChamadaEntity ca = new ChamadaEntity();
try{
ca = em.find(ChamadaEntity.class, c.getId());
}catch(java.lang.IllegalArgumentException e){
        if(ca.getId() == null)
        return false;
```

```
        }
        try{
        em.remove(ca);
        }catch(java.lang.IllegalArgumentException e){
        if(ca.getId() == null)
                return false;
        }
        return ret;
        }

        public ChamadaEntity selecioneChamadaAtualiza(String s){
        ChamadaEntity c = new ChamadaEntity();
        Query query = em.createQuery("Select c FROM ChamadaEntity c WHERE
c.nome_chamada = :s");
        query.setParameter("s", s);
        try{
        c = (ChamadaEntity) query.getSingleResult();
        }catch(NoResultException e){
                c.setNome_chamada("");
        }
        c.getId();
        c.getNome_chamada();
        c.getData_criacao_chamada();
        return c;
```

```
        }



  public boolean atualizaChamada(ChamadaEntity c){
        boolean ret=true;
        try{
        em.merge(c);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }



        //////////////  FREQUENCIA


        public boolean cadsatreFrequencia(FrequenciaEntity frequencia, Date
data_hoje){
        boolean ret=true;
        FrequenciaEntity f = new FrequenciaEntity();
        String nome= frequencia.getNome_chamada();
        Query query = em.createQuery("Select f FROM FrequenciaEntity f WHERE
f.nome_chamada = :nome AND f.data_frequencia = :data_hoje");
        query.setParameter("data_hoje", data_hoje);
```

```
query.setParameter("nome", nome);
try{
f = (FrequenciaEntity) query.getSingleResult();
}catch(NoResultException e){
        f.setNome_chamada("");
}
if(f.getNome_chamada().equals(""))
        em.persist(frequencia);
else
        return false;
return ret;
}

public List<FrequenciaEntity> selecioneFrequenciaCalcula(Date di, Date df, String chamada){
List<FrequenciaEntity> f = new ArrayList<>();
Query query = em.createQuery("Select f FROM FrequenciaEntity f WHERE f.nome_chamada = :chamada AND f.data_frequencia BETWEEN :di AND :df ");
query.setParameter("di", di);
query.setParameter("df", df);
query.setParameter("chamada", chamada);
try{
f = query.getResultList();
//System.out.println(f.size());
```

```
        }catch(NoResultException e){
                f = null;
        }
        return f;
        }

        public List<EstuChaEntity> selecioneEstuCha(Date di, Date df, String nome){
        List<EstuChaEntity> f = new ArrayList<>();
        Query query = em.createQuery("Select f FROM EstuChaEntity f WHERE f.nome_estunte = :nome AND f.data_cadastro BETWEEN :di AND :df ");
        query.setParameter("di", di);
        query.setParameter("df", df);
        query.setParameter("nome", nome);
        try{
        f = query.getResultList();
        //System.out.println(f.size());
        }catch(NoResultException e){
                f = null;
        }
        return f;
        }

        public boolean verificaFrequenciaCadastrada(String chamada, Date data_hoje){
```

```
boolean ret=true;

FrequenciaEntity f = new FrequenciaEntity();

Query query = em.createQuery("Select f FROM FrequenciaEntity f WHERE
f.nome_chamada = :chamada AND"

                    + " f.data_frequencia = :data_hoje");

query.setParameter("data_hoje", data_hoje);

query.setParameter("chamada", chamada);

try{

f = (FrequenciaEntity) query.getSingleResult();

}catch(NoResultException e){

        return false;

}

return ret;

}


public FrequenciaEntity selecioneFrequenciaAtualiza(String s, Date d) {

FrequenciaEntity f = new FrequenciaEntity();

Query query = em.createQuery("Select f FROM FrequenciaEntity f WHERE
f.nome_chamada = :s AND f.data_frequencia = :d");

query.setParameter("s", s);

query.setParameter("d", d);

try{

f = (FrequenciaEntity) query.getSingleResult();

}catch(NoResultException e){
```

```
            f.setNome_chamada("");
}
f.getId();
f.getNome_chamada();
f.getData_frequencia();
f.getEstudantes();
return f;
}

public boolean atualizaFruequencia(FrequenciaEntity f){
boolean ret=true;
try{
em.merge(f);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean removeFrequencia(FrequenciaEntity c){
boolean ret=true;
FrequenciaEntity fr = new FrequenciaEntity();
try{
```

```
        fr = em.find(FrequenciaEntity.class, c.getId());
        }catch(java.lang.IllegalArgumentException e){
                if(fr.getId() == null)
                return false;
        }
        try{
        em.remove(fr);
        }catch(java.lang.IllegalArgumentException e){
        if(fr.getId() == null)
                return false;
        }
        return ret;
        }
}
```

GerenteSessionBean:

```
package beans_sessao;

import entidades.AlunoEntity;
import entidades.ApoiadorEntity;
import entidades.AtividadeEntity;
import entidades.BeneficioConcedidoEntity;
import entidades.CargoEntity;
```

```
import entidades.DepartamentoEntity;
import entidades.DoacaoEntity;
import entidades.DoadorEntity;
import entidades.EnderecoEntity;
import entidades.FuncionarioEntity;
import entidades.InstituicaoEntity;
import entidades.ContribuinteEntity;
import entidades.PapelEntity;
import entidades.PessoaEntity;
import entidades.ProgramaEntity;
import entidades.ProjetoEntity;
import entidades.SalarioEntity;
import entidades.SocialEntity;
import entidades.TipoBeneficioEntity;
import entidades.TipoDoacaoEntity;
import entidades.TipoInstituicaoEntity;
import entidades.TurmaEntity;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.ejb.EJBException;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
```

```
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.NonUniqueResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import javax.persistence.TemporalType;

/**
 *
 * @author petterson
 */
@Stateless
@LocalBean
public class GerenteSessionBean {

        @PersistenceContext(unitName="jami-ejbPU")
        EntityManager em ;
```

```
        ////////          SALARIO


        public boolean cadastreSalario(SalarioEntity salario){
        boolean ret=true;
        Query query = em.createQuery("Select s.valor_salario FROM SalarioEntity
s");
        List<Double> valor_salario = query.getResultList();
        for(int j=0; j< valor_salario.size(); j++){
        double sa = valor_salario.get(j);
        if(sa == salario.getValor_salario())
                return false;
        }
        if(ret == true)
                em.persist(salario);
        return ret;
        }


        public List<Double> selecioneSalarios() {
        Query query = em.createQuery("Select s.valor_salario FROM SalarioEntity
s");
        List<Double> ufs = query.getResultList();
        return ufs;
        }
```

