UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO CURSO DE SISTEMAS DE INFORMAÇÃO

MARCELO TEIXEIRA CORTES

StudyNizer - Plataforma web e aplicativo para ajudar alunos a organizarem seus estudos

MARCELO TEIXEIRA CORTES

StudyNizer - Plataforma web e aplicativo para ajudar alunos a organizarem seus estudos

Trabalho de conclusão de curso submetido ao curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação. Orientador: Prof. José Eduardo De Lucca

Florianópolis

2022

AGRADECIMENTOS

Quero expressar minha mais profunda gratidão à minha mãe, Marilda Teixeira Cortes e tudo o que fez por mim durante todos esses anos. Seu constante apoio e incentivo foram fundamentais para que eu pudesse chegar até aqui, finalizando meu trabalho de conclusão de curso. Sei que você sempre esteve presente, me apoiando e acreditando em mim, mesmo nos momentos mais difíceis. Suas palavras de encorajamento e seu amor incondicional foram o que me manteve motivado para seguir em frente e alcançar esse objetivo. Não tenho palavras para expressar minha gratidão por tudo o que você fez por mim. Você é uma pessoa incrível e eu sou muito sortudo por ter uma mãe como você. Obrigado por tudo!

“Nunca, jamais, desanimeis,
embora venham ventos
contrários.”
(Santa Madre Paulina)

RESUMO

Durante a trajetória universitária muitos estudantes podem encontrar dificuldades em suas atividades cotidianas. Com o acúmulo de tarefas, provas e trabalhos é de exímia importância manter uma rotina de organização, para que assim, seja possível obter um bom desempenho acadêmico. No presente trabalho é proposto o desenvolvimento de uma plataforma web e aplicativo, esta visa auxiliar estudantes universitários a organizarem suas tarefas, de diferentes disciplinas, por meio de conceitos que estão relacionados a utilização de metodologias ágeis. A aplicação possui um painel em que é possível registrar e acompanhar o progresso de cada tarefa atribuída, juntamente com o registro de resumos, esquemas ou outras estratégias, que visam auxiliar o estudo dos conteúdos presentes nas disciplinas e na execução das tarefas. Será possível integrar eventos com o sistema de calendários e compartilhar resumos com outros estudantes.

Palavras-chave: Organização, Metodologias ágeis, Tarefas, Estudantes

ABSTRACT

During the university journey, many students may encounter difficulties in their daily activities. With the accumulation of tasks, tests and assignments, it is extremely important to maintain an organizational routine, so that way, the student can obtain a good academic performance. In this final paper, the development of a web platform and application is proposed, which aims to help students to organize their tasks, from different disciplines, through concepts that are related to the use of agile methodologies. The application has a panel in which it is possible to record and follow the progress of each assigned task, along with the registration of summaries, schemes or other strategies, that aim to help the study of the subjects present in the disciplines and in the execution of the tasks. It will be possible to integrate events with the calendar system and share summaries with other students.

Palavras-chave: Organization, Agile methodologies, Tasks, Students

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UX	User experience
UI	User interface
UFSC	Universidade Federal de Santa Catarina
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
SQL	Structured Query Language
SPA	Single page application
REST	Representational State Transfer
JSON	JavaScript Object Notation
MVP	Minimum Viable Product
JWT	JSON Web Token
HTTP	Hypertext Transfer Protocol
CRUD	Create, Read, Update and Delete
API	Application Programming Interface
SOAP	Simple Object Access Protocol
RPC	Remote Procedure Call
REST	Representational State Transfer

SUMÁRIO

1 INTRODUÇÃO

A importância da organização é primordial para o sucesso de qualquer projeto, como por exemplo o desenvolvimento de software onde qualquer parte não documentada ou mal escrita pode levar ao fracasso de uma equipe. Diversas são as técnicas que auxiliam na entrega desses projetos, como os frameworks ágeis: SCRUM, EXTREME PROGRAMMING e KANBAN que são hoje os principais no mercado onde diversas empresas ao redor do mundo utilizam para obter uma maior coordenação de ideias e assim elaborar softwares de qualidade, a agilidade envolve tanto a capacidade de adaptação a diferentes mudanças e para refinar e ajustar processos conforme necessário (State of Agile Report., 2022).

O mesmo acontece com a vida estudantil, principalmente a do universitário onde este possui diversas cadeiras para cumprir e muitas das vezes também trabalha para se sustentar, por isso é necessário cada vez mais obter formas e técnicas de organização e planejamento de estudos para que alunos consigam performar de maneira a obter sucesso na sua caminhada. Diversas são as causas para um mau desempenho no meio acadêmico, como aspectos de relações interpessoais (problemas familiares ou com amigos, professores), problemas de saúde ou aspectos como o enfrentamento diante a preconceitos que grupos minoritários podem sofrer, essas dificuldades externas acrescentadas com a falta de uma organização, podem multiplicar os problemas ao longo da caminhada do estudante (BASSO *et al.*, 2022).

As metodologias ágeis são abordagens para o gerenciamento de projetos que enfatizam a flexibilidade, a adaptabilidade e a entrega contínua de valor. Elas foram desenvolvidas como uma alternativa aos processos de gerenciamento de projetos mais tradicionais, que são mais rígidos e menos adaptáveis às mudanças. As metodologias ágeis se baseiam nos valores e princípios descritos no Manifesto Ágil, que incluem a entrega contínua de valor, a colaboração entre os membros do time, a adaptação a mudanças e a simplicidade, são diversas as opções disponíveis, o presente trabalho irá focar no KANBAN como uma de suas funcionalidades dentro do sistema, apesar destas metodologias serem focadas em equipes, o KANBAN em particular possui uma característica muito simplificada e intuitiva que pode ser usada por qualquer indivíduo para suas tarefas diárias e por isso a importância e o enfoque desta metodologia para este trabalho. A visualização do que deve ser feito e quando, juntamente com o sentimento de progresso são fatores importantes para os estudantes que procuram se organizar (ALSAQQA, 2020).

Hoje existem muitos aplicativos que ajudam os estudantes a se organizarem melhor, como o Evernote para anotações de tarefas, o Google Calendar para calendário e o Todoist para gerenciamento de tarefas. No entanto, muitos desses aplicativos se concentram em uma área específica e não oferecem uma combinação de todas essas funcionalidades. Além disso, a usabilidade nem sempre é muito eficiente, o que pode afastar ou até mesmo estressar os usuários, não existindo uma junção de todas essas funcionalidades, além disso a usabilidade não costuma ser muito eficiente, o que pode afastar e até mesmo estressar o usuário da aplicação. Este projeto visa unificar funcionalidades destes softwares existentes hoje no mercado para que assim o usuário tenha disponível a maioria destas utilidades que ajudam na organização, não precisando ter que fazer *download* de diversas aplicações tendo assim um único ponto de referência para organizar sua vida estudantil.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é criar um aplicativo mobile e plataforma web que possa ser utilizado por universitários na organização e planejamento de seus estudos durante o semestre. Serão utilizadas técnicas do desenvolvimento ágil trazidas para a realidade de uma vida acadêmica. Poderá ser feito o cadastro de disciplinas, tendo assim um painel para acompanhar o progresso de cada tarefa atribuída, juntamente com ferramentas de registro de resumos, esquemas ou outras estratégias que os auxiliem no estudo dos conteúdos e na execução das tarefas. Haverá integração com dispositivos móveis e com sistema de calendário e também um módulo para facilitar o trabalho em grupo dos alunos.

1.1.2 Objetivos Específicos

1. Criar um aplicativo mobile para dispositivos Android e que seja acessível em termos de usabilidade para os alunos universitários.

2. Adotar estratégias de frameworks ágeis para a realidade de um estudante universitário.

3. Realizar testes manuais para obter métricas de desempenho, possibilitando assim gerar melhorias no futuro.

1.2 Delimitações

Este trabalho possui as delimitações a seguir:

- A aplicação deve possuir uma versão web e mobile.
- A versão mobile deverá estar disponível apenas para Android.

1.3 Resultados Esperados

- Aplicação web e mobile para gerenciamento de tarefas.
- Testes unitários e end-to-end da aplicação.
- Design responsivo, seguindo boas práticas de *User Experience* (UX) e *User interface* (UI).
- Documentação.

1.3 MÉTODO DE PESQUISA

Para guiar o trabalho e identificar quais funcionalidades da plataforma serão mais úteis e relevantes, será realizada uma coleta de dados e um estudo detalhado sobre as maiores dificuldades enfrentadas pelos universitários nos estudos e organização. Isso incluirá a coleta de dados, como questionários e entrevistas, para obter uma visão completa das necessidades e expectativas dos usuários. Com essas informações, será possível definir as funcionalidades mais importantes e relevantes para a plataforma, de modo a garantir a sua eficácia e usabilidade.

A análise de tecnologias das plataformas já existentes também será levada em consideração para uma viabilidade técnica e assim garantir um melhor resultado no desenvolvimento do projeto. A partir disso será guiado o trabalho para quais funcionalidades na plataforma serão mais úteis e quais não valem a pena serem exploradas.

Uma análise de UX e UI também será explorada, pois a importância de uma boa interface principalmente em um app de organização pessoal é primordial. Após o desenvolvimento do projeto será feito um teste com usuários reais e estudantes para relatarem a sua experiência utilizando a plataforma web e mobile, para que assim possa se ter uma ideia do que pode ser melhorado e quais funcionalidades foram as mais bem avaliadas.

1.4 ORGANIZAÇÃO

No capítulo 2, são apresentados alguns conceitos e ferramentas importantes que serão utilizados ao longo do trabalho assim como trabalhos similares que serão usados como base para a implementação.

No capítulo 3, são discutidos alguns trabalhos similares que servirão como inspiração principalmente no que tange a organização e metodologias ágeis.

No capítulo 4, há uma seção discutindo o desenvolvimento do projeto e aspectos arquiteturais do software, assim como tecnologias que serão utilizadas, requisitos, histórias de usuários e também mockups de algumas telas a serem desenvolvidas.

No capítulo 5, é apresentado o teste de usabilidade SUS para avaliar o protótipo desenvolvido neste trabalho.

No capítulo 6, é apresentada a conclusão do presente trabalho juntamente com possíveis funcionalidades a serem implementadas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados temas e tecnologias que serão usados para o desenvolvimento do trabalho e assim explicar do por que a utilização e escolha dos mesmos na aplicação. Também será apresentado um quadro comparativo das tecnologias e metodologias ágeis para um melhor entendimento de suas características.

2.1 Conceitos

2.1.1 Organização

A organização é uma palavra originada do grego *"organon"* que significa instrumento, utensílio, órgão ou aquilo com que se trabalha e possui diversos significados dependendo do contexto, mas de forma geral o significado seria a forma como um sistema se dispõe para atingir os resultados pretendidos (CEARÁ, 2015). Para o escopo deste trabalho será usado essa palavra principalmente no sentido de "ordem" para as tarefas a serem executadas pelos alunos.

Existem diversas metodologias ágeis que ajudam indivíduos a se organizarem de forma inteligente e metódica, as quais são utilizadas principalmente em ambiente corporativos, o que há de novo nas metodologias ágeis não são as práticas usadas, mas o reconhecimento das pessoas como principal impulsionador do sucesso do projeto, juntamente com um foco intenso em eficácia e manobrabilidade (COCKBURN, 2022).

2.1.2 Tarefa

A definição de tarefa descreve uma atividade de trabalho que por sua vez é desempenhada por funções específicas. A granularidade de uma tarefa é geralmente algumas horas ou poucos dias e afeta um número pequeno de pessoas. As tarefas são utilizadas como uma base para o processo de planejamento (PERNAMBUCO, 2005).

As etapas de uma tarefa podem ser definidas como:

- **Etapas de Estudo:** em que o indivíduo que desempenha a função entende a natureza da tarefa, reúne e examina os Produtos de Trabalho de entrada e formula o resultado.
- **Etapas de Execução:** em que o indivíduo que desempenha a Função cria ou atualiza alguns Produtos de Trabalho.
- **Etapas de Revisão:** em que o indivíduo que desempenha a Função inspeciona os resultados em relação a alguns critérios.

2.1.3 Métodos de Estudos

Basso (2013) apresenta um estudo sobre a experiência universitária quanto ao rendimento dos estudantes e suas vivências acadêmicas através das Oficinas de Organização e Métodos de Estudo da Universidade Federal de Santa Catarina (UFSC). Estas oficinas fazem parte do projeto REUNI do programa de Pós-Graduação em Psicologia. Segundo as autoras, em uma primeira etapa da pesquisa foram levantados os índices de reprovação por FI (frequência insuficiente) e de evasão. No âmbito da UFSC, FI é caracterizado quando os alunos não têm 75% de presenças nas atividades da disciplina.

Foi relatado diversos motivos para a causa do mal desempenho dos estudantes, um dos fatores seria a um desajustamento psicossocial no processo ensino-aprendizagem e a incompatibilidade vocacional. Também vale muito a pena citar problemas socioeconômicos

dos estudantes que precisam trabalhar e estudar ao mesmo tempo e também a falta de uma base escolar principalmente em cálculos.

No artigo cita uma triagem feita entre estudantes da universidades com os cursos com maior evasão e assim recolhido dados dos principais motivos para a busca pelos estudantes às oficinas Organização e Métodos de estudo:

Tabela 1 - Motivos para a busca dos estudantes às oficinas Organização e Métodos de estudo

Motivos	Nº	%
Dificuldades para organizar o tempo	10	27
Necessidade de melhor organizar os estudos	08	22
Problemas de focalização da atenção	04	11
Necessidade de aprender a se disciplinar	03	8
Baixo desempenho acadêmico	03	8
Maximizar o seu rendimento	03	8
Buscar equilíbrio entre a vida acadêmica e a vida pessoal	02	5

Fonte: Elaboração do autor, 2022

Sendo assim percebe-se que grande parte dos alunos possuem problemas quanta a falta de organização segundo Maccann (2010) “Os alunos altamente conscienciosos tendem a usar mais estratégias de gerenciamento de tempo, principalmente aquelas relacionadas a cumprimento de prazos, organização e planejamento”, O mesmo autor relata que a conscienciosidade se difere de saber gerenciar o tempo pois, no último a pessoa precisa de uma forma empírica para praticar e treinar esse hábito ao longo do tempo, já a conscienciosidade está ligado a fatores genéticos muitas vezes.

Por isso a questão da organização é tão primordial para o sucesso na vida acadêmica já que sem ela pode-se acarretar em uma baixa na produtividade e assim como no desenvolvimento de software e projetos, produtividade é a palavra chave para o sucesso.Uma das ferramentas mais importantes e conhecidas de organização de trabalho e tarefas é sem sombra de dúvidas o Kanban e segundo Wakode (2015) seus princípios são:

- Visualização do trabalho
- Limitar o trabalho em andamento
- Foco no fluxo
- Melhoria Contínua

Com este tipo de metodologia ágil pode-se trazer o controle de tarefas e afazeres de um estudante para um board e assim também utilizá-lo para seu benefício no dia a dia.

2.1.4 Metodologias ágeis

As metodologias ágeis são um amplo guarda-chuva de crenças de desenvolvimento de software. É uma estrutura conceitual para engenharia de software que começa com uma fase de planejamento inicial e segue o caminho para a fase de implantação com interações iterativas e incrementais durante todo o ciclo de vida do projeto. O objetivo inicial dos métodos ágeis é reduzir a sobrecarga no processo de desenvolvimento de software com a capacidade de adotar mudanças sem arriscar o processo ou sem retrabalho excessivo (ALSAQQA, 2020).

O manifesto ágil possui uma série de valores e princípios que são a base para o desenvolvimento dos processos que os guiam. A começar pelos valores que são:

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

Além disso o manifesto possui 12 princípios complementares que fortificam a mentalidade ágil:

- Princípio 1: satisfazer o cliente através da entrega contínua
- Princípio 2: Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento
- Princípio 3: Entregar frequentemente software funcionando
- Princípio 4: Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto
- Princípio 5: Construir projetos em torno de indivíduos motivados
- Princípio 6: O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face
- Princípio 7: Software funcionando é a medida primária de progresso
- Princípio 8: Os processos ágeis promovem desenvolvimento sustentável
- Princípio 9: Contínua atenção à excelência técnica e bom design aumenta a agilidade
- Princípio 10: Simplicidade – A arte de maximizar a quantidade de trabalho não realizado – é essencial
- Princípio 11: As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis
- Princípio 12: Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

A facilidade de visualização das tarefas e sensação de progresso é um fator importante, a aplicabilidade destas metodologias é muito útil para aqueles que buscam uma organização em seu ambiente de trabalho e se encaixa perfeitamente em um ambiente acadêmico.

Algumas das metodologias ágeis mais utilizadas hoje no mercado são:

Kanban: O Kanban é uma técnica de gerenciamento de projetos que foi desenvolvida no Japão no final da década de 1950. O termo "kanban" é uma palavra japonesa que significa "cartão visual". A técnica foi criada como uma forma de melhorar a eficiência e a produtividade na produção de peças de automóveis pela Toyota. A ideia por trás do Kanban é simples: criar um sistema visual que permita aos funcionários ver o que precisa ser feito e quando precisa ser feito. Isso é feito através de cartões ou tabelas que são colocados em uma parede ou painel de forma que todos possam ver. Cada cartão representa um item ou tarefa específica que precisa ser concluída.

Foi originalmente desenvolvido como uma técnica para a produção em massa, mas hoje em dia é amplamente utilizado em uma variedade de indústrias e ambientes de trabalho. É especialmente útil para projetos que envolvem muitas tarefas pequenas e independentes, como o desenvolvimento de software. Para usar o Kanban, os funcionários criam uma lista de tarefas e as dividem em três categorias: prontas para serem iniciadas, em andamento e concluídas. Cada tarefa é então representada por um cartão ou item em uma tabela é movida através dessas três categorias conforme avança. Isso permite que todos vejam o progresso do projeto e saibam o que precisa ser feito a seguir.

Inclui o conceito de "limite de trabalho em andamento" (WIP, na sigla em inglês), que se refere à quantidade máxima de tarefas que podem estar em andamento ao mesmo tempo. Isso ajuda a evitar o sobrecarregamento de trabalho e a garantir que todas as tarefas sejam concluídas de maneira otimizada. É uma técnica de gerenciamento de projetos altamente visual e intuitiva que tem sido amplamente adotada em todo o mundo. Além disso, o Kanban é altamente adaptável e pode ser facilmente ajustado para atender às necessidades de uma ampla variedade de projetos e ambientes de trabalho. (CORONA, 2013).

Scrum: O Scrum surgiu nos anos 1980 como um framework para gerenciamento de projetos de software. Foi desenvolvido por Hirotaka Takeuchi e Ikujiro Nonaka, que estudavam como os times de desenvolvimento de software poderiam se adaptar rapidamente a mudanças e entregar valor contínuo aos clientes. Takeuchi e Nonaka observaram que os times de desenvolvimento de software tradicionais, que seguiam metodologias de linha de montagem, tendiam a ser inflexíveis e lentos para se adaptar a mudanças.

Eles propuseram uma nova abordagem, baseada em iterações curtas e entrega contínua de valor, que se tornaria conhecida como Scrum. Se baseia no modelo de desenvolvimento ágil, que enfatiza a adaptabilidade, a entrega contínua de valor e a colaboração com o cliente. Divide-se em três papéis principais: o Scrum Master, o Product Owner e o time Scrum. O Scrum Master é responsável por garantir que o time Scrum esteja seguindo as regras e práticas do Scrum, enquanto o Product Owner é responsável por definir o que o time Scrum deve fazer e o time Scrum é responsável por realizar o trabalho.

O ciclo de trabalho do Scrum é dividido em iterações curtas chamadas "sprints". Em cada sprint, o time Scrum se reúne para planejar o trabalho a ser

realizado, executar o trabalho e revisar o progresso. Isso permite que o time se adapte rapidamente a mudanças no projeto e forneça valor contínuo para o cliente. É amplamente utilizada em projetos de software e é conhecida por sua flexibilidade e eficiência, especialmente útil em projetos de grande escala ou de alto nível de incerteza, onde a adaptabilidade é crucial. (PEREIRA, 2007).

Extreme Programming (XP): Foi desenvolvida por Kent Beck no final da década de 1990. Beck trabalhava como programador e líder de projeto em diversos projetos de software e percebeu que as metodologias tradicionais de gerenciamento de projetos tendiam a ser inflexíveis e lentas para se adaptar a mudanças. Inspirado pelo modelo de desenvolvimento ágil, que enfatiza a adaptabilidade, a entrega contínua de valor e a colaboração com o cliente, Beck desenvolveu o Extreme Programming como uma forma de tornar os projetos de software mais eficientes e adaptáveis.

Beck publicou o livro "Extreme Programming Explained: Embrace Change" em 1999, que descreveu os princípios e práticas do XP como são conhecidos hoje. A metodologia XP se divide em duas fases principais: o planejamento e a refatoração. Durante a fase de planejamento, o time de desenvolvimento se reúne com o cliente para definir os objetivos e as funcionalidades do projeto. Na fase de refactoring, o time implementa e testa as funcionalidades de acordo com os objetivos definidos. É caracterizada por seus 12 princípios:

- O cliente deve sempre estar presente.
- O projeto deve ser desenvolvido em iterações curtas.
- O projeto deve ser planejado com precisão.
- Mudanças são esperadas, bem-vindas e gerenciadas.
- O projeto é desenvolvido em pares.
- O projeto é baseado em testes.
- O projeto é baseado em comunicação clara e concisa.
- A simplicidade é valorizada.
- O projeto é gerenciado com transparência.
- O projeto é gerenciado com feedback constante.
- O projeto é gerenciado com a qualidade como principal objetivo.
- O projeto é gerenciado com a colaboração como principal objetivo.

Baseia-se em práticas ágeis, como a programação em pares, o teste de unidade e o teste integrado. A metodologia XP é amplamente utilizada em projetos de software e é conhecida por sua eficiência e flexibilidade. Ela é especialmente útil em projetos de pequena escala ou de alto nível de incerteza, onde a adaptabilidade é crucial. (TELES, 2005).

3 SOLUÇÕES SIMILARES

O tema organização pessoal e empresarial é recorrente, por isso diversos são os sistemas e aplicativos que tentam suprir essa necessidade por ordem nas tarefas diárias dos indivíduos.

A seguir são mostradas algumas destas soluções tanto no âmbito de uma empresa quanto para o uso pessoal. Cada um destes sistemas possuem suas funcionalidades particulares porém sempre com o mesmo intuito. Alguns dos critérios mais importantes a serem considerados incluem:

- Eficiência: é importante escolher uma solução que seja eficiente e otimize o tempo e os esforços do usuário. Isso inclui considerar se a solução é fácil de usar e se possui recursos que facilitam a realização de tarefas.
- Customização: é importante avaliar se a solução pode ser personalizada de acordo com as necessidades e objetivos específicos do usuário. Isso inclui considerar se é possível adicionar ou remover recursos, bem como se é possível ajustar as configurações e as preferências do usuário.
- Recursos: é fundamental avaliar os recursos disponíveis na solução e verificar se eles são adequados para atender às necessidades do usuário. Isso inclui considerar se a solução possui recursos avançados, como integrações com outras ferramentas ou recursos de automação.
- Usabilidade: é importante avaliar a usabilidade da solução, ou seja, sua facilidade de uso. Isso inclui considerar se a interface é intuitiva e fácil de entender, bem como se a solução possui tutoriais ou recursos de ajuda disponíveis.

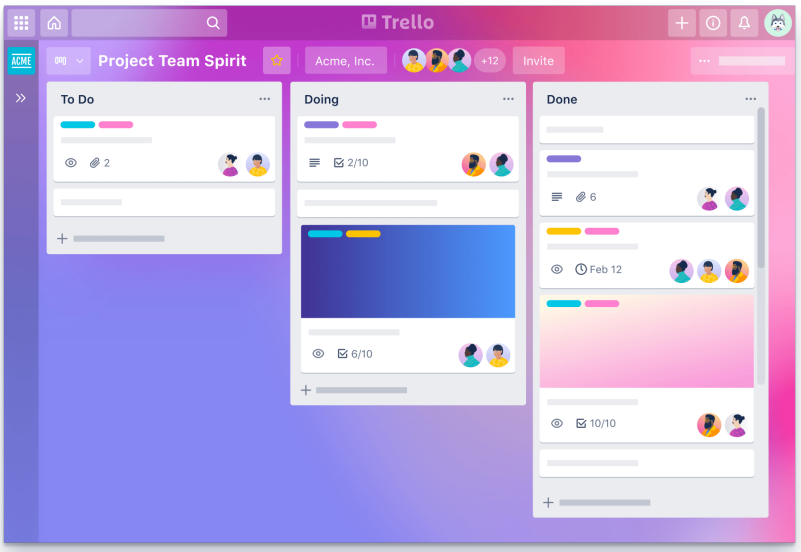
As aplicações a seguir foram encontradas através de pesquisas em artigos especializados no tema de organização de tarefas e assim escolhidos baseados nas funcionalidades que fazem sentido para um estudante.

3.1 Trello

Ferramenta visual que capacita equipes a gerenciar qualquer tipo de projeto, fluxo de trabalho ou rastreamento de tarefas. Permite adicionar arquivos, listas de verificação ou até mesmo automação.

Em sua essência, o Trello conta com os princípios dos quadros de projetos Kanban para visualizar fluxos de trabalho, fornecendo aos gerentes e membros da equipe uma visão geral simples de um projeto do início ao fim.

Figura 1 - Board do trello



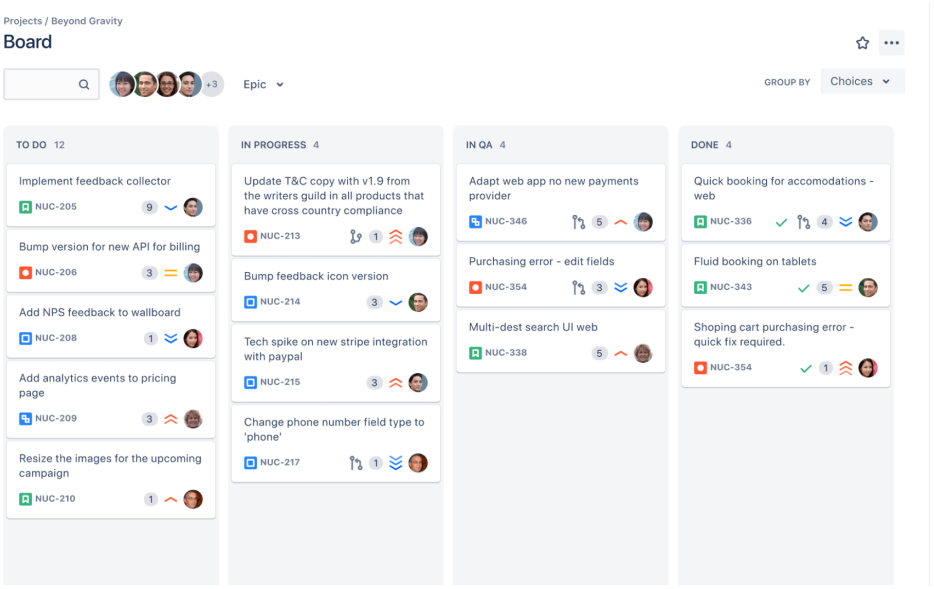
Fonte:https://tecnoblog.net/responde/trello-nao-permite-mais-criar-um-board-fora-de-um-time/

Outra característica é sua facilidade de uso e versatilidade: o aplicativo geralmente também é usado para uso pessoal, incluindo planejamento de férias a casamentos. Nesse sentido, difere do software de gerenciamento de projetos completo, priorizando uma funcionalidade leve e acessibilidade em um amplo conjunto de recursos (GLITCH, 2011).

3.2 Jira

O Jira Software faz parte de uma família de produtos desenvolvidos para ajudar equipes de todos os tipos a gerenciar o trabalho. Originalmente, o Jira foi projetado como um rastreador de bugs e problemas. Mas hoje, o Jira evoluiu para uma poderosa ferramenta de gerenciamento de trabalho para todos os tipos de casos de uso, desde requisitos e gerenciamento de casos de teste até desenvolvimento ágil de software.

Figura 2 - Board do Jira



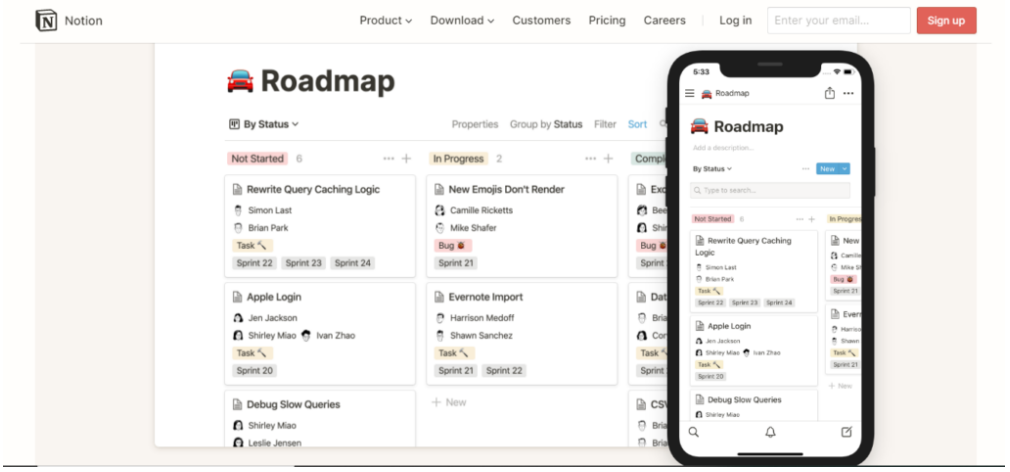
Fonte: https://www.atlassian.com/br/software/jira/features

A plataforma aproveita todos os tipos de habilidades de gerenciamento de projetos, incluindo desenvolvimento de software, gerenciamento ágil de projetos, rastreamento de bugs, gerenciamento de scrum, gerenciamento de conteúdo, marketing, gerenciamento de serviços profissionais e muito mais (ATLASSIAN, 2004).

3.3 Notion

Notion é uma ferramenta de produtividade e uma ferramenta de gerenciamento de projetos para ajudar na organização. Possui muitas ferramentas de produtividade, como Tarefas, Notas, Cadernos e Lembretes. Também possui Tabelas, Bancos de Dados. Ajuda pequenas e grandes organizações a coordenar prazos, objetivos e atribuições.

Figura 3 - Página do Notion



Fonte: https://www.apptuts.net

Permite construir um wiki pessoal com infinitas camadas de conteúdo, planejar usando uma visualização kanban, um calendário ou uma visualização de lista simples.

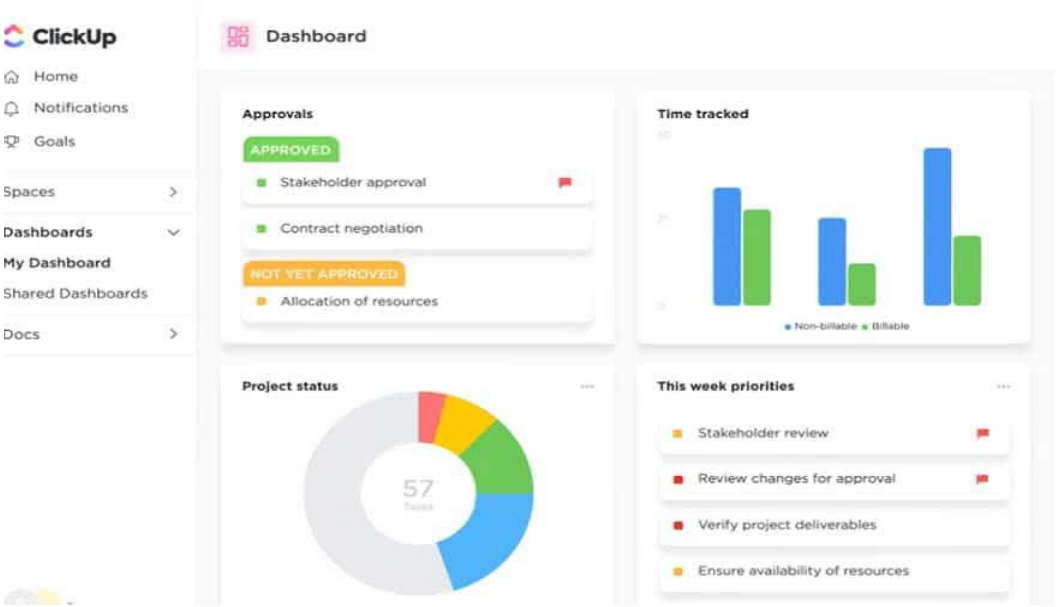
Fornece funcionalidade de equipe para colaboração em tempo real e permite que as equipes compartilhem, comentem e atribuem tarefas e lembretes. Assim como indivíduos e profissionais podem usar o Notion, as equipes também podem (NOTION, 2016).

3.4 Click Up

A clickUp é uma plataforma de colaboração e gerenciamento de projetos para times de pequena ou grande escala, possui diversas funcionalidades, entre elas ferramentas de comunicação, atribuições e status de tarefas, alertas, chat. Permite que os usuários documentem bugs ou anote atas de reuniões com o ClickUp Docs e editem em tempo real com outras pessoas, além disso Possui integração com diversas outras ferramentas como: Github, Slack, GitLab, GoogleDrive, Outlook, Figman, Google Calendar.

Os projetos podem ser visualizados em um painel Agile ou organizados pelo responsável. O fluxo de atividades exibe as tarefas à medida que são criadas e concluídas em tempo real.

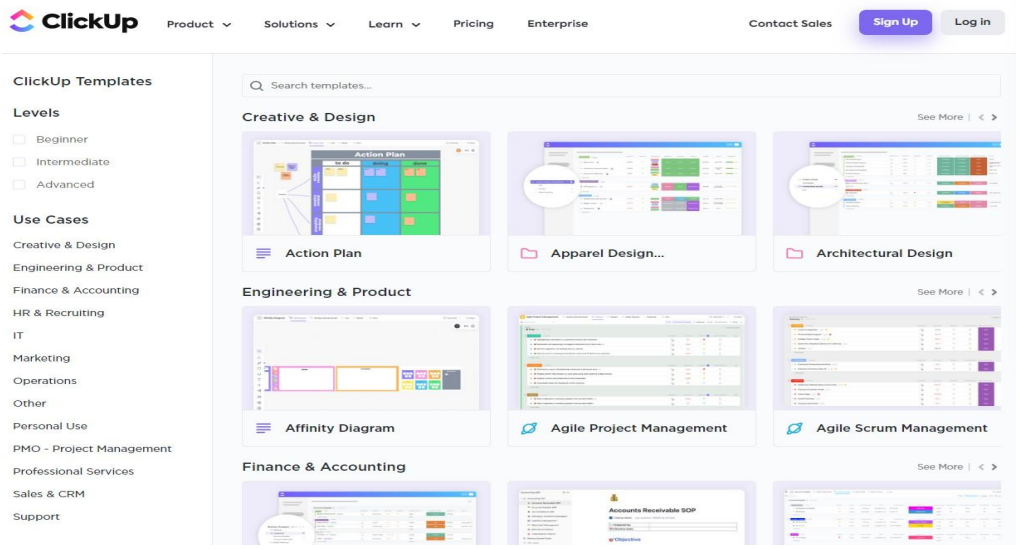
Figura 4 - Dashboard do ClickUp



Fonte: https://clickup.com/

Há também a possibilidade de importar templates através do *template center*, uma biblioteca de modelos prontos para uso. Você pode acessar o Template Center de várias maneiras para localizar modelos para localizar um modelo criado por membros do seu espaço de trabalho ou pela comunidade ClickUp. Os modelos podem ser mantidos privados ou compartilhados com pessoas específicas ou equipes (CLICKUP, 2017)

Figura 5 - Página de templates da ClickUp



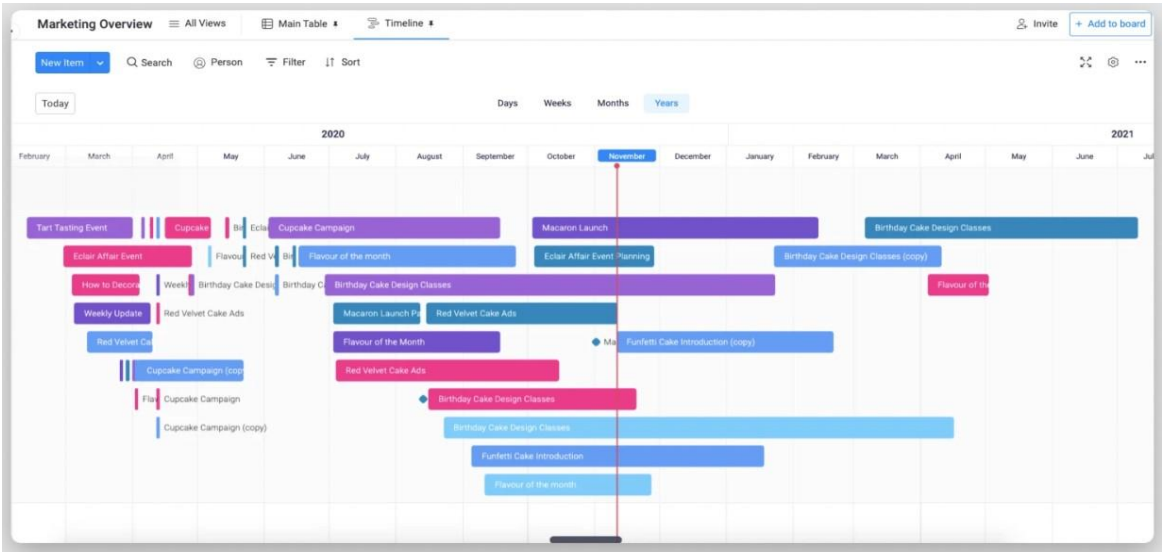
Fonte: https://clickup.com/

3.5 Monday.com

É uma ferramenta de gerenciamento de projetos, oferece uma rica visualização dos projetos com base nas preferências do usuário, incluindo Kanban, Gantt, linha do tempo e exibições de calendário. Se a equipe quiser ter uma visão geral ou quer ver as tarefas diárias, pode-se facilmente alternar as exibições, detalhar ou diminuir o zoom o quanto precisar, além de criar exibições personalizadas para reunir todos os dados e informações em um só lugar.

Similar ao ClickUp há também como adicionar templates para diferentes categorias de negócio, como design, desenvolvimento de software, RH, marketing, vendas entre outras.

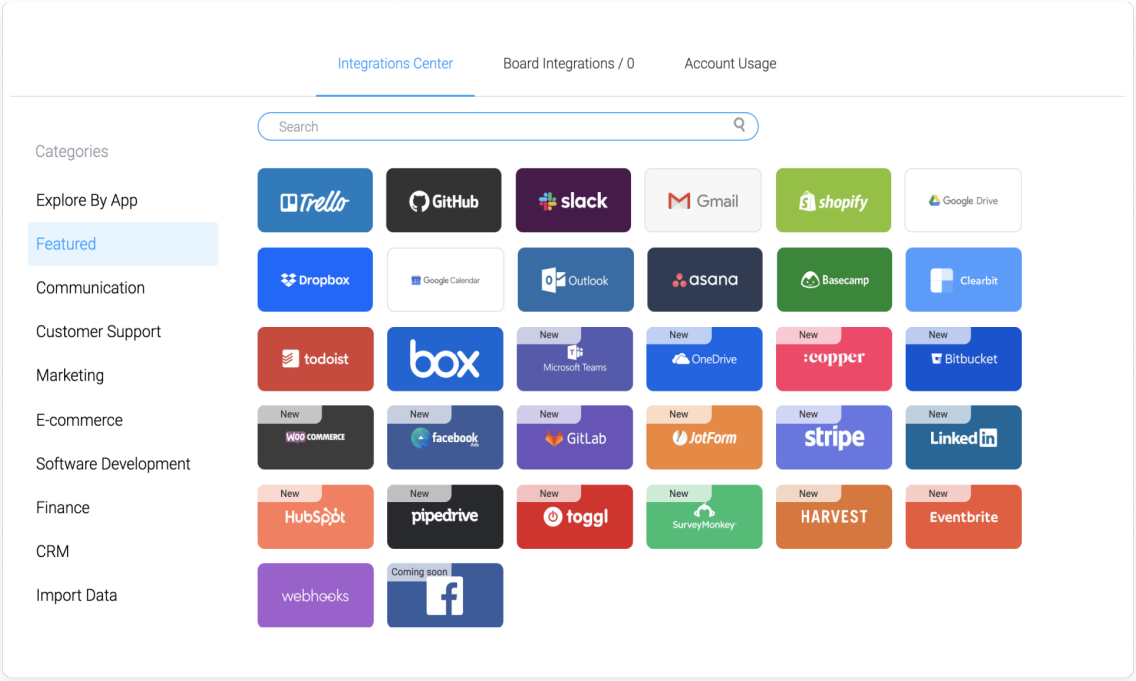
Figura 6 - Timeline da Monday.com



Fonte: https://monday.com

Outra funcionalidade é a integração com as ferramentas e aplicativos, a qual potencializa os processos de gerenciamento de projetos e assim cria automações de fluxo de trabalho para eliminar tarefas administrativas repetitivas. Alguns dos softwares que são possíveis de integrar são: Slack, Zoom, Shopify, Google Calendar e dezenas de outras ferramentas (Monday.com, 2017).