```
public boolean removeSalario(SalarioEntity salario){
boolean ret=true;
double s = salario.getValor_salario();
SalarioEntity sal = new SalarioEntity();
Query query = em.createQuery("Select s FROM SalarioEntity s WHERE s.valor_salario = :s");
query.setParameter("s", s);
try{
sal =  (SalarioEntity) query.getSingleResult();
}catch(NoResultException e){
        return false;
}
try{
em.remove(sal);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public SalarioEntity selecioneSalarioAtualiza(double s) {
SalarioEntity sa = new SalarioEntity();
```

```
Query query = em.createQuery("Select s FROM SalarioEntity s WHERE s.valor_salario = :s");
query.setParameter("s", s);
try{
sa = (SalarioEntity) query.getSingleResult();
}catch(NoResultException e){
        sa.setValor_salario(0);
}
sa.getId();
sa.getValor_salario();
return sa;
}
```

```
////////        CARGO

public boolean cadastreCargo(CargoEntity cargo){
boolean ret=true;
Query query = em.createQuery("Select c.nome_cargo FROM CargoEntity c");
List<String> nome_cargos = query.getResultList();
for(int j=0; j< nome_cargos.size(); j++){
```

```
String sa = nome_cargos.get(j);
if(sa.equals(cargo.getNome_cargo()))
        ret = false;
}
if(ret == true)
        em.persist(cargo);
return ret;
}

public List<String> selecioneCargos() {
Query query = em.createQuery("Select c.nome_cargo FROM CargoEntity c");
List<String> ufs = query.getResultList();
return ufs;
}

public CargoEntity selecioneCargoAtualiza(String s) {
CargoEntity c = new CargoEntity();
Query query = em.createQuery("Select c FROM CargoEntity c WHERE
c.nome_cargo = :s");
query.setParameter("s", s);
try{
c = (CargoEntity) query.getSingleResult();
}catch(NoResultException e){
        c.setNome_cargo("");
```

```
}
c.getId();
c.getNome_cargo();
c.getValor_salario();
return c;
}

public boolean removeCargo(CargoEntity c){
boolean ret=true;
CargoEntity ca = new CargoEntity();
try{
ca = em.find(CargoEntity.class, c.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ca.getId() == null)
        return false;
}
try{
em.remove(ca);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
```

```
}


public boolean atualizaCargo(CargoEntity c){
boolean ret=true;
try{
em.merge(c);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