Figura 7 - Página de integrações da monday.com



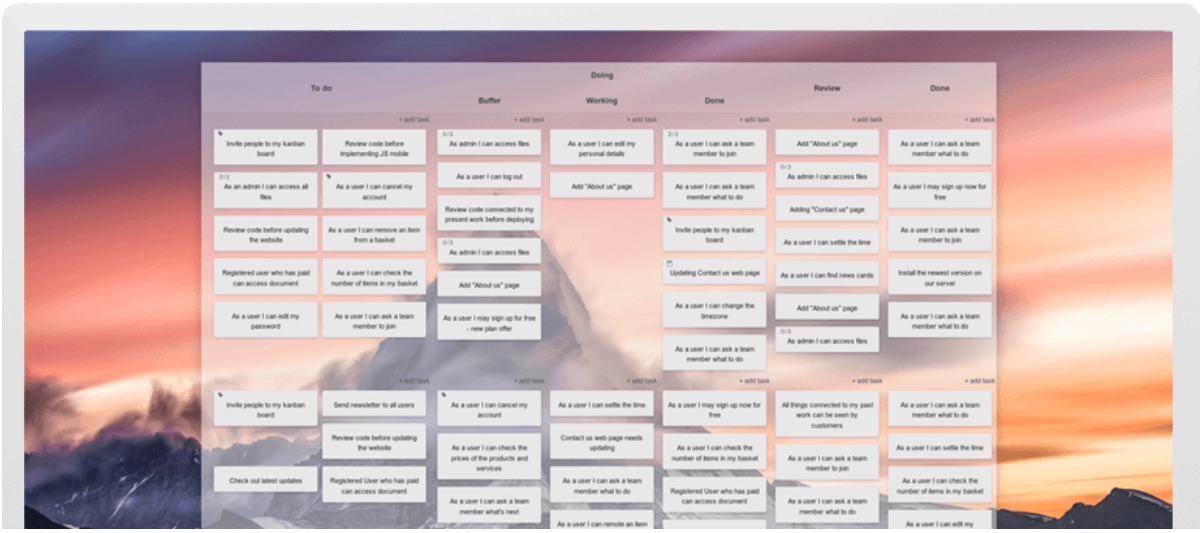
Fonte: https://monday.com

3.6 Kanban Tool

É um aplicativo para gerenciamento de projetos e tarefas que utiliza a metodologia Kanban. Pode-se criar um quadro Kanban personalizado e permitir o acesso a ele para os membros de uma equipe de trabalho, para que todos possam acompanhar o andamento das tarefas. Um quadro Kanban deve consistir em pelo menos três colunas: uma lista de pendências, na qual deve-se reunir todas as tarefas que precisam ser executadas, uma em andamento e uma coluna de tarefas concluídas - onde são colocadas todas as tarefas concluídas e de onde podem ser arquivadas para servir a um propósito analítico. Pode haver quantas colunas forem necessárias, cada uma representando um estágio específico do processo de trabalho específico.

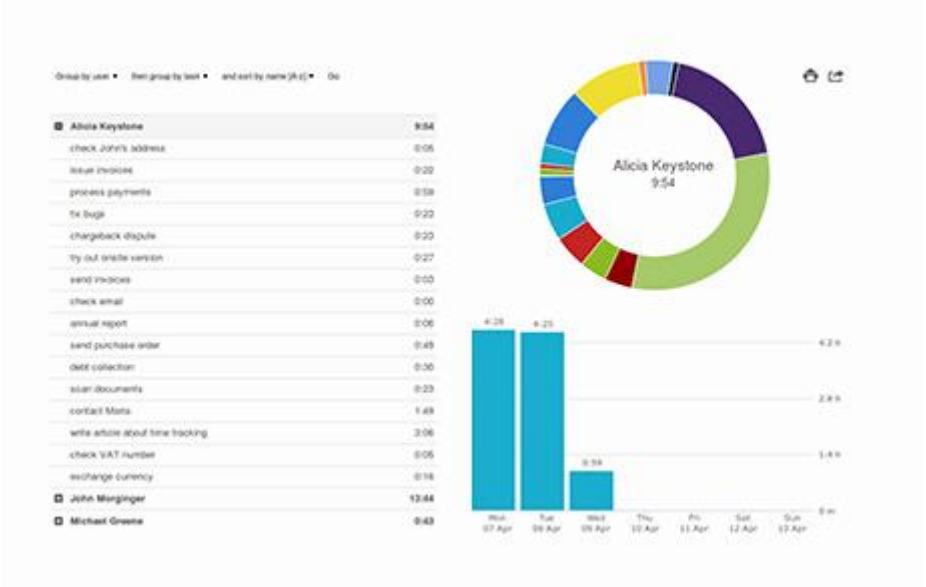
Também é possível identificar e eliminar problemas com análises e métricas Kanban. Fazendo assim um melhor planejamento, monitoramento e consequentemente melhorando o desempenho do projeto utilizando o diagrama de fluxo cumulativo e o relatório de duração de ciclo. (KANBAN TOOL, 2012).

Figura 8 - Board do Kanban Tool



Fonte: https://kanbantool.com

Figura 9 - Análise métrica do Kanban Tool



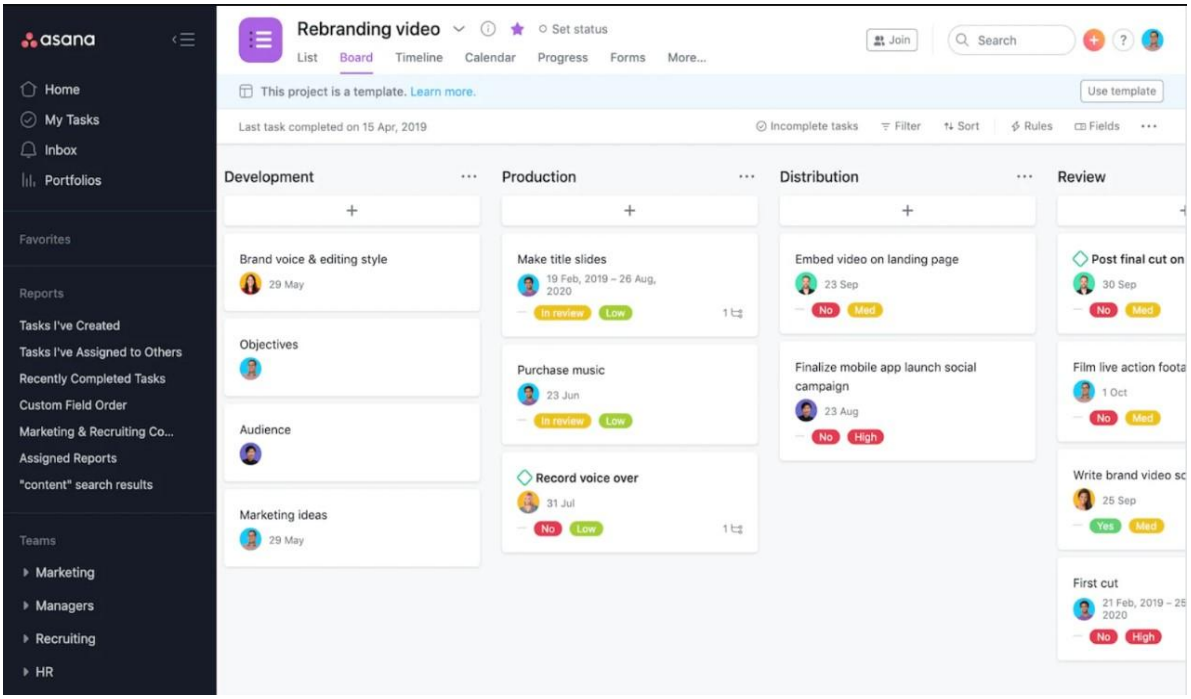
Fonte: https://kanbantool.com

3.7 Asana

Com Asana, usuários podem visualizar projetos e gerenciar tarefas de várias maneiras diferentes, incluindo quadros Kanban, listas, calendários, portfólios, cargas de trabalho e cronogramas, semelhante aos seus concorrentes ClickUp e Monday.com. Cada membro da equipe pode gerenciar tarefas da maneira que funciona melhor para eles, permitindo que sejam o mais produtivos possível. A alternância entre visualizações também oferece aos gerentes de projeto uma perspectiva geral sobre a posição do projeto e pode-se criar painéis de projeto personalizáveis que são atualizados em tempo real.

É possível configurar regras e ações personalizadas utilizando o construtor gráfico de fluxo de trabalho da Asana. Pode-se usá-lo para configurar automações básicas, como datas de vencimento em cascata, atribuir uma tarefa à próxima pessoa no fluxo ou alertar a equipe no Slack quando um projeto é concluído, até mesmo os fluxos de trabalho mais complexos que abrangem dezenas de usuários, ferramentas e tarefas.

Figura 10 - Board da Asana



Fonte: https://asana.com

3.8 Discussões sobre as ferramentas

As ferramentas apresentadas possuem uma grande quantidade de funcionalidades e cumprem o que propõem, porém algumas delas pecam na questão da usabilidade, como é o caso do jira, Monday.com, ClickUp, pois pensando em um único usuário e sendo este um estudante a maioria destas funcionalidades não serão utilizadas e são pensadas na maioria das vezes para times. O board Kanban em si destas aplicações é bastante usual, mas outras funcionalidades se perdem em meio a um tela não muito amigável em termos de

UX. O notion por sua vez tem um visual bem minimalista, mas pelo mesmo motivo do Jira, apresenta alguns aspectos complexos em relação às suas funcionalidades.

O trello e Notion em comparativo com o Jira aparenta ser muito mais intuitivo e simples para ser utilizado e isso se dá pelo fato de não ser tão focado em um meio empresarial, porém ainda sim é muito focado em times, o que não dá uma atenção maior para funcionalidades mais relacionadas a apenas uma pessoa e suas tarefas particulares de estudante. Já o Kanban Tool de todos os softwares apresentados é o que mais se aproxima de uma boa UX, é fácil e simples de ser utilizado e cumpre bem o que faz, porém é focado somente a quadros Kanban e sendo assim não apresenta outras funcionalidades comparado as outras aplicações.

4 Desenvolvimento do projeto

Para este projeto, as tecnologias a serem utilizadas foram pensadas levando em conta sua performance e praticidade. O aplicativo será multiplataforma entre web e *mobile*, por isso é importante a utilização de tecnologias que facilitem o desenvolvimento.

4.1 Tecnologias aplicáveis

Abaixo serão apresentadas algumas tecnologias que são utilizadas para o desenvolvimento do projeto.

4.1.1 ReactJs

O ReactJS[1] foi criado pela empresa Facebook (hoje Meta) que precisava de uma biblioteca para o desenvolvimento de seus produtos. Em 2013 essa biblioteca foi colocada à disposição de outros desenvolvedores e, de lá pra cá, se tornou uma das principais bibliotecas de desenvolvimento em JavaScript.

Por isso, com o intuito de facilitar a criação das interfaces do sistema web em conjunto com a biblioteca de componentes ant design, a qual fornece uma ampla variedade de componentes pré-projetados, como botões, formulários, tabelas e caixas de seleção, que podem ser facilmente incorporados em aplicativos da web. O uso da biblioteca ant design foi considerado para proporcionar uma aparência consistente de componentes tanto na web quanto no mobile, agilizando o processo de desenvolvimento.

React é uma biblioteca Javascript para criação de interfaces. O principal conceito desta tecnologia está nos chamados componentes, que nada mais são do que dividir a interface em diferentes pedaços, criando assim uma melhor abstração e principalmente organização no código do projeto. Outro conceito importante são os estados que cada componente pode ter, eles servem para guardar algum tipo de dado para ser utilizado depois.

---

[1] Disponível: https://pt-br.reactjs.org/

Portanto, por ser fácil e intuitiva de ser utilizada e com uma comunidade bem ativa, foi escolhida React para ser a tecnologia principal para a construção das interfaces do projeto (WALKE, 2013).

4.1.2 Styled Components

Styled Components é uma biblioteca de estilo para React que permite escrever código CSS (Cascading Style Sheets) em componentes de React. Isso significa que os estilos são escritos diretamente no código JavaScript e podem ser aplicados aos elementos HTML (HyperText Markup Language) que o componente renderiza. Uma das principais vantagens de usar Styled Components é que permite manter os estilos bem organizados e anexados aos componentes relevantes. Isso pode ajudar a tornar o código mais legível e fácil de manter, especialmente em projetos maiores. Outra vantagem é que é possível aplicar estilos dinâmicos aos componentes, o que é útil em situações em que os estilos precisam mudar com base em propriedades ou estado do componente (STYLED COMPONENTS, 2015).

4.1.3 Redux

Além disso, será usada a biblioteca Redux[2] para gerenciamento de estados globais, o qual é de suma importância para a organização do código. Redux é uma biblioteca JavaScript de código aberto usada para gerenciar o estado do aplicativo, é amplamente utilizado com React, mas pode ser usado com qualquer outra biblioteca ou framework. É especialmente útil em aplicações de tamanho médio a grande, onde o gerenciamento do estado da aplicação pode se tornar complexo e desafiador. Permite que componentes React leiam dados de um Redux Store e enviem Actions para o Store para atualizar dados. O Redux ajuda os aplicativos a serem dimensionados, fornecendo uma maneira sensata de gerenciar o estado por meio de um modelo de fluxo de dados unidirecional. React Redux é conceitualmente simples, os estados são salvos na “Redux store” e então verifica-se os dados que seu componente deseja foram alterados e os renderiza novamente (ABRAMOV, 2015).

4.1.4 NodeJs

Para o backend será utilizado o NodeJs[3] outra tecnologia do ecossistema javascript que irá ajudar a criar aplicativos de rede rápidos e escaláveis. O modelo de encadeamento único com loop de eventos é muito útil e permite lidar com várias solicitações de clientes. Essa é uma das principais vantagens de usar o NodeJs para aplicações web. Além disso, a sintaxe javascript comum entre front-end e back-end é outro atrativo para a escolha da tecnologia. Portanto, a principal vantagem do uso desta tecnologia para este projeto se dá

---

[2] Disponível em: https://redux.js.org/

[3] Disponível em: https://nodejs.org/en/

pelo fato da syntaxe javascript, facilitando muito o desenvolvimento da aplicação (DAHL, 2009).

4.1.5 Postgresql

PostgreSQL[4] é um sistema de gerenciamento de banco de dados relacional de código aberto (open source). Foi desenvolvido como um fork do projeto POSTGRES, que foi iniciado na Universidade de Califórnia, Berkeley, nos Estados Unidos, em 1986. Desde então, tem se tornado um dos sistemas de gerenciamento de banco de dados mais populares e avançados do mundo. PostgreSQL é conhecido por sua robustez, flexibilidade e escalabilidade. Suporta muitos tipos de dados diferentes, incluindo texto, números, geometria e tipos de dados de data/hora. Também oferece uma ampla variedade de ferramentas de análise de dados, incluindo suporte para consultas SQL avançadas, índices e triggers. Além disso, o PostgreSQL é altamente compatível com padrões da indústria e é amplamente utilizado em aplicativos empresariais, científicos e governamentais ao redor do mundo. Ele é compatível com muitos sistemas operacionais diferentes, incluindo Linux, MacOS e Windows, e é compatível com muitas linguagens de programação (STONEBRAKER, 1986).

4.2 Requisitos do projeto

4.2.1 Tabelas de requisitos

A seguir será apresentado os requisitos do sistema, baseado nas pesquisas realizadas e das necessidades dos estudantes.

Tabela 2 - Tabela de requisitos

Ident	Requisito	
R1	Logar e deslogar do sistema	**Obrigatório**
R2	Cadastrar disciplinas: cada disciplina terá um id, nome e uma cor	**Obrigatório**
R3	Remover disciplinas	**Obrigatório**
R4	Cadastrar uma tarefa: cada tarefa terá um id, titulo e descrição	**Obrigatório**
R5	Editar uma tarefa	**Obrigatório**
R6	Adicionar uma tarefa no calendário: para cada disciplina poderá ser adicionada uma tarefa com uma data e hora para sua finalização	**Obrigatório**

[4] Disponível em: https://www.postgresql.org

R7	Exibir as tarefas das disciplinas no board: as tarefas das disciplinas poderão ser arrastadas em um board com os seguintes estados: “Tarefas”, “Fazendo”, “Concluído”	**Obrigatório**
R8	Filtrar board das disciplinas: O usuário irá poder filtrar por disciplina, prioridade e tempo de término (30 dias, 15 dias, 7 dias)	**Obrigatório**
R9	Remover tarefa do board	**Obrigatório**
R10	Exibir chat para comunicação entre alunos: O sistema terá um meio de comunicação via texto.	**Desejável**
R11	Visualizar detalhes das disciplinas	**Obrigatório**
R12	Visualizar disciplinas	**Obrigatório**
RN1	Linguagem Javascript	**Obrigatório**
RN2	Idioma: Português	**Obrigatório**
RN3	Banco de dados: Postgresql	**Obrigatório**
RN4	IDE de desenvolvimento: Visual Studio Code	**Obrigatório**

Fonte: Elaboração do autor, 2022

4.2.2 Histórias de usuário

- Como **estudante**, eu gostaria de conseguir **organizar** melhor minhas **tarefas** da faculdade para que eu consiga obter um **melhor resultado** nas minhas **notas**.

- Para ter um **melhor proveito** do semestre eu como **estudante** quero ter um **calendário organizado** com as **tarefas** que preciso fazer.

- Como **estudante**, gostaria de ter um meio em que possa fazer **anotações** em texto rico das disciplinas e poder **compartilhá-las** com colegas a fim de ter **resumos** melhor elaborados.

- Para conseguir melhor **visualizar** as minhas **tarefas**, eu como **estudante** quero ter um **board** estilo **kanban** para que assim tenha uma melhor noção de **progresso** das minhas tarefas.

4.2.3 Mockups

Uma primeira iteração com os requisitos listados no item 4.2 permitiu gerar as propostas de interfaces apresentadas abaixo, com suas respectivas funcionalidades. Tais telas poderão ser revisadas/alteradas à medida que forem sendo desenvolvidas as funcionalidades definitivas.

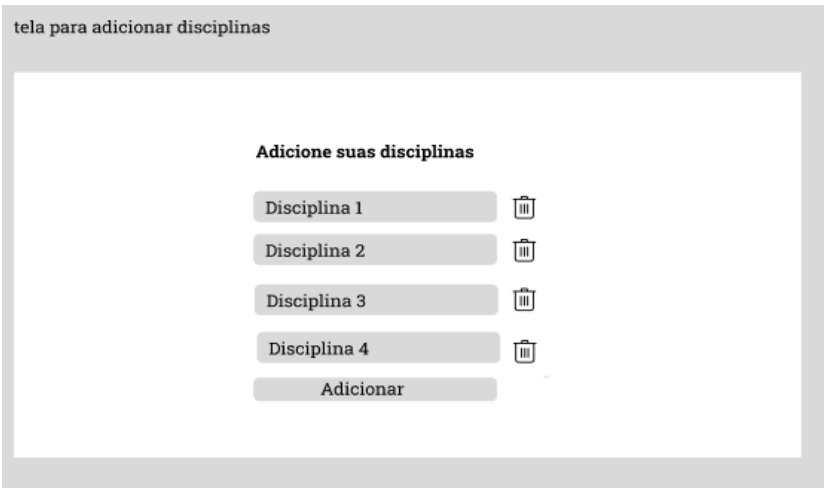
Tela de login (Requisito R1)

Figura 11 - Mockup tela de login

Fonte: Elaboração do autor, 2022

Tela para adicionar as disciplinas (Requisito R2)

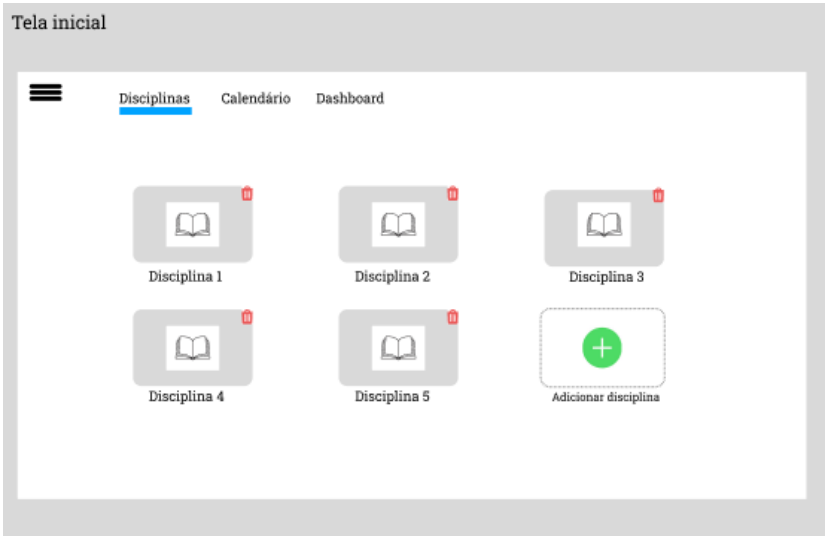
Figura 12 - Mockup para adicionar as disciplinas



Fonte: Elaboração do autor, 2022

Tela das disciplinas (Requisito R12)

Figura 13 - Mockup da tela de disciplinas



Fonte: Elaboração do autor, 2022

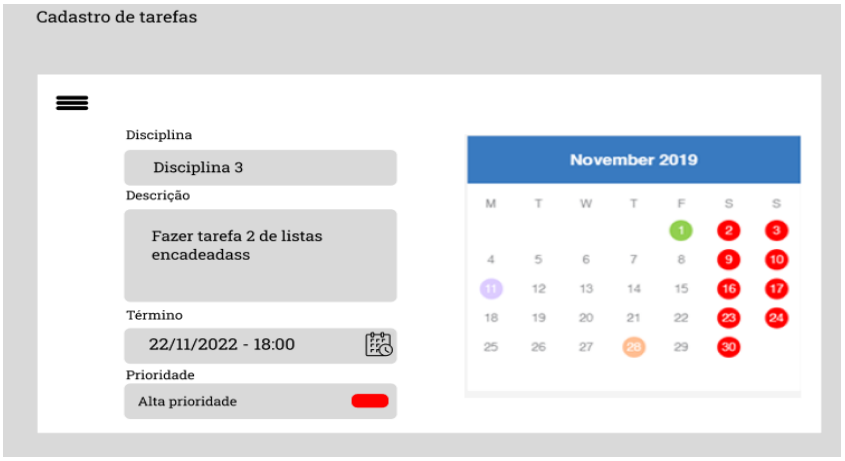
Tela de detalhes de uma disciplina (Requisito R11)

Figura 14 - Mockup da tela detalhes de uma disciplina

Fonte: Elaboração do autor, 2022

Cadastro de tarefas no calendário (Requisito R6)

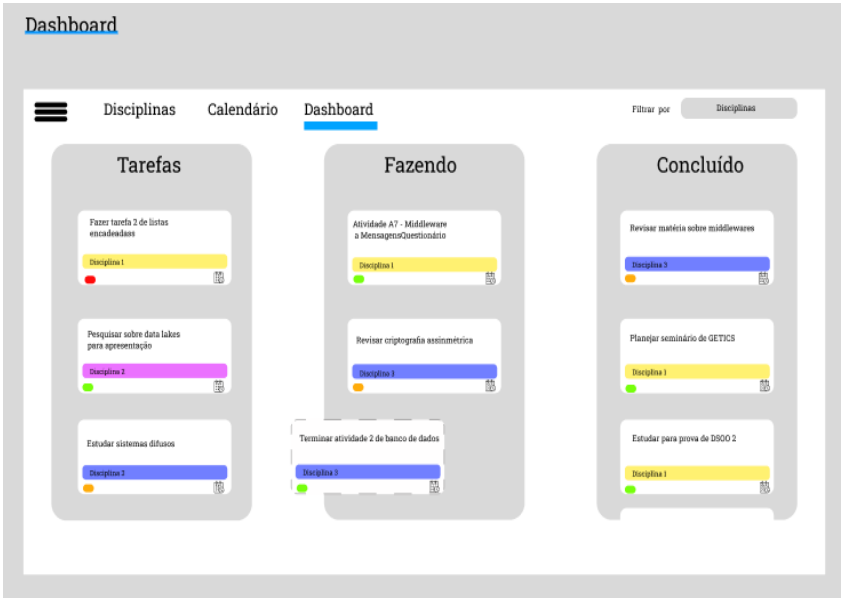
Figura 15 - Mockup da tela de cadastro de uma tarefa no calendário



Fonte: Elaboração do autor, 2022

Board (Requisito R7)

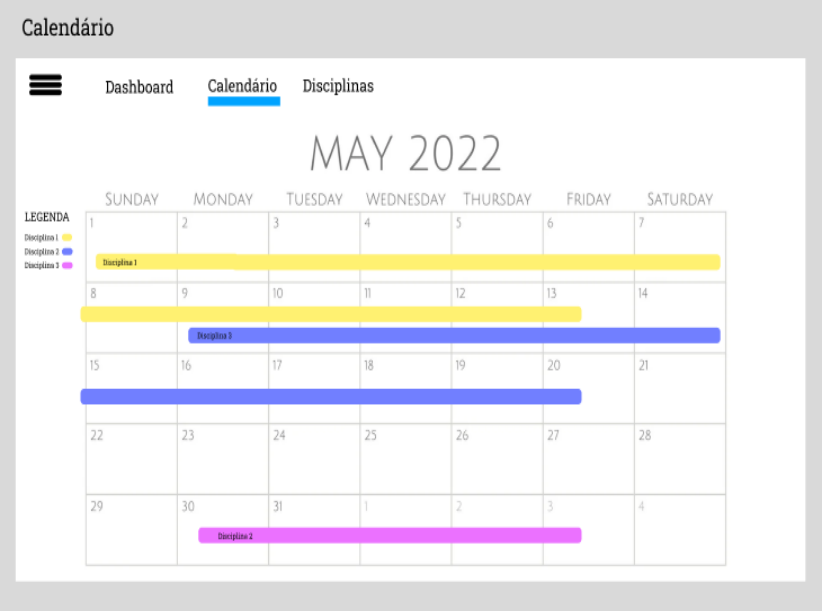
Figura 16 - Mockup da tela de board



Fonte: Elaboração do autor, 2022

Calendário (Requisito R6)

Figura 17 - Mockup da tela de calendário



Fonte: Elaboração do autor, 2022

4.3 Proposta de projeto

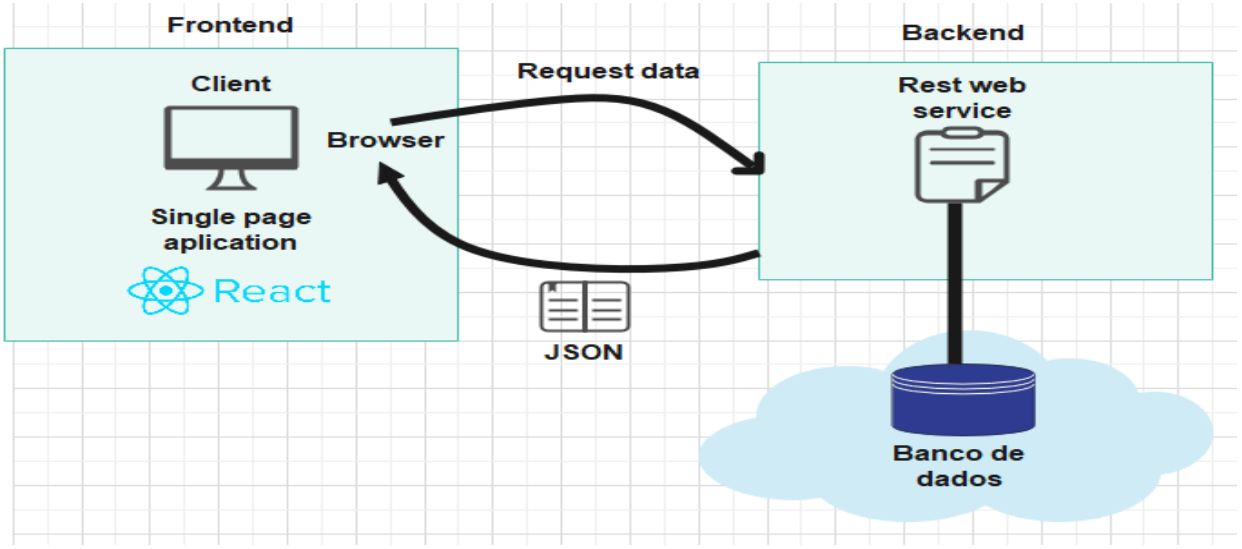
A seguir será apresentada uma proposta de como será executado o projeto em termos arquiteturais de software.

4.3.1 Arquitetura e tecnologias

À luz dos requisitos registrados no item 5.2.1, foi modelada a seguinte arquitetura para dar suporte a um sistema como o proposto:

Aplicação Web

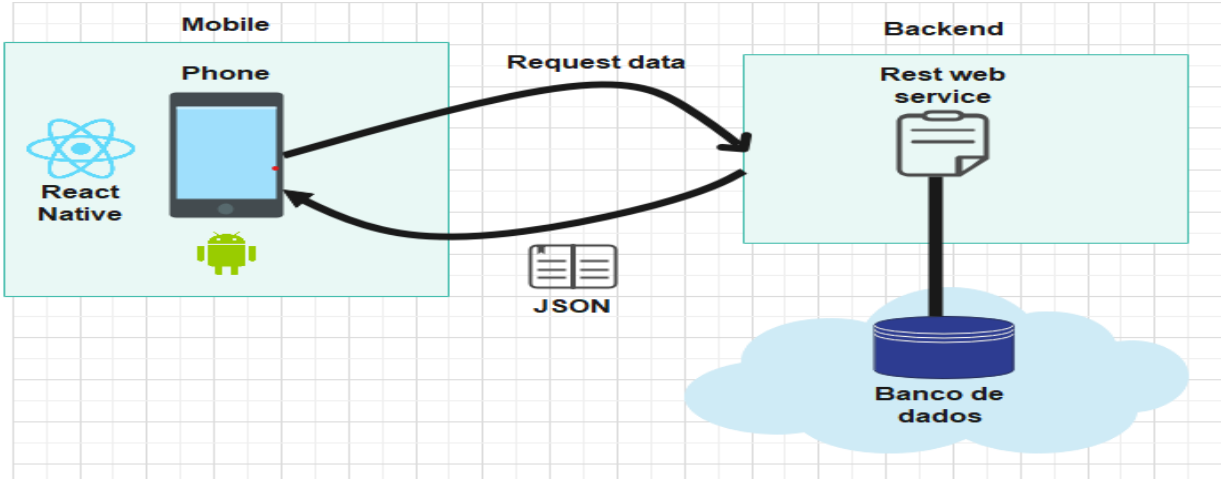
Figura 18 - Arquitetura da aplicação web



Fonte: Elaboração do autor, 2022

Aplicação Mobile

Figura 19 - Arquitetura da aplicação mobile



Fonte: Elaboração do autor, 2022

Pode-se observar na imagem acima, uma arquitetura de cliente servidor, onde o frontend será um Single Page Application (SPA) e o backend utilizará Representational State Transfer (REST) para a construção dos webservices que por sua vez “entregará” os dados no formato de JavaScript Object Notation (JSON) para o Cliente.

Já na aplicação mobile haverá uma arquitetura muito parecida onde a comunicação será feita de forma igual a da web através de webservices e também recebendo um JSON como resposta.

4.4 FRONTEND

O front-end da aplicação foi desenvolvido utilizando o editor de código Visual Studio e utilizando a linguagem Javascript juntamente com a biblioteca para criação de interfaces ReactJs pois essa tecnologia apresenta diversas vantagens como performance, velocidade, usabilidade, fácil de se aprender, além de ter uma grande comunidade e apresentar a maior porcentagem de frameworks mais amados pelos desenvolvedores segundo pesquisa[5] realizada pelo site stack-overflow. Para facilitar a criação dos componentes, foi utilizado a biblioteca ant design e styled components para facilitar na estilização.

Para facilitar na explicação de cada desafio técnico tido, será apresentada uma separação em cada tela do sistema:

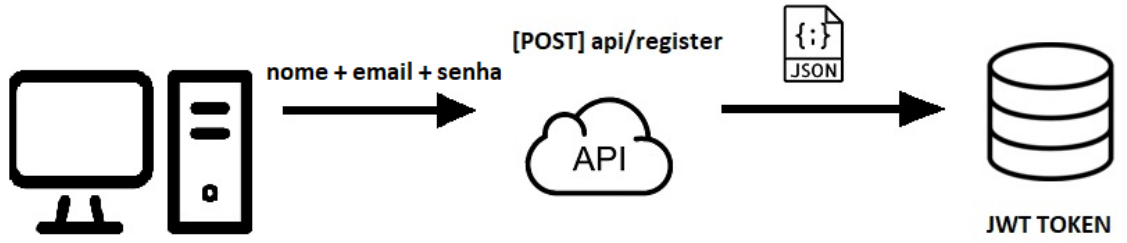
- Tela de Login
- Tela de Board
- Tela de Disciplina
- Tela de Anotações
- Tela de agenda

4.4.1 Tela de Login e Cadastro

O login é realizado através de uma interface simples contendo os campos de email e senha, o mesmo para o cadastro porém com nome, email e senha, não foi feita nenhuma verificação de confirmação de email, algo comum entre as aplicações, já que segurança não é o foco deste trabalho e sim foi feito algo mais simplificado a fim de obter um MVP (Minimum Viable Product). Abaixo pode-se observar um esquema como funciona a lógica no frontend para que o usuário consiga logar na plataforma:

[5] Disponível em: https://survey.stackoverflow.co/2022/#technology

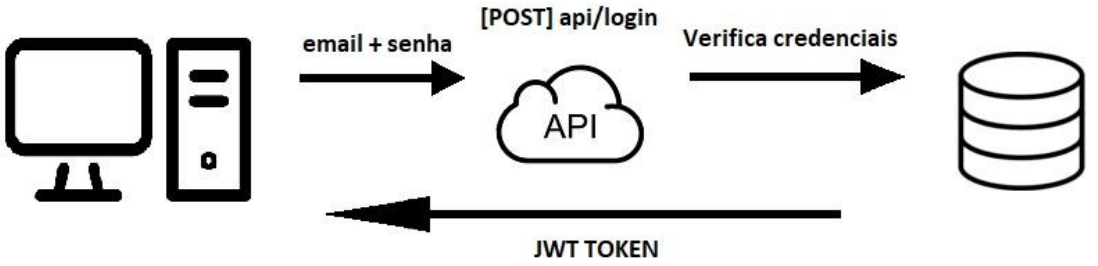
Figura 20 - Esquema do processo de cadastro



Fonte: Elaboração do autor, 2022

Como pode-se observar na imagem acima o usuário insere o nome, email e senha, estes são enviados para a api que é salvo no formato JSON para o banco de dados, por fim utilizado o algoritmo de criptografia, um JWT (JSON Web Token) é gerado e salvo no banco de dados.

Figura 21 - Esquema do processo de login

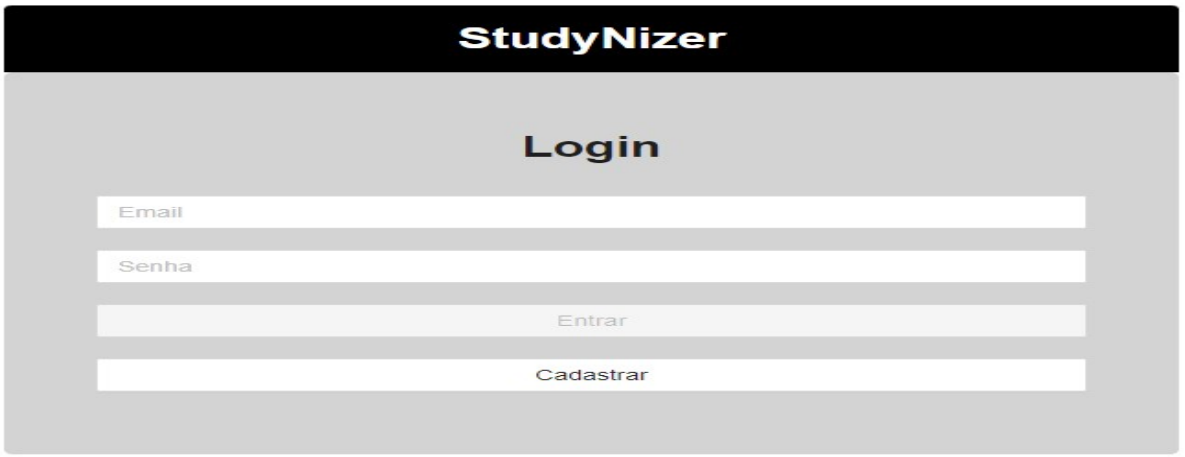


Fonte: Elaboração do autor, 2022

De maneira similar para o login, o usuário insere o email e senha, a api verifica as credenciais, se forem válidas, um JWT token é enviado para o usuário através de uma requisição HTTP (Hypertext Transfer Protocol). Para que o frontend identifique se este possui o JWT. Foi utilizado o local storage do navegador e assim feita uma verificação através das rotas da aplicação para que ele seja redirecionado para a primeira tela principal do sistema. Como é mostrado no **anexo 1,** após a requisição ser feita é salvo no local storage através da chave *@StudyNizer:userSession* que será utilizada em seguida para o redirecionamento de rota da aplicação. No arquivo *Router* em **anexo 2** observa-se que o token é pego através do local storage e assim enviado para o componente AuthLayout em **anexo 3** que é responsável por fazer a verificação da disponibilidade do Token no frontend e também o redirecionamento para a tela principal da aplicação.

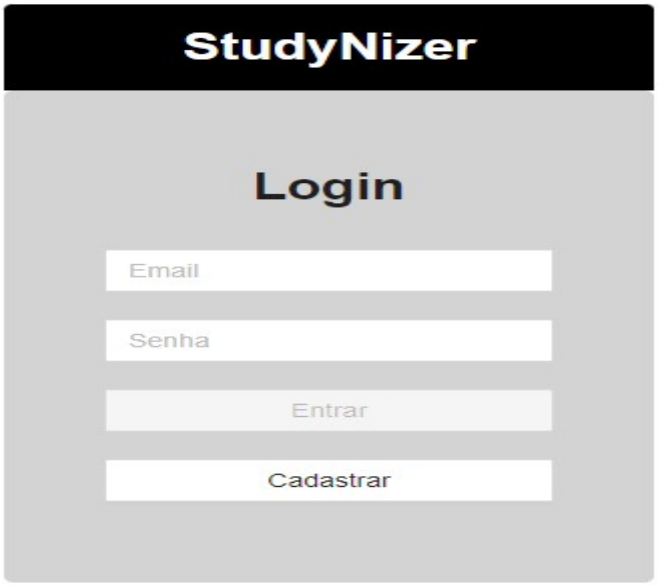
Abaixo a tela de login na versão para web desktop e mobile.