//////////////////////// FUNCIONARIO

```
	public boolean cadastreFuncionario(FuncionarioEntity funcionario){
		em.persist(funcionario);
	return true;
	}

	public FuncionarioEntity selecioneFuncionarioAtualiza(String s) {
	FuncionarioEntity f = new FuncionarioEntity();
	Query query = em.createQuery("Select f FROM FuncionarioEntity f, f.pessoa
p WHERE p.nome = :s");
	query.setParameter("s", s);
	try{
	f = (FuncionarioEntity) query.getSingleResult();
	}catch(NoResultException e){
		f.setFone_funcionario("");
	}
	f.getId();
	f.getRg();
	f.getFone_funcionario();
	f.getPis_funcionario();
	f.getClt_funcionario();
	f.getRua_funcionario();
	f.getNumero_casa();
	f.getCargo_funcionario();
	f.getData_contratacao();
```

```
f.getDepartamento();

f.getPessoa();

return f;

}


public List<String> selecioneFuncionarios() {

Query query = em.createQuery("Select p.nome FROM FuncionarioEntity f, f.pessoa p");

List<String> ufs = query.getResultList();

return ufs;

}


public List<FuncionarioEntity> selecioneFuncionariosDepart() {

Query query = em.createQuery("Select f FROM FuncionarioEntity f, f.departamento d ORDER BY d.nome_departamento");

List<FuncionarioEntity> ufs = query.getResultList();

return ufs;

}


public List<String> selecioneProfessores() {

String n= "Professor";

Query query = em.createQuery("Select p.nome FROM FuncionarioEntity f, f.cargo_funcionario c, f.pessoa p WHERE c.nome_cargo = :n");

query.setParameter("n", n);
```

```
List<String> ufs = query.getResultList();
return ufs;
}

public List<FuncionarioEntity> selecioneListaFuncionario(List<String> funcionarios_selecionadas) {
Query query = em.createQuery("Select f FROM FuncionarioEntity f, f.pessoa p WHERE p.nome in :funcionarios_selecionadas");
query.setParameter("funcionarios_selecionadas", funcionarios_selecionadas);
List<FuncionarioEntity> ufs = query.getResultList();
return ufs;
}

public boolean removeFuncionario(FuncionarioEntity u){
boolean ret=true;
DepartamentoEntity d = u.getDepartamento();
if(d.getFuncionarios().size() > 1){
d.getFuncionarios().remove(u);
u.setDepartamento(null);
try{
        em.merge(u);
}catch(java.lang.IllegalArgumentException e){System.out.println("erro d");return false;}
try{
```

```
            em.merge(d);
        }catch(java.lang.IllegalArgumentException e){System.out.println("erro
d");return false;}
            FuncionarioEntity fr = new FuncionarioEntity();
        try{
            fr = em.find(FuncionarioEntity.class, u.getId());
        }catch(java.lang.IllegalArgumentException e){System.out.println("fr b");return
false;}
        try{
            em.remove(fr);
        }catch(java.lang.IllegalArgumentException e){System.out.println("fr r");return
false;}
        }else{
            DepartamentoEntity de = new DepartamentoEntity();
        try{
            de = em.find(DepartamentoEntity.class, d.getId());
        }catch(java.lang.IllegalArgumentException e){System.out.println("fr b");return
false;}
        try{
            em.remove(de);
        }catch(java.lang.IllegalArgumentException e){System.out.println("rem
d");return false;}
        }
        return ret;
        }
```

```
public boolean atualizaFuncionario(FuncionarioEntity f){
boolean ret=true;
try{
em.merge(f);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
////////////////////  INSTITUICAO

public boolean cadastreInstituicao(InstituicaoEntity instituicao){
boolean ret=true;
Query query = em.createQuery("Select i.cnpj FROM InstituicaoEntity i");
List<String> nome_instituicoes = query.getResultList();
for(int j=0; j< nome_instituicoes.size(); j++){
String sa = nome_instituicoes.get(j);
if(sa.equals(instituicao.getCnpj()))
```

```
            ret = false;
        }
        if(ret == true)
            em.persist(instituicao);
        return ret;
        }

        /*public List<String> selecioneNomeInstituicoesDoadoras() {
        String e ="EVENTOS";
        String r="RECEITAS FINANCEIRAS";
        Query query = em.createQuery("Select i.nome_instituicao FROM InstituicaoEntity i where i.tipoInstituicao.nome <> :e AND i.tipoInstituicao.nome <> :r");
        query.setParameter("e", e);
        query.setParameter("r", r);
        List<String> ufs = query.getResultList();
        return ufs;
        }*/

        public List<String> selecioneInstituicoes() {
        Query query = em.createQuery("Select i.nome_instituicao FROM InstituicaoEntity i");
        List<String> ufs = query.getResultList();
        return ufs;
```

```
}

public List<InstituicaoEntity> selecioneListaInstituicao(List<String> parceiros_selecionadas) {
Query query = em.createQuery("Select i FROM InstituicaoEntity i WHERE i.nome_instituicao in :parceiros_selecionadas");
query.setParameter("parceiros_selecionadas", parceiros_selecionadas);
List<InstituicaoEntity> ufs = query.getResultList();
return ufs;
}

public InstituicaoEntity selecioneInstituicaoAtualiza(String s) {
InstituicaoEntity i = new InstituicaoEntity();
Query query = em.createQuery("Select i FROM InstituicaoEntity i WHERE i.cnpj = :s");
query.setParameter("s", s);
try{
i = (InstituicaoEntity) query.getSingleResult();
}catch(NoResultException e){
        i.setNome_instituicao("");
}
i.getId();
i.getNome_instituicao();
i.getCnpj();
```

```
i.getFone_instiuicao();
i.getFone_instiuicao();
i.getTipoInstituicao();
return i;
}

public boolean atualizaInstituicao(InstituicaoEntity i){
boolean ret=true;
try{
em.merge(i);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean removeInstituicao(InstituicaoEntity i){
boolean ret=true;
InstituicaoEntity in = new InstituicaoEntity();
try{
in = em.find(InstituicaoEntity.class, i.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
```

```
            if(in.getId() == null)
            return false;
        }
        try{
        em.remove(in);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

        public String getTipoInstituicao(String nome){
        String u = "";
        Query query = em.createQuery("Select n.nome FROM InstituicaoEntity i,
i.tipoInstituicao n WHERE i.nome_instituicao =:nome");
        query.setParameter("nome", nome);
        try{
            u =  (String) query.getSingleResult();
        }catch(NoResultException e){
        return "nao";
        }
        return u;
        }
```

```
//////////////////////TIPO INSTITUICAO//////////////////////


public boolean cadastreTipoInstituicao(TipoInstituicaoEntity tipoInstituicao){
boolean ret=true;
Query query = em.createQuery("Select t.nome FROM TipoInstituicaoEntity t");
List<String> nome_instituicoes = query.getResultList();
for(int j=0; j< nome_instituicoes.size(); j++){
String s = nome_instituicoes.get(j);
if(s.equals(tipoInstituicao.getNome()))
        return false;
}
if(ret == true)
        em.persist(tipoInstituicao);
return ret;
}


public List<String> selecioneTipoInstituicao() {
Query query = em.createQuery("Select t.nome FROM TipoInstituicaoEntity t");
List<String> ufs = query.getResultList();
return ufs;
```

```
        }


        public TipoInstituicaoEntity selecioneTipoInstituicaoAtualiza(String s) {
        TipoInstituicaoEntity t = new TipoInstituicaoEntity();
        Query query = em.createQuery("Select t FROM TipoInstituicaoEntity t
WHERE t.nome = :s");
        query.setParameter("s", s);
        try{
        t = (TipoInstituicaoEntity) query.getSingleResult();
        }catch(NoResultException e){
                t.setNome("");
        }
        t.getId();
        t.getNome();
        t.getIntituicoes();
        return t;
        }


        public boolean removeTipoInstituicao(TipoInstituicaoEntity t){
        boolean ret=true;
        String s = t.getNome();
        TipoInstituicaoEntity ti = new TipoInstituicaoEntity();
        Query query = em.createQuery("Select t FROM TipoInstituicaoEntity t
WHERE t.nome = :s");
```

```
query.setParameter("s", s);
try{
ti =  (TipoInstituicaoEntity) query.getSingleResult();
}catch(NoResultException e){
        return false;
}
try{
em.remove(ti);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaTipoInstituicao(TipoInstituicaoEntity t){
boolean ret=true;
try{
em.merge(t);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
        ////////////////// DOADOR


  public boolean cadastreDoador(DoadorEntity doador){
            em.persist(doador);
        return true;
        }

        public List<String> selecioneDoadores() {
        Query query = em.createQuery("Select p.nome FROM DoadorEntity d, d.pessoa p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public List<String> selecioneDoadoresComDoacoes() {
        Query query = em.createQuery("Select DISTINCT p.nome FROM DoacaoEntity d, d.doador o, o.pessoa p");
        List<String> ufs = query.getResultList();
```

```
return ufs;
}

public DoadorEntity selecioneDoadorAtualiza(String s) {
DoadorEntity d = new DoadorEntity();
Query query = em.createQuery("Select o FROM DoadorEntity o, o.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
try{
d = (DoadorEntity) query.getSingleResult();
}catch(NoResultException e){
        d.setFone_doador("");
}
d.getId();
d.getPessoa();
d.getFone_doador();
d.getEmail_doador();
return d;
}

public boolean removeDoador(DoadorEntity d){
boolean ret=true;
DoadorEntity dr = new DoadorEntity();
try{
```

```
dr = em.find(DoadorEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(dr.getId() == null)
        return false;
}
try{
em.remove(dr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaDoador(DoadorEntity d){
boolean ret=true;
try{
em.merge(d);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
//////////////////////TIPO BENEFICIO//////////////////////


public boolean cadastreTipoBeneficio(TipoBeneficioEntity tipobeneficio){
boolean ret=true;
Query query = em.createQuery("Select t.nome_beneficio FROM TipoBeneficioEntity t");
List<String> nome_beneficios = query.getResultList();
for(int j=0; j< nome_beneficios.size(); j++){
String s = nome_beneficios.get(j);
if(s.equals(tipobeneficio.getNome_beneficio()))
        return false;
}
if(ret == true)
        em.persist(tipobeneficio);
return ret;
}
```

```
public List<String> selecioneTipoNomeBeneficioConcedidos() {
Query query = em.createQuery("Select t.nome_beneficio FROM TipoBeneficioEntity t");
List<String> ufs = query.getResultList();
return ufs;
}

public TipoBeneficioEntity selecioneTipoBeneficioAtualiza(String s) {
TipoBeneficioEntity t = new TipoBeneficioEntity();
Query query = em.createQuery("Select t FROM TipoBeneficioEntity t WHERE t.nome_beneficio = :s");
query.setParameter("s", s);
try{
t = (TipoBeneficioEntity) query.getSingleResult();
}catch(NoResultException e){
        t.setNome_beneficio("");
}
t.getId();
t.getNome_beneficio();
return t;
}

public boolean removeTipoBeneficio(TipoBeneficioEntity t){
boolean ret=true;
```

```
        String s = t.getNome_beneficio();
        TipoBeneficioEntity ti = new TipoBeneficioEntity();
        Query query = em.createQuery("Select t FROM TipoBeneficioEntity t WHERE
t.nome_beneficio = :s");
        query.setParameter("s", s);
        try{
        ti =  (TipoBeneficioEntity) query.getSingleResult();
        }catch(NoResultException e){
                return false;
        }
        try{
        em.remove(ti);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

        public boolean atualizaTipoBeneficio(TipoBeneficioEntity t){
        boolean ret=true;
        try{
        em.merge(t);
        }catch(java.lang.IllegalArgumentException e){
        return false;
```

```
        }
        return ret;
        }

        ////////////////////    TIPO DOAÇAO

  public boolean cadastreTipoDoacao(TipoDoacaoEntity tipodoacao){
        boolean ret=true;
        Query query = em.createQuery("Select t.nome_doacao FROM
TipoDoacaoEntity t");
        List<String> nome_doacoes = query.getResultList();
        for(int j=0; j< nome_doacoes.size(); j++){
        String s = nome_doacoes.get(j);
        if(s.equals(tipodoacao.getNome_doacao()))
                return false;
        }
```

```
        if(ret == true)
                em.persist(tipodoacao);
        return ret;

}
        public List<String> selecioneTipoDoacoes() {
        Query query = em.createQuery("Select t.nome_doacao FROM TipoDoacaoEntity t");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public TipoDoacaoEntity selecioneTipoDoacaoAtualiza(String s) {
        TipoDoacaoEntity t = new TipoDoacaoEntity();
        Query query = em.createQuery("Select t FROM TipoDoacaoEntity t WHERE t.nome_doacao = :s");
        query.setParameter("s", s);
        try{
        t = (TipoDoacaoEntity) query.getSingleResult();
        }catch(NoResultException e){
                t.setNome_doacao("");
        }
        t.getId();
        t.getNome_doacao();
```

```
return t;
}

public boolean removeTipoDoacao(TipoDoacaoEntity t){
boolean ret=true;
String s = t.getNome_doacao();
TipoDoacaoEntity ti = new TipoDoacaoEntity();
Query query = em.createQuery("Select t FROM TipoDoacaoEntity t WHERE t.nome_doacao = :s");
query.setParameter("s", s);
try{
ti =  (TipoDoacaoEntity) query.getSingleResult();
}catch(NoResultException e){
        return false;
}
try{
em.remove(ti);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaTipoDoacao(TipoDoacaoEntity t){
```

```
boolean ret=true;
try{
em.merge(t);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

//////////////////////BeneficioConcedido//////////////////////

public boolean cadastreBeneficioConcedido(BeneficioConcedidoEntity beneficio){
boolean ret=true;
        em.persist(beneficio);
return ret;
}

public List<String> selecioneBeneficioConcedidos() {
```

```
Query query = em.createQuery("Select b.nome_beneficio FROM BeneficioConcedidoEntity b");

List<String> ufs = query.getResultList();

return ufs;

}


public List<BeneficioConcedidoEntity> getBeneficioConcedidos(String s, Date ini, Date fin){

Query query = em.createQuery("Select b FROM BeneficioConcedidoEntity b, b.beneficiario e WHERE e.pessoa.nome = :s AND b.data_beneficio BETWEEN :ini AND :fin");

query.setParameter("s", s);

query.setParameter("ini",ini , TemporalType.DATE);

query.setParameter("fin",fin , TemporalType.DATE);

List<BeneficioConcedidoEntity> d = (List<BeneficioConcedidoEntity>) query.getResultList();

return d;

}


public List<BeneficioConcedidoEntity> getTodasBeneficioConcedidos(Date ini, Date fin){

Query query = em.createQuery("Select b FROM BeneficioConcedidoEntity b WHERE b.data_beneficio BETWEEN :ini AND :fin");

query.setParameter("ini",ini , TemporalType.DATE);

query.setParameter("fin",fin , TemporalType.DATE);