Figura 22 - Tela de login do StudyNizer



Fonte: Elaboração do autor, 2022

Figura 23 - Tela de login do StudyNizer versão mobile

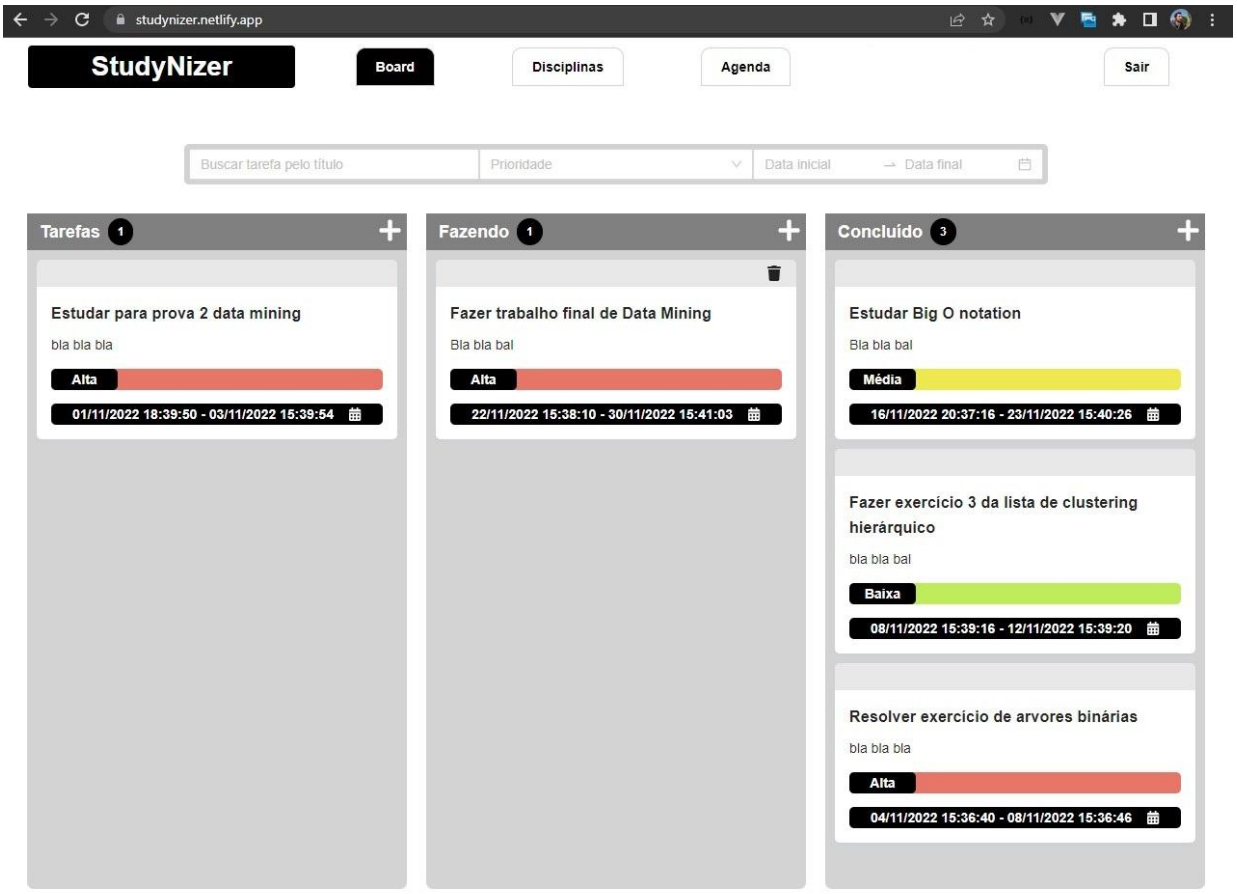


Fonte: Elaboração do autor, 2022

4.4.2 Tela de Board

Uma vez que o usuário tem suas credenciais validadas este é redirecionado para o board, onde ele irá fazer o acompanhamento das tarefas. A tela foi desenvolvida utilizando a biblioteca *react-beautiful-dnd*, criada pela empresa atlassian, a qual facilita a escrita do código para a funcionalidade de *drag-and-drop,* que é fundamental para o quadro Kanban. Nesta tela é feito um CRUD (Create, Read, Update and Delete) completo, o usuário pode adicionar, editar, deletar e listar tarefas, além de filtrar os cards pelo título, nível de prioridade e data. A seguir são apresentadas as telas desta parte do sistema:

Figura 24 - Tela de board do StudyNizer



Fonte: Elaboração do autor, 2022

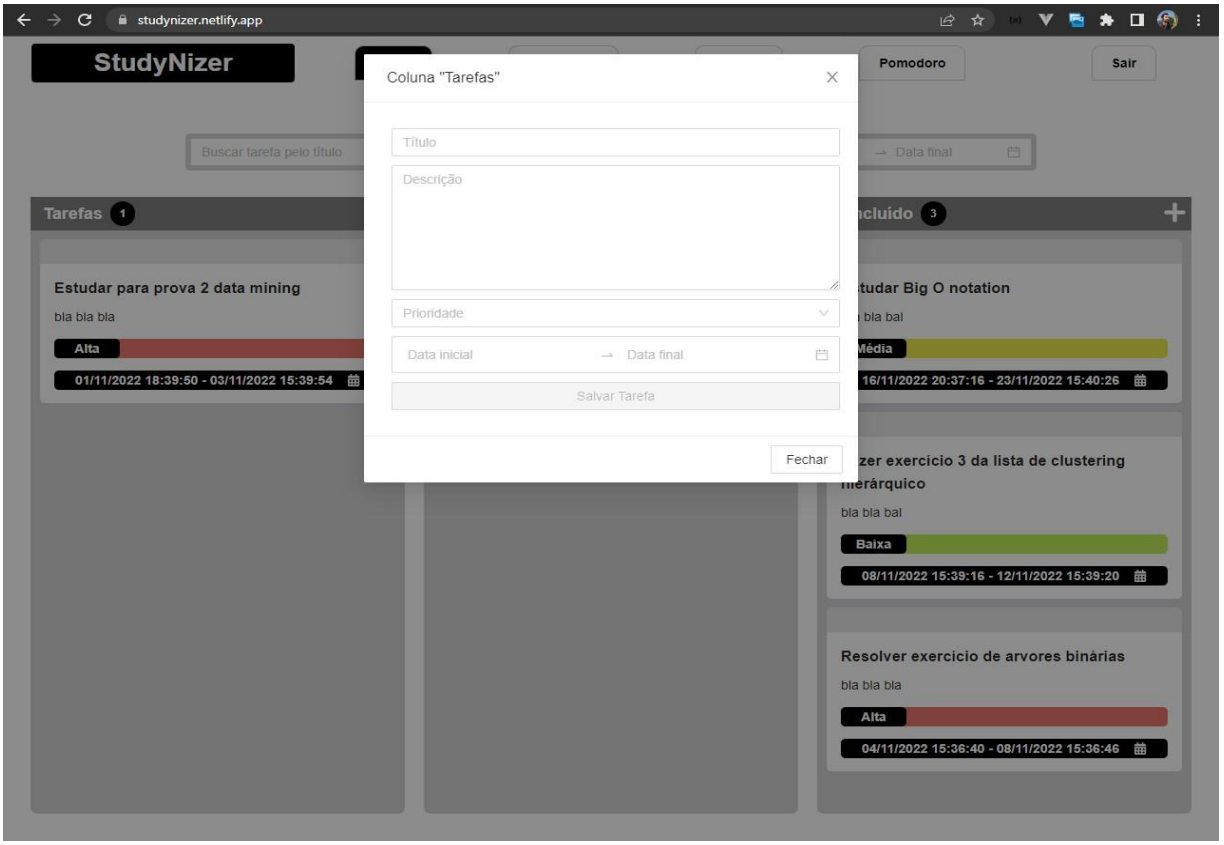
O board apresenta três colunas de nomes intuitivos para cada status de progresso , que são:

- Tarefas: Representa os cards que o usuário precisa fazer mas ainda não deu início.
- Fazendo: Representa os cards que o usuário começou a fazer.
- Concluído: Representa os cards que o usuário finalizou e serve também como um histórico.

Ao clicar no ícone de “adição” no topo da coluna um *modal* se abre com os campos: Título, Descrição, Prioridade e Data de início e Fim para serem atribuídos a uma tarefa. Ao passar o mouse por cima de uma card um tooltip

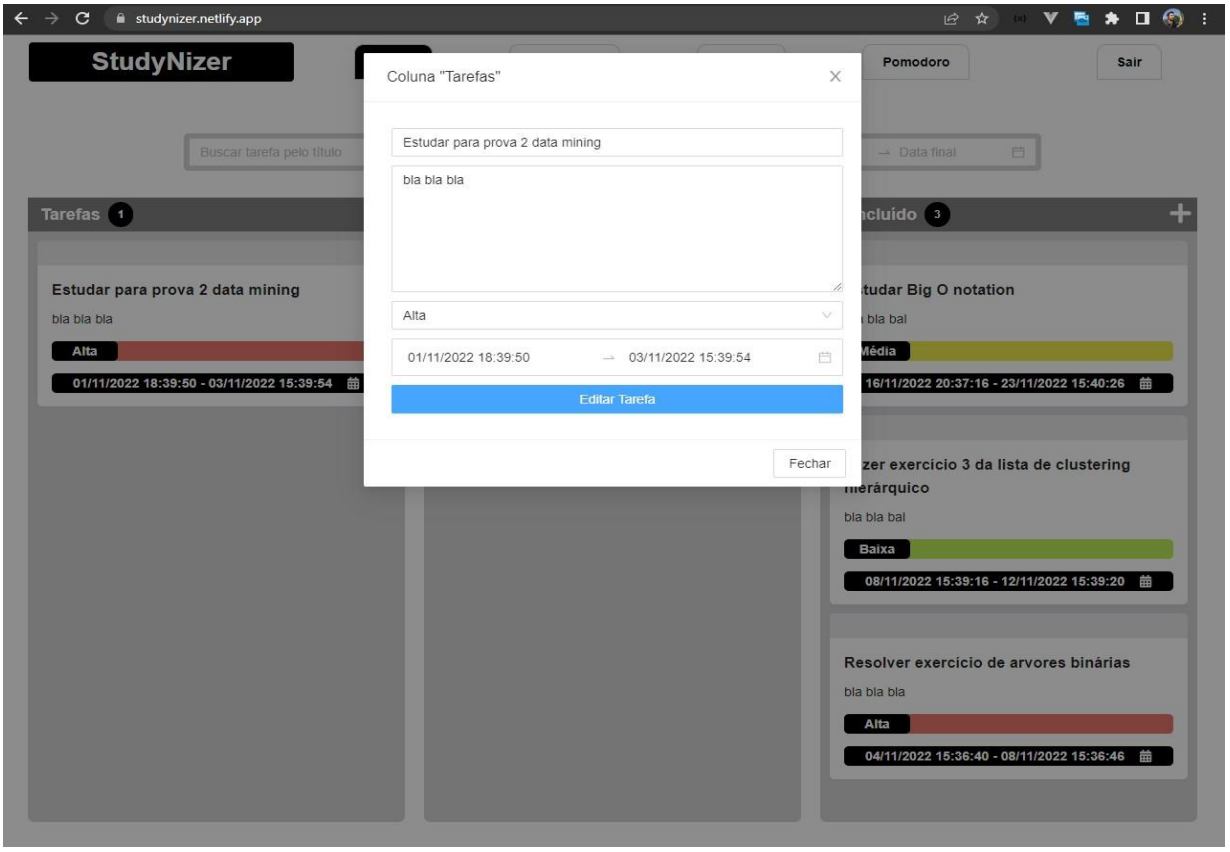
aparece com uma mensagem “Clique para ver detalhes desta tarefa”, com os dados pré-preenchidos do card selecionado e assim o usuário consegue editar as informações se necessário.

Figura 25 - Modal para adicionar uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

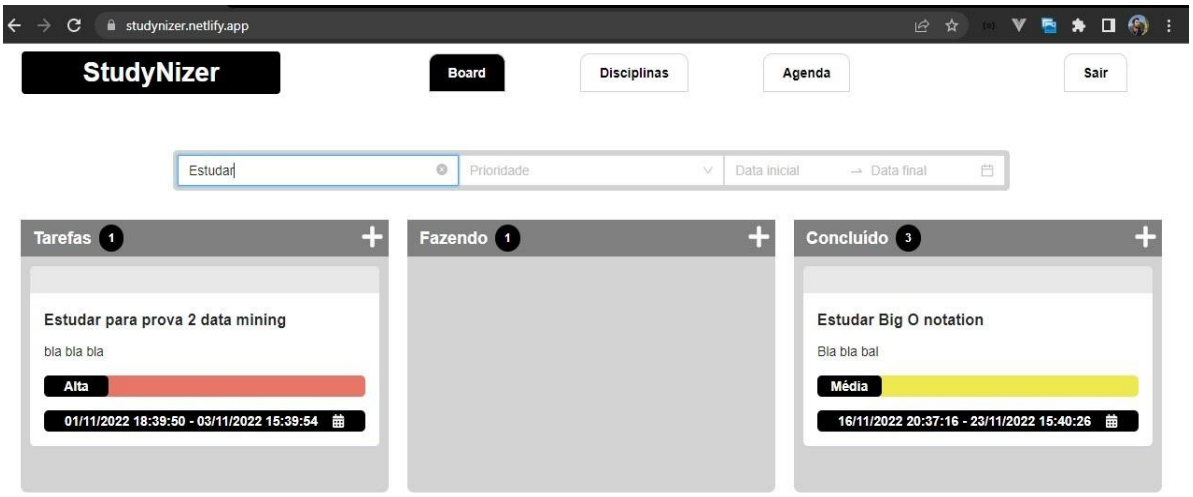
Figura 26 - Modal para editar uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

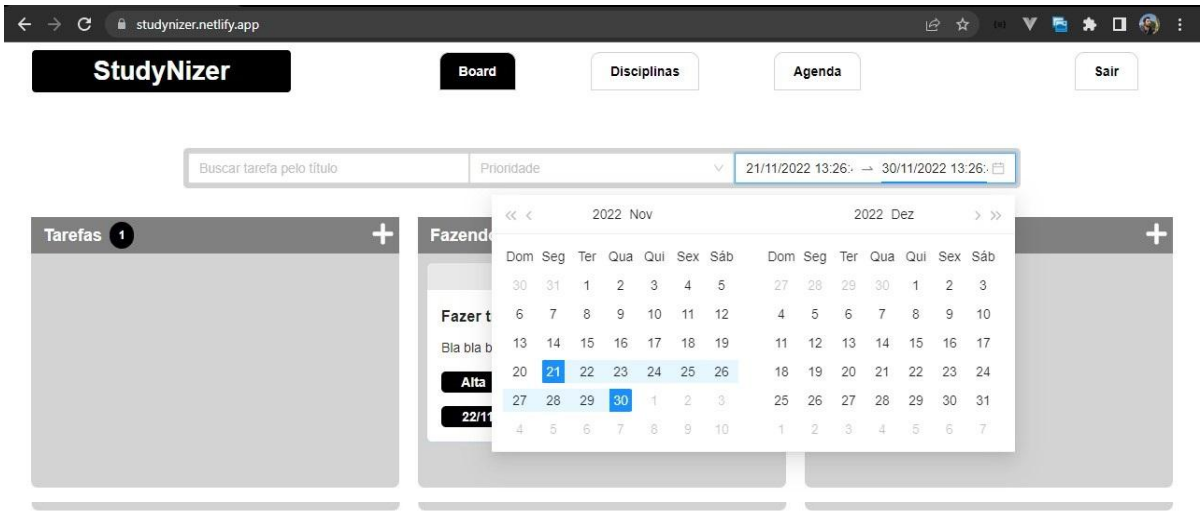
As próximas imagens são referentes a funcionalidade de filtro dos cards onde é possível filtrar pelo título, prioridade e data.

Figura 27 - Filtro por título de uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

Figura 28 - Filtro por data de uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

O código para realizar a função de *drag-and-drop* pode ser separada em dois momentos, o primeiro sendo a mudança de um card de uma coluna para outra e assim salvando no estado da aplicação, foi utilizado a api de hook do react *useState* que serve para “salvar na memória" uma certa informação e poder capturá-la ou modificá-la dependo da necessidade. Como pode-se observar no **anexo 3**, tem-se uma estrutura *json* em um formato chave valor

para cada coluna. Na função *handleDragEnd* em **anexo 4** é onde é feita a lógica para identificar onde deve-se posicionar os cards e salvá-los no estado, utilizando o *setState.*

O segundo momento seria salvar no banco de dados, no backend existe três *endpoints* para cada coluna:

- [POST] /board-tasks-todo
- [POST] /board-tasks-doing
- [POST] /board-tasks-completed

Foi realizado uma lógica no frontend para identificar a partir da detecção do *hover* do mouse do usuário qual coluna deve-se inserir o card selecionado e assim acionar o enpoint para salvar no banco de dados, como pode ser observado no **anexo 5**.

4.4.3 Tela de Disciplina

A tela de disciplina é onde o usuário irá registrar o nome das matérias para em seguida ter os resumos de cada uma. As funcionalidades desta tela permitem a adição de uma disciplina, excluir, importar e exportar os arquivos com seus respectivos resumos para que um usuário possa compartilhar seus resumos com um colega.

Conforme as disciplinas são adicionadas, estas são apresentadas em forma de cards, como pode-se observar abaixo também há dois botões para adicionar e importar uma disciplina. Ao passar o mouse sobre um dos cards fica visível para o usuário dois ícones, um de exclusão e outro para exportar a disciplina.

Figura 29 - Tela de disciplinas do StudyNizer

Fonte: Elaboração do autor, 2022

Figura 30 - modal para adicionar uma disciplina do StudyNizer

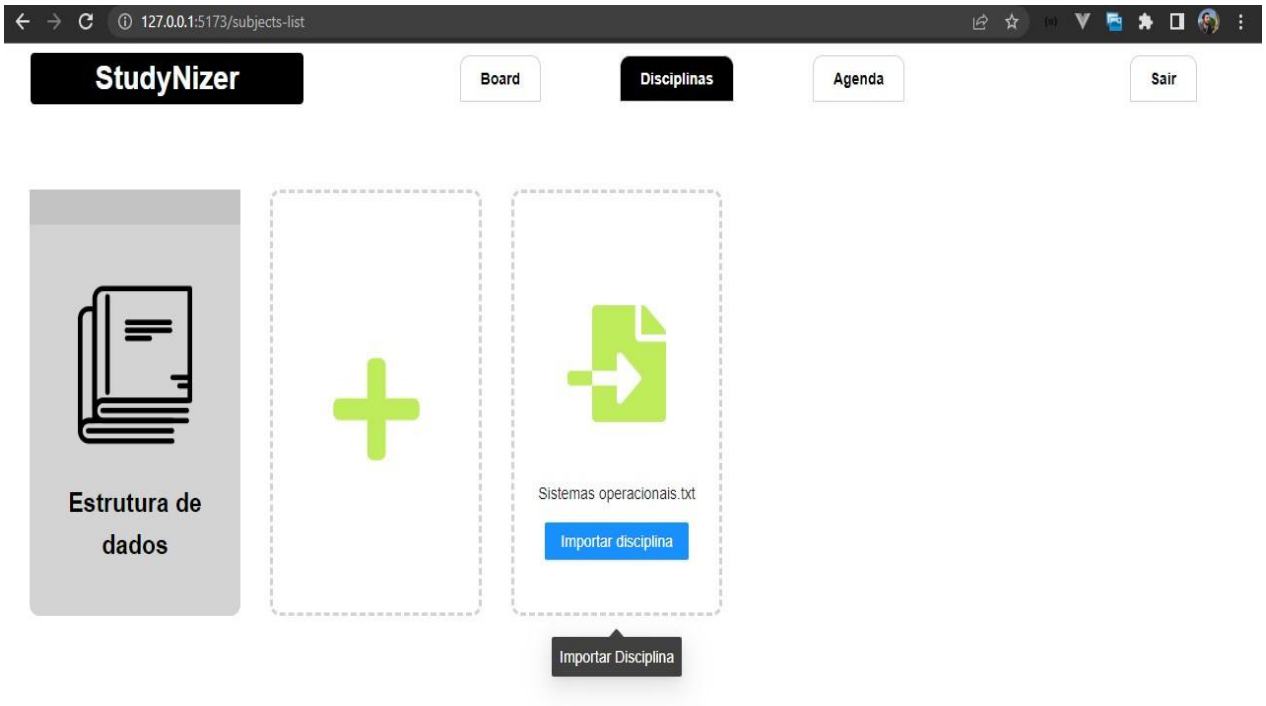
Fonte: Elaboração do autor, 2022

Figura 31 - Exportando uma disciplina

Fonte: Elaboração do autor, 2022

Na imagem abaixo mostra um arquivo em que o usuário selecionou de seu computador para ser importado e em seguida ele terá todos os resumos feitos por um outro usuário.