```

```
        List<BeneficioConcedidoEntity> d = query.getResultList();
        return d;
        }

        public BeneficioConcedidoEntity selecioneBeneficioConcedidoAtualiza(String
s) {
        BeneficioConcedidoEntity d = new BeneficioConcedidoEntity();
        Query query = em.createQuery("Select b FROM BeneficioConcedidoEntity b
WHERE b.nome_beneficio = :s");
        query.setParameter("s", s);
        try{
        d = (BeneficioConcedidoEntity) query.getSingleResult();
        }catch(NoResultException e){
                d.getNome_beneficio().setNome_beneficio("");
        }
        d.getId();
        d.getNome_beneficio();
        d.getBeneficiario();
        d.getData_beneficio();
        d.getValor_beneficio();
        return d;
        }

        public boolean removeBeneficioConcedido(BeneficioConcedidoEntity b){
```

```
boolean ret=true;
BeneficioConcedidoEntity da = new BeneficioConcedidoEntity();
try{
da = em.find(BeneficioConcedidoEntity.class, b.getId());
}catch(java.lang.IllegalArgumentException e){
        return false;
}
try{
em.remove(da);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaBeneficioConcedido(BeneficioConcedidoEntity b){
boolean ret=true;
try{
em.merge(b);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
```

```
        }



        ////////////////////            DOAÇAO


  public boolean cadastreDoacao(DoacaoEntity doacao){
        boolean ret=true;
                em.persist(doacao);
        return ret;
        }

        public List<String> selecioneDoacoes() {
        Query query = em.createQuery("Select d.nome_doacao FROM DoacaoEntity
d");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public List<DoacaoEntity> getDoacoes(String s, Date ini, Date fin){
        Query query = em.createQuery("Select d FROM DoacaoEntity d, d.doador o,
o.pessoa p WHERE p.nome = :s AND d.data_doacao BETWEEN :ini AND :fin");
        query.setParameter("s", s);
```

```
query.setParameter("ini",ini , TemporalType.DATE);
query.setParameter("fin",fin , TemporalType.DATE);
List<DoacaoEntity> d = (List<DoacaoEntity>) query.getResultList();
return d;
}

public List<DoacaoEntity> getTodasDoacoes( Date ini, Date fin){
Query query = em.createQuery("Select d FROM DoacaoEntity d WHERE d.data_doacao BETWEEN :ini AND :fin");
query.setParameter("ini",ini , TemporalType.DATE);
query.setParameter("fin",fin , TemporalType.DATE);
List<DoacaoEntity> d = query.getResultList();
return d;
}

public DoacaoEntity selecioneDoacaoAtualiza(String s) {
DoacaoEntity d = new DoacaoEntity();
Query query = em.createQuery("Select d FROM DoacaoEntity d WHERE d.nome_doacao = :s");
query.setParameter("s", s);
try{
d = (DoacaoEntity) query.getSingleResult();
}catch(NoResultException e){
        d.getNome_doacao().setNome_doacao("");
```

```
}
d.getId();
d.getNome_doacao();
d.getDoador();
d.getData_doacao();
d.getValor_doacao();
return d;
}

public boolean removeDoacao(DoacaoEntity d){
boolean ret=true;
DoacaoEntity da = new DoacaoEntity();
try{
da = em.find(DoacaoEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(da.getId() == null)
        return false;
}
try{
em.remove(da);
}catch(java.lang.IllegalArgumentException e){
return false;
```

```
        }
        return ret;
        }

        public boolean atualizaDoacao(DoacaoEntity d){
        boolean ret=true;
        try{
        em.merge(d);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

        ////////////////////        PARCEIRO

public boolean cadastreParceiro(ContribuinteEntity parceiro){
        boolean ret=true;
                em.persist(parceiro);
```

```
return ret;
}

public List<String> selecioneParceiros() {
Query query = em.createQuery("Select p.nome FROM ContribuinteEntity c, c.pessoa p");
List<String> ufs = query.getResultList();
return ufs;
}

public List<ContribuinteEntity> selecioneListaParceiro(List<String> parceiros_selecionadas) {
Query query = em.createQuery("Select p FROM ContribuinteEntity c, c.pessoa p WHERE p.nome in :parceiros_selecionadas");
query.setParameter("parceiros_selecionadas", parceiros_selecionadas);
List<ContribuinteEntity> ufs = query.getResultList();
return ufs;
}

public ContribuinteEntity selecioneParceiroAtualiza(String s) {
ContribuinteEntity pa = new ContribuinteEntity();
Query query = em.createQuery("Select e FROM ContribuinteEntity e, e.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
```

```
try{
pa = (ContribuinteEntity) query.getSingleResult();
}catch(NoResultException e){
        pa.setFone("");
}
pa.getId();
pa.getPessoa();
pa.getFone();
pa.getEmail();
return pa;
}

public boolean removeParceiro(ContribuinteEntity p){
boolean ret=true;
ContribuinteEntity pa = new ContribuinteEntity();
try{
pa = em.find(ContribuinteEntity.class, p.getId());
}catch(java.lang.IllegalArgumentException e){
        if(pa.getId() == null)
        return false;
}
try{
em.remove(pa);
```

```
        }catch(EJBException e){
        e.getMessage();
        }
        return ret;
        }









        ////////////////////            APOIADOR



public boolean cadastreApoiador(ApoiadorEntity apoiador){
                em.persist(apoiador);
        return true;
        }

        public List<String> selecioneApoiadores() {
```

```
        Query query = em.createQuery("Select p.nome FROM ApoiadorEntity a, a.pessoa p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public ApoiadorEntity selecioneApoiadorAtualiza(String s) {
        ApoiadorEntity a = new ApoiadorEntity();
        Query query = em.createQuery("Select a FROM ApoiadorEntity a, a.pessoa p WHERE p.nome = :s");
        query.setParameter("s", s);
        try{
        a = (ApoiadorEntity) query.getSingleResult();
        }catch(NoResultException e){
                a.setFone_apoiador("");
        }
        a.getId();
        a.getFone_apoiador();
        a.getEmail_apoiador();
        a.getPessoa();
        return a;
        }

        public boolean removeApoiador(ApoiadorEntity a){
```

```
boolean ret=true;
ApoiadorEntity ap = new ApoiadorEntity();
try{
ap = em.find(ApoiadorEntity.class, a.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(ap.getId() == null)
        return false;
}
try{
em.remove(ap);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaApoiador(ApoiadorEntity a){
boolean ret=true;
try{
em.merge(a);
}catch(java.lang.IllegalArgumentException e){
return false;
```

```
        }
        return ret;
        }




        ////////////////////            DEPARTAMENTO



  public boolean cadastreDepartamento(DepartamentoEntity departamento){
        boolean ret=true;
        Query query = em.createQuery("Select d.nome_departamento FROM
DepartamentoEntity d");
        //try{
        List<String> nome_departamentos = query.getResultList();
        //}catch(){
        //}
        for(int j=0; j< nome_departamentos.size(); j++){
        String sa = nome_departamentos.get(j);
        if(sa.equals(departamento.getNome_departamento()))
                ret = false;
```

```
        }
        if(ret == true)
                em.persist(departamento);
        return ret;
        }

        public List<String> selecioneDepartamentos() {
        Query query = em.createQuery("Select d.nome_departamento FROM