Figura 32 - Importando uma disciplina



Fonte: Elaboração do autor, 2022

O desafio técnico maior desta etapa foi sem dúvidas a lógica para importar e exportar as disciplinas, foi utilizado a biblioteca *file-saver* que é responsável por salvar arquivos no formato txt. No **anexo 6** podemos ver como isto se faz presente no código.

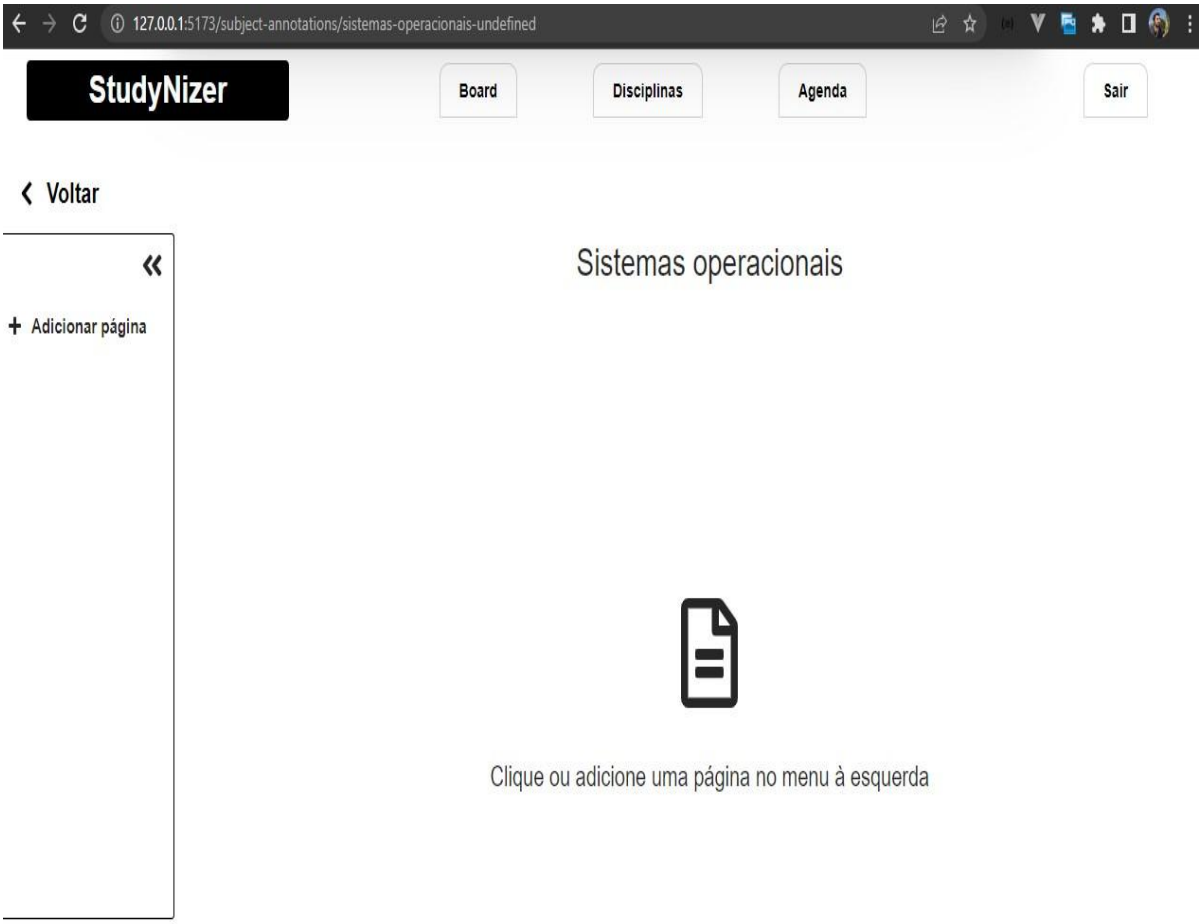
Para a exportação é feita uma chamada ao endpoint [GET] /user/markdown/:id, através deste tem-se o conteúdo das anotações de cada disciplina e com isso foi convertido a resposta da requisição para o formato *blob,* que será usado como primeiro parâmetro na função *FileSaver.saveAs* e como segundo o nome do arquivo.

Na parte da importação, como pode ser observado no **anexo 7**, foi realizada uma chamada para o endpoint [POST] /user/markdownImport que irá receber o conteúdo do arquivo exportado, também foi feita uma verificação para saber se a disciplina já foi importada.

4.4.4 Tela de Anotações

Ao clicar em uma matéria na tela de disciplina, o usuário é redirecionado para a tela de anotações, nesta o aluno poderá adicionar uma página para um determinado resumo, em seguida tem disponível uma área com texto rico. A configuração inicial desta seção indica o que o usuário deve fazer, neste caso adicionar uma página no menu lateral.

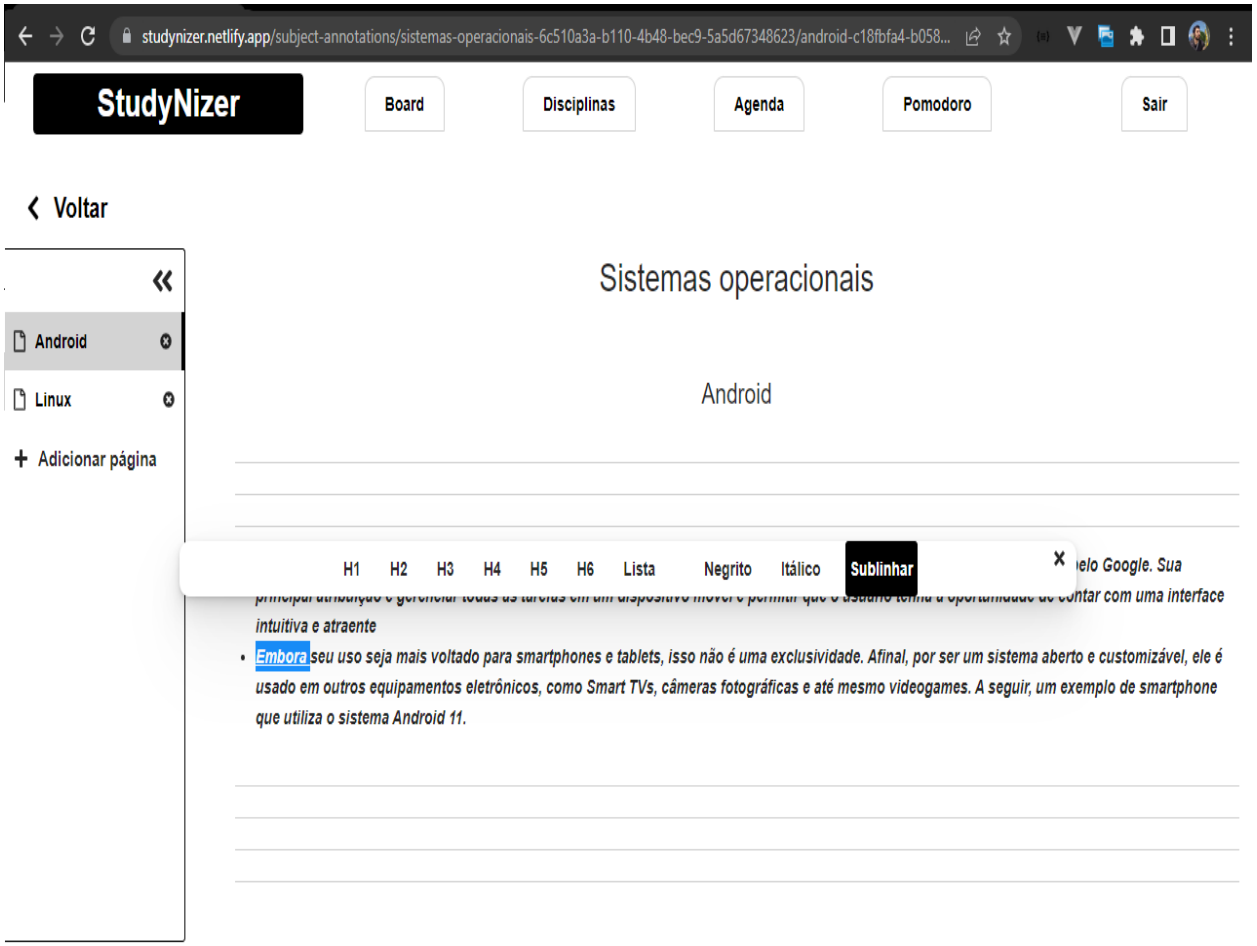
Figura 33 - Tela de anotações do StudyNizer



Fonte: Elaboração do autor, 2022

Adicionada a página, obtém-se agora disponível uma área de texto rico para escrever, ao selecionar uma linha, surge um painel para o usuário customizar seus resumos com as opções para aumentar o tamanho da fonte, criar listas, negrito, itálico e sublinhado, ao digitar este painel some para não interferir na visão enquanto o usuário digita.

Figura 34 - Caixa de texto rico



Fonte: Elaboração do autor, 2022

Foi utilizada a biblioteca *draft-js* como pode-se observar no **anexo 8,** para o texto rico, o componente <Editor /> representa o campo de texto e com a propriedade editorState capturamos o conteúdo escrito pelo usuário na função handleChangeEdtior é onde é feita a lógica para salvar no banco de dados.

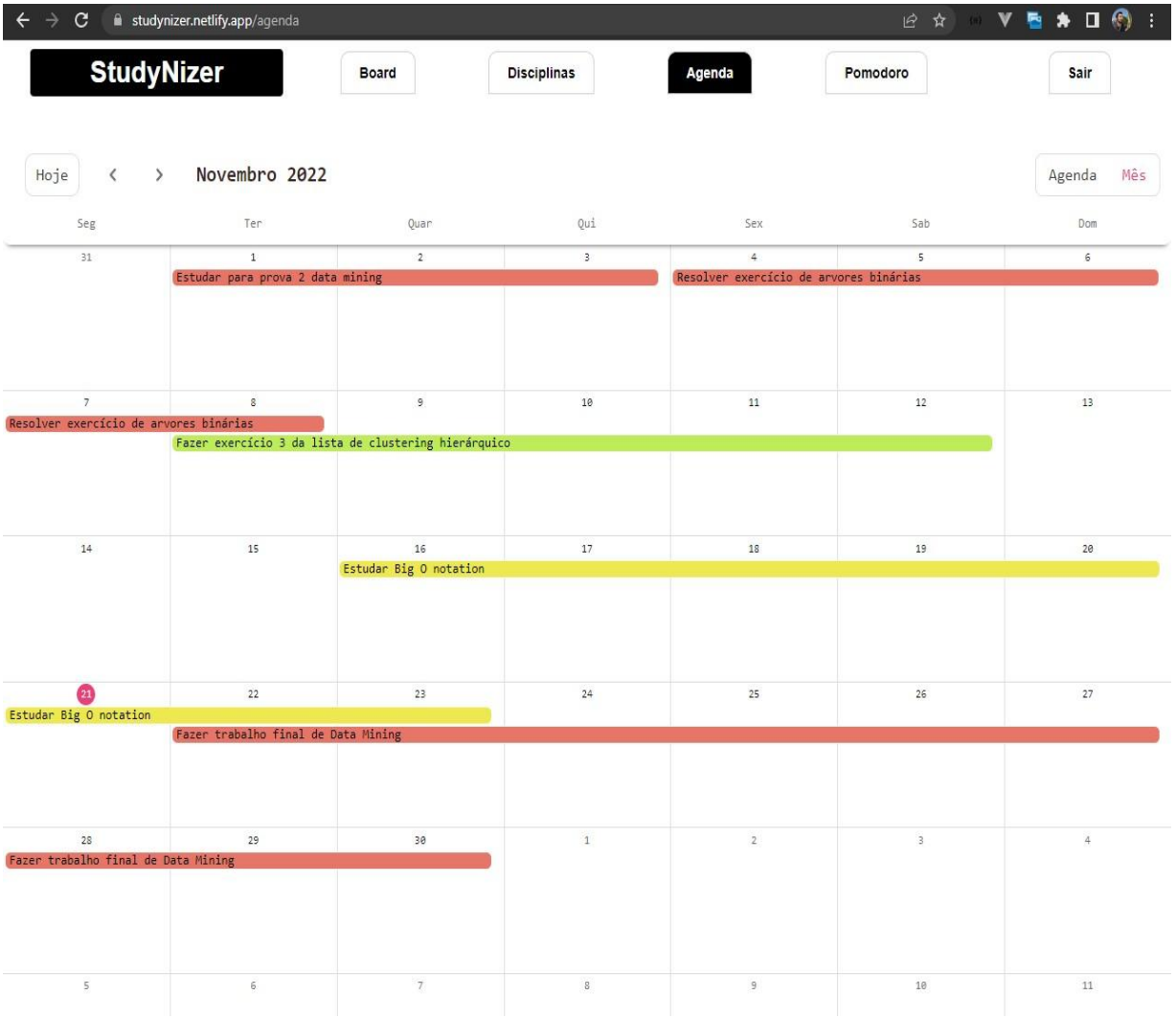
No código mostrado do **anexo 9,** o conteúdo do texto é passado para uma função converRaw da biblioteca *draft-js,* isso irá converter as informações para um formato de texto, para ser aceito pelo banco de dados.

É feita uma chamada para uma função de *debounce* que é comumente utilizada em situações onde precisa-se salvar o conteúdo digitado pelo usuário apenas uma vez quando este terminar de digitar, neste caso após um segundo. Desta forma prevenimos chamadas abundantes para a api [PUT] /user/markdown/:id, onde é feito o registro do conteúdo do texto no banco de dados como observado no **anexo 10**.

4.4.5 Tela de Agenda

Ao adicionar uma tarefa na tela de board, estas aparecerão automaticamente na agenda, a qual possui dois modos de visualização, “agenda” e “Mês”, No modo agenda as tarefas são listadas em formato vertical com suas respectivas datas, horários, nível de prioridade representado por cores e o título.

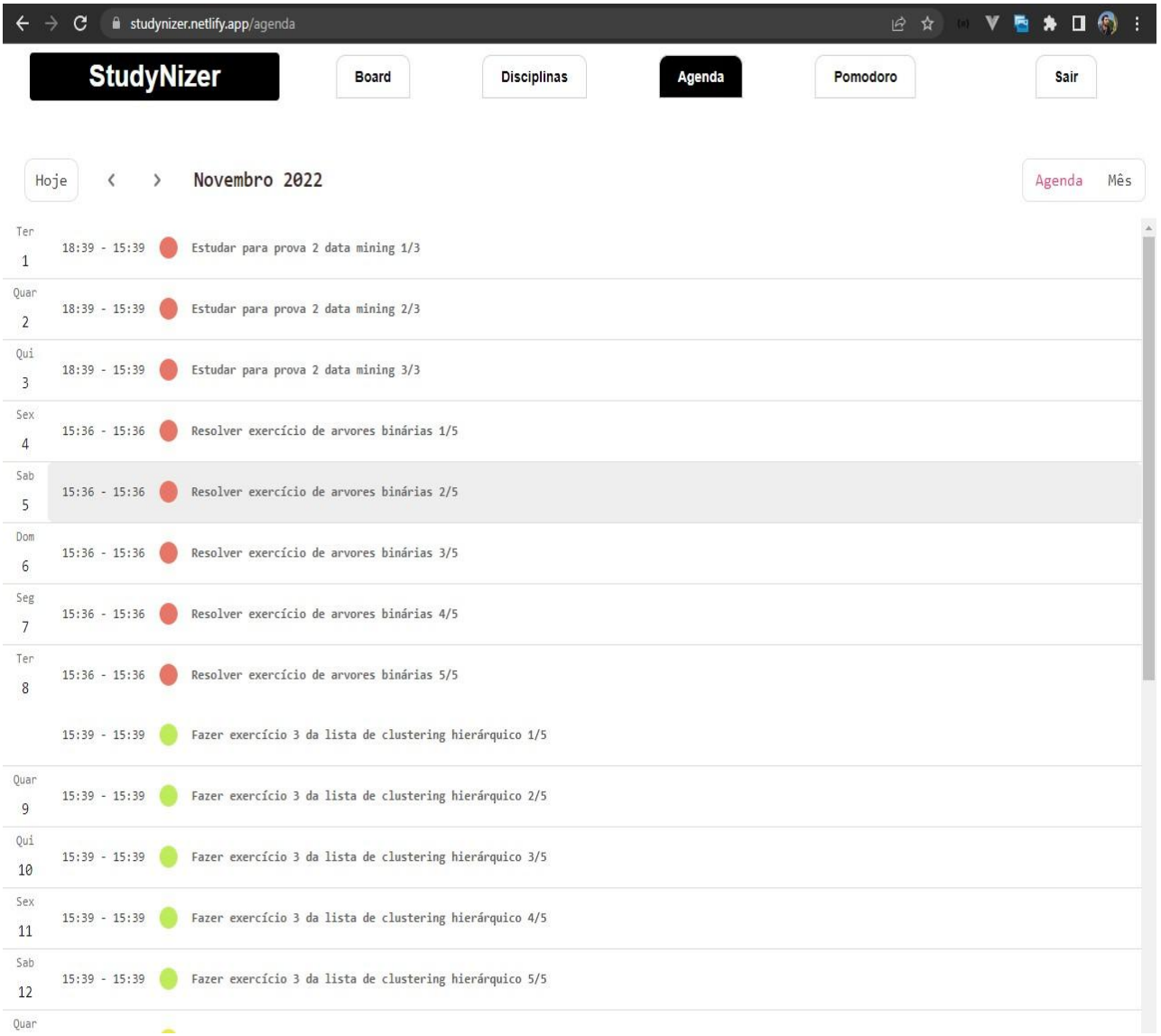
Figura 35 - Tela de calendário do StudyNizer



Fonte: Elaboração do autor, 2022

Já no modo de mês observa-se uma visualização em modo de calendário, representada por linhas, desta forma o usuário tem uma visão melhor das tarefas a serem feitas ao decorrer dos dias.

Figura 36 - Tela de agenda do StudyNizer



Fonte: Elaboração do autor, 2022

Foi utilizada a biblioteca *kalend* para a implantação do calendário, uma de suas propriedades deste componente é o *events* onde é passado as tarefas e assim listados para o usuário como observado no código do **anexo 11**.

Para disponibilizar as tarefas no calendário foi realizada três chamadas aos endpoints:

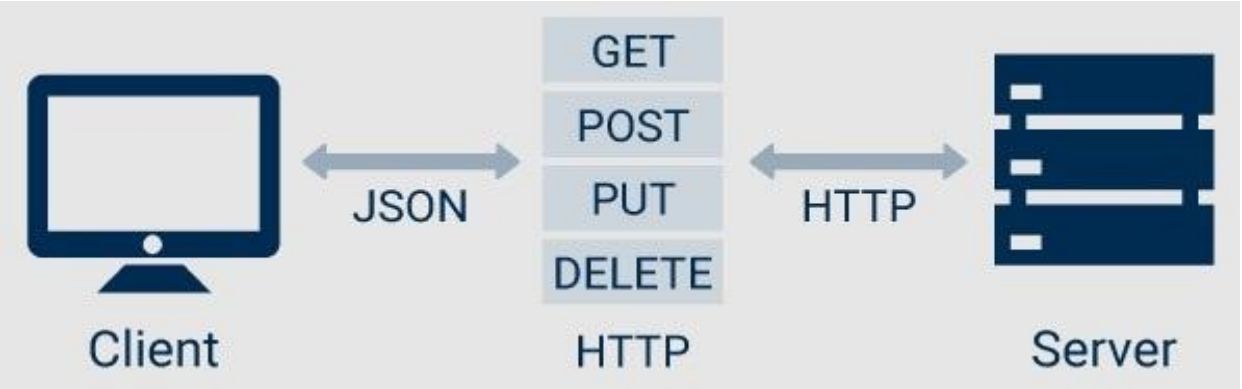
- [GET] /user/board/tasks-todo
- [GET] /user/board/tasks-doing
- [GET] /user/board/tasks-completed

Em seguida foi passado a resposta destas requisições para a função *generateDemoEvents* que é responsável por preencher as tarefas no calendário como mostrado no **anexo 12**.

4.5 BACKEND

Para o backend da aplicação, foram utilizadas ferramentas que são hoje populares no mercado e além de tudo performáticas, além de usar Rest como arquitetura de software. As API's (Application Programming Interface) vêm crescendo rapidamente. Os desenvolvedores passaram do SOAP (Simple Object Access Protocol) ou RPC (Remote Procedure Call) para implantar o REST (Representational State Transfer), como o meios para os consumidores usarem seus serviços. Isto é corroborado por grandes sites como Google, Facebook ou Twitter, que agora estão implantando serviços REST para fornecer acesso fácil a seus recursos. Esta arquitetura fornece padrões entre sistemas de computador na web, tornando mais fácil para os sistemas se comunicarem uns com os outros. Os sistemas compatíveis com REST, geralmente chamados de sistemas RESTful, seguem os "cinco verbos RESTFUL" ou os "cinco princípios RESTFUL" para garantir a integridade e a interoperabilidade dos dados trocados entre clientes e servidores. Um sistema não é RESTFULL se não seguir os cinco princípios. Por exemplo, se um sistema mantém o estado entre as requisições ou não usa verbos HTTP para operar sobre recursos, ele não é RESTFUL. (NEUMANN, 2018).

Figura 37 - Esquema do fluxo Cliente servidor



Fonte: https://holdertech.com.br

Outra tecnologia utilizada foi o NodeJs que por ser escrita em javascript facilita a implementação do projeto principalmente para desenvolvedores com experiência em front-end, é muito utilizada entre diversas startups e também empresas de grande porte, Possui uma arquitetura orientada a eventos capaz de implementar eventos de E/S assíncrona (é uma forma de processamento de entrada/saída que permite que outro processamento continue antes da transmissão). Destina-se a otimizar a escalabilidade em aplicações web (DAHL, 2009).

Para a comunicação com o banco de dados foi utilizado o postgresql, que é um sistema de banco de dados objeto-relacional de código aberto que usa e estende a linguagem SQL. Conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para oferecer consistentemente soluções inovadoras e de alto desempenho, é executado em todos os principais sistemas operacionais. (STONEBRAKER, 1986).

Para a interação e testes com as APIs foi utilizado um aplicativo desktop chamado Insomnia[6], uma tecnologia multi plataforma gratuita que facilita a interação e o design de API's baseadas em HTTP. O Insomnia combina uma interface fácil de usar com funcionalidades avançadas, como auxiliares de autenticação, geração de código e variáveis de ambiente, é primordial a utilização de tal aplicação para ajudar a descobrir bugs, anomalias ou discrepâncias do comportamento esperado de uma API durante o desenvolvimento (KONG, 2017).

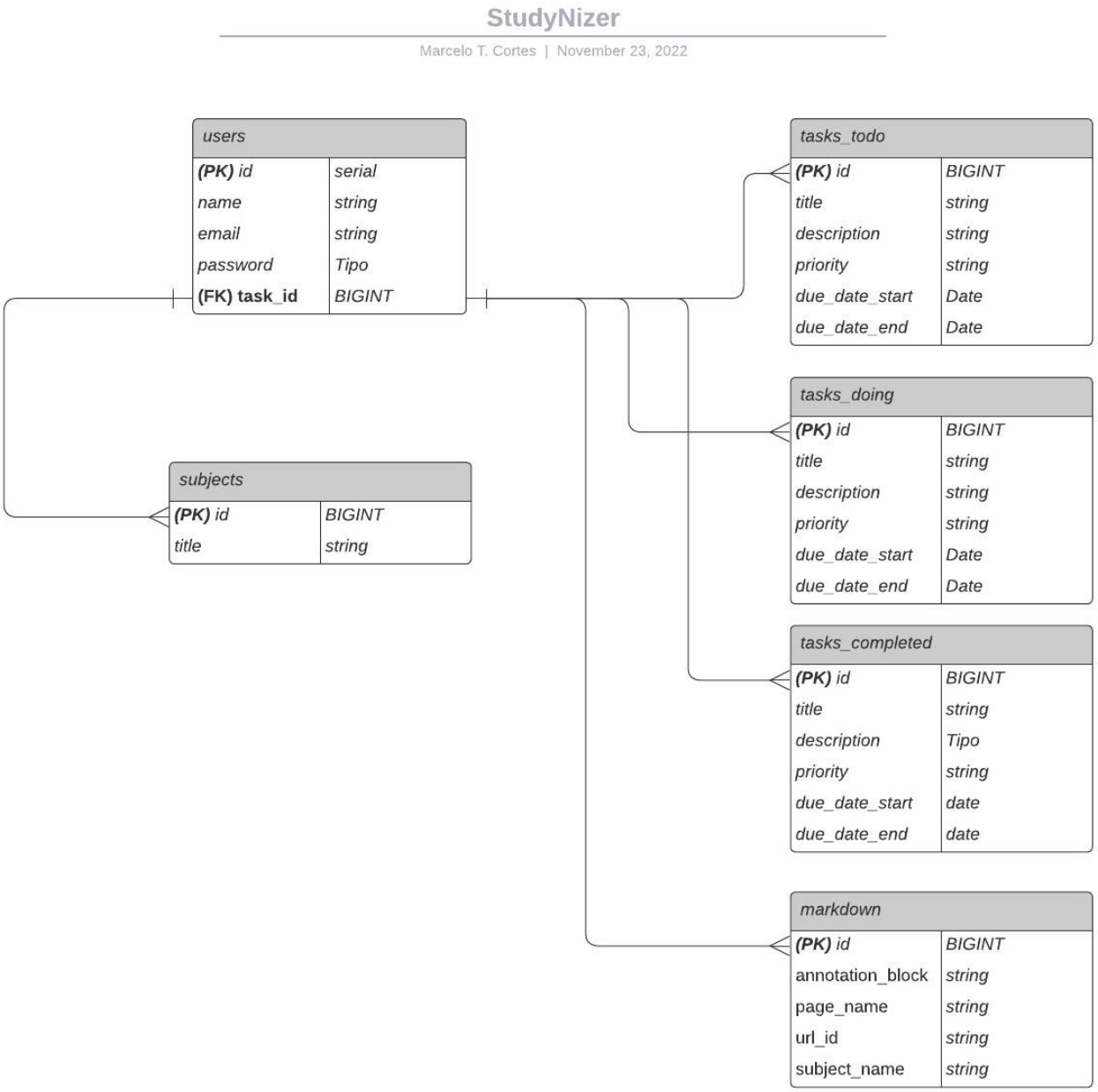
A modelagem do banco de dados se fez com as seguintes tabelas:

- users: Responsável por guardar os usuários.
- subjects: Responsável por guardar as disciplinas de cada usuário..
- tasks_todo: Responsável por guardar as tarefas que precisam ser feitas..
- tasks_doing: Responsável por guardar as tarefas que estão sendo feitas..
- tasks_completed: Responsável por guardar as tarefas que foram feitas.
- markdown: Responsável por guardar o conteúdo dos resumos das disciplinas.

[6] Disponível em: https://insomnia.rest

Como pode-se observar a tabela de usuário possui um relacionamento de um para muitos com os demais através de sua chave estrangeira com a chave primária.

Figura 38 - Diagrama de banco de dados do StudyNizer



Fonte: Elaboração do autor, 2022

5 Tabela comparativa das funcionalidades das aplicações

A seguir é apresentada uma tabela comparativa entre as aplicações exploradas onde o principal objetivo é analisar a viabilidade destas ferramentas serem pensadas especificamente para um estudante, englobando critérios como UX e UI, facilidade de uso e funcionalidades que façam sentido para este tipo de usuário.

Tabela 3 - Tabela comparativa entre as plataformas

Aplicação	Fácil de usar para um único usuário?	É pensado para utilização de um estudante e possui funcionalidades relacionadas com esta área?	Possui uma boa UX e UI?	Permite fazer anotações?	Possui quadro Kanban?	Possui Calendário?
Trello	Sim	Sim	Não	Não	Sim	Sim
Jira	Não	Não	Não	Sim	Sim	Sim
Notion	Sim	Sim	Sim	Sim	Sim	Sim
ClickUp	Não	Não	Não	Sim	Sim	Sim
Monday.com	Não	Não	Não	Sim	Sim	Sim

Kanban Tool	Sim	Sim	Sim	Não	Sim	Sim
Asana	Não	Não	Não	Sim	Sim	Sim
StudyNizer	Sim	Sim	Sim	Sim	Sim	Sim

Fonte: Elaboração do autor, 2022

Analisando a tabela acima percebe-se que nenhuma das aplicações preenche todos os requisitos, já o software deste trabalho possui todas essas condições, se enquadrando desta forma em uma ferramenta pensada exclusivamente para um estudante dando assim o suporte necessário para uma boa experiência durante sua vida acadêmica.

6 Conclusão

O intuito do presente trabalho foi mostrar que hoje no mercado existem diversas aplicações para organização pessoal com múltiplas funcionalidades, porém a maioria delas é pensada em grupos de trabalho, não sendo voltadas para apenas um único usuário, desta forma a dificuldade de utilização desses softwares atreladas a problemas de UI e UX faz com que o estudante possa ter dificuldade ao utilizar o sistema. Este estudo abordou a necessidade de juntar funcionalidades de kanban, anotações e calendário em uma única plataforma. Atualmente, existem plataformas excelentes para cada uma dessas funcionalidades, mas elas geralmente são oferecidas de forma isolada, o que requer que os usuários tenham várias contas diferentes e não tenham acesso a todas essas utilidades em um só lugar. A junção dessas funcionalidades em uma única plataforma permite com que os usuários tenham tudo o que precisam em um único local, aumentando a praticidade e a eficiência.

O protótipo desenvolvido pôde mostrar que uma plataforma que possui uma interface minimalista e com funcionalidades focadas para a vida estudantil, pode ser muito útil durante a vida acadêmica daqueles que porventura irão utilizá-la. As tecnologias também aplicadas fazem com que a aplicação seja performática, dando uma melhor experiência para os usuários.

Por fim, espera-se que este trabalho tenha conseguido dar uma ideia de como um software de organização deve ser aplicado às necessidades específicas de um estudante e assim fazer com que estes obtenham um melhor resultado ao longo de seus cursos.

Para trabalhos futuros algumas funcionalidades podem ser adicionadas:

- Integração com o moodle para que os alunos consigam importar as disciplinas que estão fazendo no momento.
- Desenvolvimento de uma aplicação móvel para dispositivos android e IOS.
- Acrescentar mais funcionalidades na plataforma como sistema de notificações, pomodoro, chat por texto e vídeo conferência.
- Ser possível adicionar colegas.
- Fóruns de discussões para disciplinas.

REFERÊNCIAS

BASSO, Cláudia *et al*. **Organização de tempo e métodos de estudo: Oficinas com estudantes universitários**.
Disponível em: http://pepsic.bvsalud.org/pdf/rbop/v14n2/12.pdf. Acesso em: 22 jul. 2022

COCKBURN, Alistair. **Agile software development: The people factor**. Disponível em: https://www.researchgate.net/publication/2955526_Agile_software_development_The_people_factor. Acesso em: 20 jul. 2022.

CEARÁ. SECRETARIA DE EDUCAÇÃO DO ESTADO DO CEARÁ. (ed.). **Gestão Organizacional**: Curso Técnico em administração. Curso Técnico em Administração. 2015. Escola Estadual de Educação Profissional. Disponível em: https://educacaoprofissional.seduc.ce.gov.br/images/material_didatico/administracao/administracao_gestao_organizacional.pdf. Acesso em: 22 jul. 2022

PERNAMBUCO, Ufpe: Universidade Federal de. **Rational Unified Process**. 2005. Disponível em: https://www.cin.ufpe.br/~gta/rup-vc/core.base_concepts/guidances/concepts/task_86A4917F.html. Acesso em: 22 jul. 2022.

MACCANN, Carolyn. **Strategies for success in education**: time management is more important for part-time than full-time community college students. Time management is more important for part-time than full-time community college students. 2010. Disponível em: https://www.sciencedirect.com/science/article/pii/S1041608011001786. Acesso em: 08 jul. 2022.

WAKODE, Rajat B.. **Overview on Kanban Methodology and its Implementation**. 2015. Disponível em: https://www.academia.edu/27885179/Overview_on_Kanban_Methodology_and_its_Implementation. Acesso em: 20 jul. 2022.

ALSAQQA, Samar. **Agile Software Development: Methodologies and Trends**. 2020. Disponível em: https://www.researchgate.net/profile/Samar-Alsaqqa/publication/342848746_Agile_Software_Development_Methodologies_and_Trends/links/5f09bcdfa6fdcc4ca45e36f0/Agile-Software-Development-Methodologies-and-Trends.pdf. Acesso em: 20 jul. 2022.

PEREIRA, Paulo. **Entendendo Scrum para Gerenciar Projetos de Forma Ágil**. 2007. Disponível em: https://www.academia.edu/28954250/Entendendo_Scrum_para_Gerenciar_Projetos_de_Forma_%C3%81gil. Acesso em: 22 jul. 2022.

ARBULU, Roberto. **KANBAN IN CONSTRUCTION**. 2003. Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.875.4062&rep=rep1&type=pdf. Acesso em: 20 jul. 2022.

CORONA, Erika. **A Review of Lean-Kanban Approaches in the Software Development** ERIKA CORONA e FILIPPO EROS PANI. 2013. Disponível em: http://www.wseas.us/journal/pdf/information/2013/5709-110.pdf. Acesso em: 20 jul. 2022.

TELES, Vinícius Manhães. **UM ESTUDO DE CASO DA ADOÇÃO DAS PRÁTICAS E VALORES DO EXTREME PROGRAMMING**. 2005. Disponível em: https://www.ime.usp.br/~ale/Dissertacao.pdf. Acesso em: 22 jul. 2022.

NEUMANN, Andy. **Analysis of Public REST Web Service APIs.** 2018**.** Disponível em: https://ieeexplore.ieee.org/abstract/document/8385157. Acesso 20 nov. 2022.

BROOKE, John. **SUS: A Retrospective**. 2013. Disponível em: https://uxpajournal.org/sus-a-retrospective/ . Acesso 20 nov. 2022.

State of Agile Report**. State of Agile**. 2022. Disponível em: https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf. Acesso em: 22 dez. 2022.

GLITCH. **Trello**. 2011. Disponível em: https://trello.com/. Acesso em: 22 jul. 2022.

ATLASSIAN. **Jira**. 2004. Disponível em: https://www.atlassian.com/br/software/jira. Acesso em: 22 jul. 2022.

INC., Notion Labs. **Notion**. 2016. Disponível em: https://www.notion.so/. Acesso em: 22 jul. 2022.

Mango Technologies, Inc. **ClickUp.** 2017. Disponível em: https://clickup.com/. Acesso em: 15 nov. 2022.

monday.com Ltd**. Monday.com.** 2012. Disponível em: https://www.monday.com. Acesso em: 15 nov. 2022.

Shore Labs. **Kanban Tool.** 2012. Disponível em: https://kanbantool.com. Acesso em: 15 nov. 2022.

Asana, Inc. **Asana.** 2008. Disponível em: https://kanbantool.com. Acesso em: 15 nov. 2022.

WALKE, Jordan. **React**. 2013. Disponível em: https://reactjs.org/. Acesso em: 22 jul. 2022.

Vercel Inc. **Styled Components**. 2015. Disponível em: https://reactjs.org/. Acesso em: 20 dez. 2022.

ABRAMOV, Dan. **Redux**. 2015. Disponível em: https://styled-components.com/. Acesso em: 22 jul. 2022.

DAHL, Ryan. **NodeJs**. 2009. Disponível em: https://nodejs.org/en/. Acesso em: 22 jul. 2022.

STONEBRAKER, Michael. **PostgresSQL**. 1986. Disponível em: https://www.postgresql.org/. Acesso em: 22 jul. 2022.

KONG, Inc. **insomnia.** 2017**.** Disponível em: https://insomnia.rest/. Acesso em: 22 jul. 2022.

Anexos

Anexo 1

```
26  const handleSignIn = async () => {
27      try {
28          setLoading(true);
29          const res = await api.post('/user/login', {
30              email: loginPaylod.email,
31              password: loginPaylod.password
32          })
33          dispatch(setCalendarDate(res?.data));
34          localStorage.setItem('@StudyNizer:userSession', JSON.stringify(res?.data));
35          setLoading(false);
```

Anexo 2

```
src > routes > AuthLayout > index.jsx > AuthLayout
      You, last month | 1 author (You)
1   import { Navigate, Outlet } from 'react-router-dom';
2
3   export const AuthLayout = ({ signed }) => {
4
5       if (!signed) {
6           return <Navigate to={"/sign-in"} replace />
7       }
8
9       if (signed) {
10          return <Outlet />
11      }
12  };
```

Anexo 3

```
46
47      const [state, setState] = useState({
48        "Tarefas": {
49          title: "Tarefas",
50          items: [],
51          columnType: 'todo'
52        },
53        "Fazendo": {
54          title: "Fazendo",
55          items: [],
56          columnType: 'doing'
57        },
58        "Concluído": {
59          title: "Concluído",
60          items: [],
61          columnType: 'completed'
62        }
63    });
```

Anexo 4

```
157   const handleDragEnd = async ({destination, source}) => {      Marcelo T. Cortes, 4 weeks ago •
158     if (!destination) {
159       return
160     }
161
162     if (destination.index === source.index && destination.droppableId === source.droppableId) {
163       return
164     }
165
166     const itemCopy = {...state[source.droppableId].items[source.index]}
167
168     setState(prev => {
169       prev = {...prev}
170       prev[source.droppableId].items.splice(source.index, 1)
171
172       prev[destination.droppableId].items.splice(destination.index, 0, itemCopy)
173
174       return prev
175     })
```

Anexo 5

```
try {
  await Axios.all[
    await api.post(`/user/board-tasks-${columnType}`, {
      id: cardId,
      users_id: userId,
      title: text,
      description: description,
      priority: priority,
      due_date_start: taskDueDate[0],
      due_date_end: taskDueDate[1]
    }, {headers}),
```

Anexo 6

```
const handleExportSubject = async (subject) => {      You, 3 weeks ago • import and export subjects and markdowns
  try {
    const res = await api.get(`/user/markdown/${userId}/${subject.title.replace(/ /g, '-').toLowerCase()}`, {headers});
    var blob = new Blob([JSON.stringify(res.data)], {type: "text/plain;charset=utf-8"});
    FileSaver.saveAs(blob, `${subject.title}.txt`);
  } catch (error) {
    notification.info({
      message: `${error?.response?.data?.error}`,
      placement: 'top',
    });
  }
}
```

Anexo 7

```
const handleImportSubject = async (e) => {
  e.stopPropagation();
  if(subjects.find(subject =>  subject.title === fileNameSanitized)) {
    notification.info({
      message: `Disciplina já importada!`,
      placement: 'top',
    });
    setFileName('');
  } else {
    try {
      await api.post(`/user/markdownImport`, {
       users_id: userId,
       markdown_data: fileContent
      }, {headers});
```