DepartamentoEntity d");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public DepartamentoEntity selecioneDepartamentoAtualiza(String s) {
        DepartamentoEntity d = new DepartamentoEntity();
        Query query = em.createQuery("Select d FROM DepartamentoEntity d
WHERE d.nome_departamento = :s");
        query.setParameter("s", s);
        try{
        d = (DepartamentoEntity) query.getSingleResult();
        }catch(NoResultException e){
                d.setNome_departamento("");
        }
        d.getId();
```

```
d.getNome_departamento();
d.getFuncionarios();
return d;
}

public boolean removeDepartamento(DepartamentoEntity d){
boolean ret=true;
DepartamentoEntity pr = new DepartamentoEntity();
try{
pr = em.find(DepartamentoEntity.class, d.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
public boolean atualizaDepartamento(DepartamentoEntity d){
boolean ret=true;
try{
em.merge(d);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
////////////////                TURMA

public boolean cadastreTurma(TurmaEntity turma){
boolean ret=true;
Query query = em.createQuery("Select t.nome_turma FROM TurmaEntity t");
List<String> nome_turmas = query.getResultList();
for(int j=0; j< nome_turmas.size(); j++){
String sa = nome_turmas.get(j);
if(sa.equals(turma.getNome_turma()))
```

```
            ret = false;
        }
        if(ret == true)
            em.persist(turma);
        return ret;
        }

        public List<String> selecioneTurmas() {
        Query query = em.createQuery("Select t.nome_turma FROM TurmaEntity t");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public List<String> selecioneTurmasComFrequencia() {
        Query query = em.createQuery("Select DISTINCT f.nome_chamada  FROM
FrequenciaEntity f");
        List<String> ufs = query.getResultList();

        Query query2 = em.createQuery("Select t.nome_turma FROM TurmaEntity t,
t.chamada c where c.nome_chamada in :ufs");
        query2.setParameter("ufs", ufs);
        List<String> turmas = query2.getResultList();
        return turmas;
        }
```

```
public List<String> selecioneTurmasPorAtividades(String nome_atividades) {

Query query = em.createQuery("Select a.nome_turma FROM TurmaEntity a WHERE a.atividade.nome_atividade = :nome_atividades");

query.setParameter("nome_atividades", nome_atividades);

List<String> ufs = query.getResultList();

return ufs;

}

public List<String> selecioneAlunosPorTurma(String nome_turma) {

Query query = em.createQuery("Select p.nome FROM TurmaEntity a JOIN a.alunos s, s.pessoa p WHERE a.nome_turma = :nome_turma");

query.setParameter("nome_turma", nome_turma);

List<String> ufs = query.getResultList();

return ufs;

}

public List<TurmaEntity> selecioneListaTurma(List<String> turmas_selecionadas) {

Query query = em.createQuery("Select a FROM TurmaEntity a WHERE a.nome_turma in :turmas_selecionadas");

query.setParameter("turmas_selecionadas", turmas_selecionadas);

List<TurmaEntity> ufs = query.getResultList();

return ufs;
```

```
}

public String selecioneChamada(String turma_selecionada){
Query query = em.createQuery("Select a.chamada.nome_chamada FROM TurmaEntity a WHERE a.nome_turma = :turma_selecionada");
query.setParameter("turma_selecionada", turma_selecionada);
String c = (String) query.getSingleResult();
return c;
}

public List<String> selecioneTurmasDesistente(String nome_aluno) {
Query query = em.createQuery("Select t.nome_turma FROM TurmaEntity t JOIN t.alunos a, a.pessoa p WHERE p.nome = :nome_aluno");
query.setParameter("nome_aluno", nome_aluno);
List<String> turmas = query.getResultList();
return turmas;
}

public List<String> selecioneTurmasPorTurno(String turno,String nome_aluno) {
List<String> turmas = new ArrayList<>();
Query query = em.createQuery("Select t FROM TurmaEntity t WHERE t.turno = :turno");
query.setParameter("turno", turno);
```

```
List<TurmaEntity> ufs = query.getResultList();
for(int i=0; i< ufs.size(); i++){
        List<AlunoEntity> no_alunos = ufs.get(i).getAlunos();
        for(int j=0; j< no_alunos.size(); j++){
        if(no_alunos.get(j).getPessoa().getNome().equals(nome_aluno))
        turmas.add(ufs.get(i).getNome_turma());
        }
}
return turmas;
}

public List<String> selecioneNomesAlunosTurmas(String s){
Query query = em.createQuery("select p.nome from TurmaEntity t, t.alunos a,
a.pessoa p, t.chamada c where c.nome_chamada = :s");
  query.setParameter("s", s).getResultList();
        List<String> ufs = query.getResultList();
return ufs;
}

public TurmaEntity selecioneTurmaAtualiza(String s) {
TurmaEntity t = new TurmaEntity();
Query query = em.createQuery("Select t FROM TurmaEntity t WHERE
t.nome_turma = :s");
query.setParameter("s", s);
```

```
try{
t = (TurmaEntity) query.getSingleResult();
}catch(NoResultException e){
        if(t.getTurno()==null)
        t.setTurno("");
        else
        t.setNome_turma("");
}
t.getId_turma();
t.getNome_turma();
t.getNome_professor();
t.getData_criacao();
t.getAlunos();
t.getChamada();
t.getTurno();
t.getAtividade();
return t;
}

public boolean atualizaTurma(TurmaEntity t){
boolean ret=true;
try{
em.merge(t);
```

```
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

        public boolean removeTurma(TurmaEntity t){
        boolean ret=true;
        TurmaEntity tu = new TurmaEntity();
        try{
                tu = em.find(TurmaEntity.class, t.getId_turma());
        }catch(java.lang.IllegalArgumentException e){
                System.out.println("erro tu f");
                return false;
        }
        try{
        em.remove(tu);
        }catch(java.lang.IllegalArgumentException e){
        System.out.println("erro tu r");
                return false;
        }
        return ret;
}
```

```
//////////////////              ATIVIDADE


public boolean cadastreAtividade(AtividadeEntity atividade){
boolean ret=true;
Query query = em.createQuery("Select a.nome_atividade FROM AtividadeEntity a");
List<String> nome_atividades = query.getResultList();
for(int j=0; j< nome_atividades.size(); j++){
String sa = nome_atividades.get(j);
if(sa.equals(atividade.getNome_atividade()))
        ret = false;
}
if(ret == true)
em.persist(atividade);

return ret;
}
```

```
public AtividadeEntity selecioneAtividadeAtualiza(String s) {
AtividadeEntity a = new AtividadeEntity();
Query query = em.createQuery("Select s FROM AtividadeEntity s WHERE s.nome_atividade = :s");
query.setParameter("s", s);
try{
a = (AtividadeEntity) query.getSingleResult();
}catch(NoResultException e){
        a.setNome_atividade("");
}
a.getId();
a.getNome_atividade();
a.getNome_turmas();
a.getData_criacao();
a.getProjeto();
return a;
}

public List<String> selecioneAtividades() {
Query query = em.createQuery("Select a.nome_atividade FROM AtividadeEntity a");
List<String> ufs = query.getResultList();
return ufs;
}
```

```
/*public List<String> selecioneTurmasPorAtividades(String nome_atividades) {

Query query = em.createQuery("Select a.nome_turmas FROM AtividadeEntity a JOIN a.nome_turmas t WHERE t.nome_turma in :nome_atividades");

query.setParameter("nome_atividades", nome_atividades);

List<String> ufs = query.getResultList();

return ufs;

}*/

public List<AtividadeEntity> selecioneListaAtividade(List<String> ativis_selecionadas) {

Query query = em.createQuery("Select a FROM AtividadeEntity a WHERE a.nome_atividade in :ativis_selecionadas");

query.setParameter("ativis_selecionadas", ativis_selecionadas);

List<AtividadeEntity> ufs = query.getResultList();

return ufs;

}

/*public List<String> selecioneAtividadesDesistente(String s) {

List<String> atividades = new ArrayList<>();

Query query = em.createQuery("Select a FROM AtividadeEntity a");

List<AtividadeEntity> ufs = query.getResultList();

for(int i=0; i< ufs.size(); i++){
```

```
            List<TurmaEntity> no_turmas = ufs.get(i).getNome_turmas();
            for(int j=0; j< no_turmas.size(); j++){
            if(no_turmas.get(j).getNome_turma().equals(s))
            atividades.add(ufs.get(i).getNome_atividade());
            }
        }
        return atividades;
        }*/

        public List<String> selecioneAtividadesDesistente(String s) {
        Query query = em.createQuery("Select a.nome_atividade FROM AtividadeEntity a JOIN a.nome_turmas t WHERE t.nome_turma = :s");
        query.setParameter("s", s);
        List<String> atividades = query.getResultList();
        return atividades;
        }

        public boolean verificaTurmaCadastrado(List<String> nome_turmas){
        if(nome_turmas.size() > 0){
        Query query = em.createQuery("Select a FROM AtividadeEntity a");
        List<AtividadeEntity> a = query.getResultList();
        for(int i=0; i<a.size(); i++){
        List<TurmaEntity> tu = a.get(i).getNome_turmas();
        for(int j=0; j<tu.size(); j++){
```

```
        for(int x=0; x<nome_turmas.size(); x++){
        if(tu.get(j).getNome_turma().equals(nome_turmas.get(x)))
                return false;
        }
}
}
}else{
        return true;
}
return true;
}

public boolean removeAtividade(AtividadeEntity a){
boolean ret=true;
AtividadeEntity p = new AtividadeEntity();
try{
p = em.find(AtividadeEntity.class, a.getId());
}catch(java.lang.IllegalArgumentException e){
        if(p.getId() == null)
        return false;
}
try{
em.remove(p);
```

```
        }catch(java.lang.IllegalArgumentException e){
                if(p.getId() == null)
                return false;
        }
        return ret;
        }

        public boolean atualizaAtividade(AtividadeEntity a){
        boolean ret=true;
        try{
        em.merge(a);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

        ////////////////// PROJETO


   public boolean cadastreProjeto(ProjetoEntity projeto){
        boolean ret=true;
        Query query = em.createQuery("Select p.nome_projeto FROM ProjetoEntity
p");
```

```
        List<String> nome_projetos = query.getResultList();
        for(int j=0; j< nome_projetos.size(); j++){
        String sa = nome_projetos.get(j);
        if(sa.equals(projeto.getNome_projeto()))
                ret = false;
        }
        if(ret == true)
                em.persist(projeto);
        return ret;
        }

public boolean verificaAtividadeCadastrada(List<String> nome_atividades){
        if(nome_atividades.size() > 0){
        Query query = em.createQuery("Select p FROM ProjetoEntity p");
        List<ProjetoEntity> ps = query.getResultList();
        for(int i=0; i<ps.size(); i++){
        List<AtividadeEntity> as = ps.get(i).getNome_atividades();
        for(int j=0; j<as.size(); j++){
                for(int x=0; x<nome_atividades.size(); x++){
                if(as.get(j).getNome_atividade().equals(nome_atividades.get(x)))
                        return false;
                }
        }
```

```
        }
        }else{
                return true;
        }
        return true;
  }

        public List<String> selecioneProjetos() {
        Query query = em.createQuery("Select p.nome_projeto FROM ProjetoEntity p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public List<ProjetoEntity> selecioneListaProjeto(List<String> projeto_selecionadas) {
        Query query = em.createQuery("Select a FROM ProjetoEntity a WHERE a.nome_projeto in :projeto_selecionadas");
        query.setParameter("projeto_selecionadas", projeto_selecionadas);
        List<ProjetoEntity> ufs = query.getResultList();
        return ufs;
        }

        public ProjetoEntity selecioneProjetoAtualiza(String s) {
```

```
ProjetoEntity p = new ProjetoEntity();
Query query = em.createQuery("Select p FROM ProjetoEntity p WHERE p.nome_projeto = :s");
query.setParameter("s", s);
try{
p = (ProjetoEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setNome_projeto("");
}
p.getId();
p.getNome_projeto();
p.getData_inicio();
p.getNome_atividades();
p.getInstituicoes();
p.getCust_anual();
return p;
}

public boolean removeProjeto(ProjetoEntity pro){
boolean ret=true;
ProjetoEntity pr = new ProjetoEntity();
try{
pr = em.find(ProjetoEntity.class, pro.getId());
}catch(java.lang.IllegalArgumentException e){
```

```
e.getMessage();
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaProjeto(ProjetoEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
//////////////////              PROGRAMA



  public boolean cadastrePrograma(ProgramaEntity programa){
        boolean ret=true;
        Query query = em.createQuery("Select p.nome_programa FROM
ProgramaEntity p");
        List<String> nome_programas = query.getResultList();
        for(int j=0; j< nome_programas.size(); j++){
        String sa = nome_programas.get(j);
        if(sa.equals(programa.getNome_programa()))
                return false;
        }
        if(ret == true)
                em.persist(programa);


        em.persist(programa);
        return ret;
        }


        public List<String> selecioneProgramas() {
```

```
        Query query = em.createQuery("Select p.nome_programa FROM
ProgramaEntity p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public ProgramaEntity selecioneProgramaAtualiza(String s) {
        ProgramaEntity p = new ProgramaEntity();
        Query query = em.createQuery("Select p FROM ProgramaEntity p WHERE
p.nome_programa = :s");
        query.setParameter("s", s);
        try{
        p = (ProgramaEntity) query.getSingleResult();
        }catch(NoResultException e){
                p.setNome_programa("");
        }
        p.getId();
        p.getNome_programa();
        p.getData_criacao();
        p.getNome_projetos();
        p.getDescricao_programa();
        return p;
        }
```

```
public boolean removePrograma(ProgramaEntity po){
boolean ret=true;
ProgramaEntity pr = new ProgramaEntity();
try{
pr = em.find(ProgramaEntity.class, po.getId());
}catch(java.lang.IllegalArgumentException e){
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean verificaProjetoCadastrado(List<String> nome_projetos){
if(nome_projetos.size() > 0){
Query query = em.createQuery("Select p FROM ProgramaEntity p");
List<ProgramaEntity> ps = query.getResultList();
for(int i=0; i<ps.size(); i++){
List<ProjetoEntity> as = ps.get(i).getNome_projetos();
```

```
for(int j=0; j<as.size(); j++){
        for(int x=0; x<nome_projetos.size(); x++){
        if(as.get(j).getNome_projeto().equals(nome_projetos.get(x)))
                return false;
        }
}
}
}else{
        return true;
}
return true;
}

public boolean atualizaPrograma(ProgramaEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

```
        ////////////////////          DADASTRO SOCIAL



  public boolean cadastreSocial(SocialEntity social){
                em.persist(social);
        return true;
        }

        public List<String> selecioneSociais() {
        Query query = em.createQuery("Select p.nome FROM SocialEntity s, s.pessoa p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public List<String> selecioneSociaisComBeneficios() {
        Query query = em.createQuery("Select DISTINCT p.nome FROM BeneficioConcedidoEntity b, b.beneficiario e, e.pessoa p");
        List<String> ufs = query.getResultList();
        return ufs;
        }

        public SocialEntity selecioneSocialAtualiza(String s) {
```

```
SocialEntity i = new SocialEntity();
Query query = em.createQuery("Select o FROM SocialEntity o, o.pessoa p WHERE p.nome = :s");
query.setParameter("s", s);
try{
i = (SocialEntity) query.getSingleResult();
}catch(NoResultException e){
        i.setFone_social("");
}
i.getId();
i.getPessoa();
i.getRg_social();
i.getFone_social();
i.getProfissao_social();
i.getSituacao_social();
i.getRua_social();
i.getNumero_casa();
i.getComplemento_rua();
i.getAuxilio_governo();
i.getGestante();
i.getObs_gestante();
i.getAcamado();
i.getObs_acamado();
i.getDeficiente();
```

```
i.getObs_deficiente();
i.getNro_adultos();
i.getNro_joves();
i.getNro_adolecentes();
i.getNro_criancas();
i.getNro_bebes();
i.getGrau_instrucao();
i.getRenda_mensal_familiar();
i.getData_cadastro();
return i;
}

public boolean removeSocial(SocialEntity s){
boolean ret=true;
SocialEntity so = new SocialEntity();
try{
so = em.find(SocialEntity.class, s.getId());
}catch(java.lang.IllegalArgumentException e){
e.getMessage();
        if(so.getId() == null)
        return false;
}
try{
```

```
em.remove(so);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}

public boolean atualizaSocial(SocialEntity s){
boolean ret=true;
try{
em.merge(s);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}
```

////////////////// CADASTRO PAPEL

```
public boolean cadastrePapel(PapelEntity papel){
        em.persist(papel);
```

```
return true;
}

public PapelEntity selecionePapelAtualiza(String s) {
PapelEntity p = new PapelEntity();
Query query = em.createQuery("Select p FROM PapelEntity p, p.pessoa a WHERE a.nome = :s");
query.setParameter("s", s);
try{
p = (PapelEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setLogin("");
}
p.getId();
p.getPessoa();
p.getLogin();
p.getSenha();
p.getPapel();
return p;
}

public boolean removePapel(PapelEntity po){
boolean ret=true;
PapelEntity pr = new PapelEntity();
```

```
try{
pr = em.find(PapelEntity.class, po.getId());
}catch(java.lang.IllegalArgumentException e){
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
}



public boolean atualizaPapel(PapelEntity p){
boolean ret=true;
try{
em.merge(p);
}catch(java.lang.IllegalArgumentException e){
return false;
}
return ret;
```

```
        }


        public String verificarLogin(String login, String senha){
        String papel="";
        Query query = em.createQuery("Select p.papel FROM PapelEntity p WHERE
p.login = :login AND p.senha =:senha");
        query.setParameter("login", login);
        query.setParameter("senha", senha);
        try{
        papel = (String) query.getSingleResult();
        }catch(NoResultException e){
                return papel;
        }
        return papel;
        }


        public List<PapelEntity> getPapel(){
        List<PapelEntity> papel = new ArrayList<>();
        Query query = em.createQuery("Select p FROM PapelEntity p");
        try{
        papel = query.getResultList();
        }catch(NoResultException e){
                return papel;
        }
```

```
        return papel;
        }




        ////////////////////              CADASTRO PAPEL


public boolean cadastrePessoa(PessoaEntity pessoa){
        boolean ret=true;
        PessoaEntity p = new PessoaEntity();
        Query query = em.createQuery("Select p.cpf FROM PessoaEntity p");
        List<String> cpfs = query.getResultList();
        for(int j=0; j< cpfs.size(); j++){
        if(cpfs.get(j).equals(pessoa.getCpf())){
                ret = false;
                break;
        }
        }
        if(ret == true)
                em.persist(pessoa);
        return ret;
        }
```

```
public PessoaEntity selecionePessoaAtualiza(String s) {
PessoaEntity p = new PessoaEntity();
Query query = em.createQuery("Select p FROM PessoaEntity p WHERE p.nome = :s");
query.setParameter("s", s);
try{
p = (PessoaEntity) query.getSingleResult();
}catch(NoResultException e){
        p.setNome("");
}
p.getId();
p.getNome();
p.getCpf();
p.getData_nasc();
return p;
}

public PessoaEntity selecionePessoaCPFAtualiza(String cpf) {
PessoaEntity p = new PessoaEntity();
Query query = em.createQuery("Select p FROM PessoaEntity p WHERE p.cpf = :cpf");
query.setParameter("cpf", cpf);
try{
p = (PessoaEntity) query.getSingleResult();
```

```
}catch(NoResultException e){
        p.setNome("");
}
p.getId();
p.getNome();
p.getCpf();
p.getData_nasc();
return p;
}

public boolean removePessoa(PessoaEntity po){
boolean ret=true;
PessoaEntity pr = new PessoaEntity();
try{
pr = em.find(PessoaEntity.class, po.getId());
}catch(java.lang.IllegalArgumentException e){
        if(pr.getId() == null)
        return false;
}
try{
em.remove(pr);
}catch(java.lang.IllegalArgumentException e){
return false;
```

```
        }
        return ret;
        }


        public boolean atualizaPessoa(PessoaEntity p){
        boolean ret=true;
        try{
        em.merge(p);
        }catch(java.lang.IllegalArgumentException e){
        return false;
        }
        return ret;
        }

}
```