Anexo 8

```
const renderEditor = () => {
    if(pageName !== '' && pageArray.length !== 0) {
        return (<div onClick={(e) => handleShowMarkupPanel(e)}>
            <Editor
                ref={editor}
                editorState={editorState}
                onChange={(editorState) => handleChangeEditor(editorState)}
                blockStyleFn={getBlockStyle}
            />
        </div>)
    }

    return (<BlankAnnotationContainer>
        <FaRegFileAlt />
        <p>Clique ou adicione uma página no menu à esquerda</p>
    </BlankAnnotationContainer>)
}
```

Anexo 9

```
62      const handleChangeEditor = (editorState) => {
63          let contentRaw = convertToRaw(editorState.getCurrentContent());
64
65          const filteredResult = pageArray.find((obj) => obj.url_id === location.pathname);
66
67          if (filteredResult){
68              filteredResult.annotation_block.annotationBlock = contentRaw;
69          }
70
71          setEditorState(editorState);
72          debouncedSave(contentRaw, pageName, pageId, markdownId, urlId);
73
74          setMarkdownPanelVisible('none');
75      }
76
```

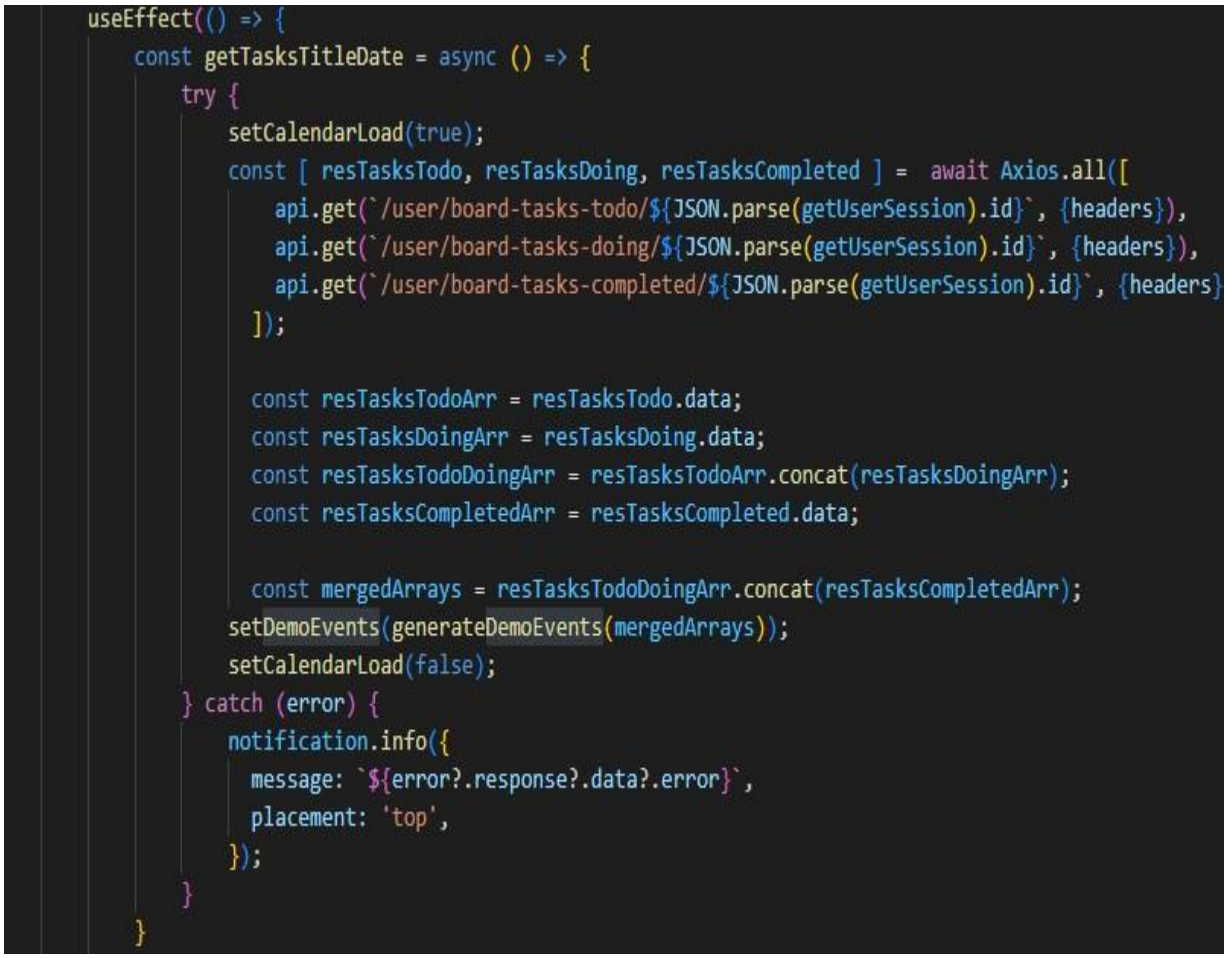
Anexo 10

```
2
3   const debouncedSave = useRef(debounce(async (annotationBlockValue, pageName, pageId, markdownId, urlId) => {
4       try {
5           setIsSaving(true);
6           await api.put(`/user/markdown/${markdownId}`, {
7               annotation_block: { annotationBlock: annotationBlockValue },
8               page_name: pageName,
9               url_id: urlId,
0               page_id: pageId,
1               subject_name: location.state.subject.title.replace(/ /g, '-').toLowerCase(),
2           }, {headers});
3           setIsSaving(false);
4       } catch (error) {
5           console.log('error?.response?.data?.error', error?.response?.data?.error);
6           setIsSaving(false);
7       }
8   }, 1000)).current;
9
```

Anexo 11

```
            <Kalend
                kalendRef={props.kalendRef}
                onNewEventClick={onNewEventClick}
                initialView={CalendarView.AGENDA}
                disabledViews={[CalendarView.WEEK, CalendarView.THREE_DAYS, CalendarView.DAY]}
                onEventClick={onEventClick}
                events={demoEvents}
                initialDate={new Date().toISOString()}
                hourHeight={60}
```

Anexo 12



```
useEffect(() => {
    const getTasksTitleDate = async () => {
        try {
            setCalendarLoad(true);
            const [ resTasksTodo, resTasksDoing, resTasksCompleted ] =  await Axios.all([
                api.get(`/user/board-tasks-todo/${JSON.parse(getUserSession).id}`, {headers}),
                api.get(`/user/board-tasks-doing/${JSON.parse(getUserSession).id}`, {headers}),
                api.get(`/user/board-tasks-completed/${JSON.parse(getUserSession).id}`, {headers}
              ]);

              const resTasksTodoArr = resTasksTodo.data;
              const resTasksDoingArr = resTasksDoing.data;
              const resTasksTodoDoingArr = resTasksTodoArr.concat(resTasksDoingArr);
              const resTasksCompletedArr = resTasksCompleted.data;

              const mergedArrays = resTasksTodoDoingArr.concat(resTasksCompletedArr);
            setDemoEvents(generateDemoEvents(mergedArrays));
            setCalendarLoad(false);
        } catch (error) {
            notification.info({
              message: `${error?.response?.data?.error}`,
              placement: 'top',
            });
        }
    }
```

APÊNDICE A – ARTIGO STUDYNIZER - PLATAFORMA WEB E APLICATIVO PARA AJUDAR ALUNOS A ORGANIZAREM SEUS ESTUDOS

STUDYNIZER - PLATAFORMA WEB E APLICATIVO PARA AJUDAR ALUNOS A ORGANIZAREM SEUS ESTUDOS

José E. De Lucca, Marcelo Teixeira Cortes

Departamento de Informática e Estatística Universidade Federal de Santa Catarina (UFSC) –Florianópolis, SC – Brazil
jose.lucca@ufsc.br, marcelo.tc@grad.ufsc.br

ABSTRACT

During the university journey, many students may encounter difficulties in their daily activities. With the accumulation of tasks, tests and assignments, it is extremely important to maintain an organizational routine, so that way, the student can obtain a good academic performance. In this final paper, the development of a web platform and application is proposed, which aims to help students to organize their tasks, from different disciplines, through concepts that are related to the use of agile methodologies. The application has a panel in which it is possible to record and follow the progress of each assigned task, along with the registration of summaries, schemes or other strategies, that aim to help the study of the subjects present in the disciplines and in the execution of the tasks. It will be possible to integrate events with the calendar system and share summaries with other students.

Palavras-chave: *Organization, Agile methodologies, Tasks, Students*

RESUMO

Durante a trajetória universitária muitos estudantes podem encontrar dificuldades em suas atividades cotidianas. Com o acúmulo de tarefas, provas e trabalhos é de exímia importância manter uma rotina de organização, para que assim, seja possível obter um bom desempenho acadêmico. No presente trabalho é proposto o desenvolvimento de uma plataforma web e aplicativo, esta visa auxiliar estudantes universitários a organizarem suas tarefas, de diferentes disciplinas, por meio de conceitos que estão relacionados a utilização de metodologias ágeis. A aplicação possui um painel em que é possível registrar e acompanhar o progresso de cada tarefa atribuída, juntamente com o registro de resumos, esquemas ou outras estratégias, que visam auxiliar o estudo dos conteúdos presentes nas disciplinas e na execução das tarefas. Será possível integrar eventos com o sistema de calendários e compartilhar resumos com outros estudantes.

Palavras-chave: *Organização, Metodologias ágeis, Tarefas, Estudantes*

1 INTRODUÇÃO

A importância da organização é primordial para o sucesso de qualquer projeto, como por exemplo o desenvolvimento de software onde qualquer parte não documentada ou mal escrita pode levar ao fracasso de uma equipe. Diversas são as técnicas que auxiliam na entrega desses projetos, como os frameworks ágeis: SCRUM, EXTREME PROGRAMMING e KANBAN que são hoje os principais no mercado onde diversas empresas ao redor do mundo utilizam para obter uma maior coordenação de ideias e assim elaborar softwares de qualidade, a agilidade envolve tanto a capacidade de adaptação a diferentes mudanças e para refinar e ajustar processos conforme necessário (State of Agile Report., 2022).

O mesmo acontece com a vida estudantil, principalmente a do universitário onde este possui diversas cadeiras para cumprir e muitas das vezes também trabalha para se sustentar, por isso é necessário cada vez mais obter formas e técnicas de organização e planejamento de estudos para que alunos consigam performar de maneira a obter sucesso na sua caminhada. Diversas são as causas para um mau desempenho no meio acadêmico, como aspectos de relações interpessoais (problemas familiares ou com amigos, professores), problemas de saúde ou aspectos como o enfrentamento diante a preconceitos que grupos minoritários podem sofrer, essas dificuldades externas acrescentadas com a falta de uma organização, podem multiplicar os problemas ao longo da caminhada do estudante (BASSO *et al.*, 2022).

As metodologias ágeis são abordagens para o gerenciamento de projetos que enfatizam a flexibilidade, a adaptabilidade e a entrega contínua de valor. Elas foram desenvolvidas como uma alternativa aos processos de gerenciamento de projetos mais tradicionais, que são mais rígidos e menos adaptáveis às mudanças. As metodologias ágeis se baseiam nos valores e princípios descritos no Manifesto Ágil, que incluem a entrega contínua de valor, a colaboração entre os membros do time, a adaptação a mudanças e a simplicidade, são diversas as opções disponíveis, o presente trabalho irá focar no KANBAN como uma de suas funcionalidades dentro do sistema, apesar destas metodologias serem focadas em equipes, o KANBAN em particular possui uma característica muito simplificada e intuitiva que pode ser usada por qualquer indivíduo para suas tarefas diárias e por isso a importância e o enfoque desta metodologia para este trabalho. A visualização do que deve ser feito e quando, juntamente com o sentimento de progresso são fatores importantes para os estudantes que procuram se organizar (ALSAQQA, 2020).

Hoje existem muitos aplicativos que ajudam os estudantes a se organizarem melhor, como o Evernote para anotações de tarefas, o Google Calendar para calendário e o Todoist para gerenciamento de tarefas. No entanto, muitos desses aplicativos se concentram em uma área específica e não oferecem uma combinação de todas essas funcionalidades. Além disso, a usabilidade nem sempre é muito eficiente, o que pode afastar ou até mesmo estressar os usuários, não existindo uma junção de todas essas funcionalidades, além disso a usabilidade não costuma ser muito eficiente, o que pode afastar e

até mesmo estressar o usuário da aplicação. Este projeto visa unificar funcionalidades destes softwares existentes hoje no mercado para que assim o usuário tenha disponível a maioria destas utilidades que ajudam na organização, não precisando ter que fazer *download* de diversas aplicações tendo assim um único ponto de referência para organizar sua vida estudantil.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é criar um aplicativo mobile e plataforma web que possa ser utilizado por universitários na organização e planejamento de seus estudos durante o semestre. Serão utilizadas técnicas do desenvolvimento ágil trazidas para a realidade de uma vida acadêmica. Poderá ser feito o cadastro de disciplinas, tendo assim um painel para acompanhar o progresso de cada tarefa atribuída, juntamente com ferramentas de registro de resumos, esquemas ou outras estratégias que os auxiliem no estudo dos conteúdos e na execução das tarefas. Haverá integração com dispositivos móveis e com sistema de calendário e também um módulo para facilitar o trabalho em grupo dos alunos.

1.1.2 Objetivos Específicos

4. Criar um aplicativo mobile para dispositivos Android e que seja acessível em termos de usabilidade para os alunos universitários.

5. Adotar estratégias de frameworks ágeis para a realidade de um estudante universitário.

6. Realizar testes manuais para obter métricas de desempenho, possibilitando assim gerar melhorias no futuro.

1.2 Delimitações

Este trabalho possui as delimitações a seguir:

- A aplicação deve possuir uma versão web e mobile.
- A versão mobile deverá estar disponível apenas para Android.

1.3 Resultados Esperados

- Aplicação web e mobile para gerenciamento de tarefas.
- Testes unitários e end-to-end da aplicação.
- Design responsivo, seguindo boas práticas de *User Experience* (UX) e *User interface* (UI).
- Documentação.

1.3 MÉTODO DE PESQUISA

Para guiar o trabalho e identificar quais funcionalidades da plataforma serão mais úteis e relevantes, será realizada uma coleta de dados e um estudo detalhado sobre as maiores dificuldades enfrentadas pelos universitários nos estudos e organização. Isso incluirá a coleta de dados, como questionários e entrevistas, para obter uma visão completa das necessidades e expectativas dos usuários. Com essas informações, será possível definir as funcionalidades mais importantes e relevantes para a plataforma, de modo a garantir a sua eficácia e usabilidade.

A análise de tecnologias das plataformas já existentes também será levada em consideração para uma viabilidade técnica e assim garantir um melhor resultado no desenvolvimento do projeto. A partir disso será guiado o trabalho para quais funcionalidades na plataforma serão mais úteis e quais não valem a pena serem exploradas.

Uma análise de UX e UI também será explorada, pois a importância de uma boa interface principalmente em um app de organização pessoal é primordial. Após o desenvolvimento do projeto será feito um teste com usuários reais e estudantes para relatarem a sua experiência utilizando a plataforma web e mobile, para que assim possa se ter uma ideia do que pode ser melhorado e quais funcionalidades foram as mais bem avaliadas.

1.4 ORGANIZAÇÃO

No capítulo 2, são apresentados alguns conceitos e ferramentas importantes que serão utilizados ao longo do trabalho assim como trabalhos similares que serão usados como base para a implementação.

No capítulo 3, são discutidos alguns trabalhos similares que servirão como inspiração principalmente no que tange a organização e metodologias ágeis.

No capítulo 4, há uma seção discutindo o desenvolvimento do projeto e aspectos arquiteturais do software, assim como tecnologias que serão utilizadas, requisitos, histórias de usuários e também mockups de algumas telas a serem desenvolvidas.

No capítulo 5, é apresentado o teste de usabilidade SUS para avaliar o protótipo desenvolvido neste trabalho.

No capítulo 6, é apresentada a conclusão do presente trabalho juntamente com possíveis funcionalidades a serem implementadas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados temas e tecnologias que serão usados para o desenvolvimento do trabalho e assim explicar do por que a utilização e escolha dos mesmos na aplicação. Também será apresentado um quadro comparativo das tecnologias e metodologias ágeis para um melhor entendimento de suas características.

2.1 Conceitos

2.1.1 Organização

A organização é uma palavra originada do grego *"organon"* que significa instrumento, utensílio, órgão ou aquilo com que se trabalha e possui diversos significados dependendo do contexto, mas de forma geral o significado seria a forma como um sistema se dispõe para atingir os resultados pretendidos (CEARÁ, 2015). Para o escopo deste trabalho será usado essa palavra principalmente no sentido de "ordem" para as tarefas a serem executadas pelos alunos.

Existem diversas metodologias ágeis que ajudam indivíduos a se organizarem de forma inteligente e metódica, as quais são utilizadas principalmente em ambiente corporativos, o que há de novo nas metodologias ágeis não são as práticas usadas, mas o reconhecimento das pessoas como principal impulsionador do sucesso do projeto, juntamente com um foco intenso em eficácia e manobrabilidade (COCKBURN, 2022).

2.1.2 Tarefa

A definição de tarefa descreve uma atividade de trabalho que por sua vez é desempenhada por funções específicas. A granularidade de uma tarefa é geralmente algumas horas ou poucos dias e afeta um número pequeno de pessoas. As tarefas são utilizadas como uma base para o processo de planejamento (PERNAMBUCO, 2005).

As etapas de uma tarefa podem ser definidas como:

- **Etapas de Estudo:** em que o indivíduo que desempenha a função entende a natureza da tarefa, reúne e examina os Produtos de Trabalho de entrada e formula o resultado.
- **Etapas de Execução:** em que o indivíduo que desempenha a Função cria ou atualiza alguns Produtos de Trabalho.
- **Etapas de Revisão:** em que o indivíduo que desempenha a Função inspeciona os resultados em relação a alguns critérios.

2.1.3 Métodos de Estudos

Basso (2013) apresenta um estudo sobre a experiência universitária quanto ao rendimento dos estudantes e suas vivências acadêmicas através das Oficinas de Organização e Métodos de Estudo da Universidade Federal de Santa Catarina (UFSC). Estas oficinas fazem parte do projeto REUNI do programa de Pós-Graduação em Psicologia. Segundo as autoras, em uma primeira etapa da pesquisa foram levantados os índices de reprovação por FI (frequência insuficiente) e de evasão. No âmbito da UFSC, FI é caracterizado quando os alunos não têm 75% de presenças nas atividades da disciplina.

Foi relatado diversos motivos para a causa do mal desempenho dos estudantes, um dos fatores seria a um desajustamento psicossocial no processo ensino-aprendizagem e a incompatibilidade vocacional. Também vale muito a pena citar problemas socioeconômicos dos estudantes que precisam trabalhar e estudar ao mesmo tempo e também a falta de uma base escolar principalmente em cálculos.

No artigo cita uma triagem feita entre estudantes da universidades com os cursos com maior evasão e assim recolhido dados dos principais motivos para a busca pelos estudantes às oficinas Organização e Métodos de estudo:

Tabela 1 - Motivos para a busca dos estudantes às oficinas Organização e Métodos de estudo

Motivos	Nº	%
Dificuldades para organizar o tempo	10	27
Necessidade de melhor organizar os estudos	08	22
Problemas de focalização da atenção	04	11
Necessidade de aprender a se disciplinar	03	8
Baixo desempenho acadêmico	03	8
Maximizar o seu rendimento	03	8
Buscar equilíbrio entre a vida acadêmica e a vida pessoal	02	5

Fonte: Elaboração do autor, 2022

Sendo assim percebe-se que grande parte dos alunos possuem problemas quanta a falta de organização segundo Maccann (2010) “Os alunos altamente conscienciosos tendem a usar mais estratégias de gerenciamento de tempo, principalmente aquelas relacionadas a cumprimento de prazos,

organização e planejamento", O mesmo autor relata que a conscienciosidade se difere de saber gerenciar o tempo pois, no último a pessoa precisa de uma forma empírica para praticar e treinar esse hábito ao longo do tempo, já a conscienciosidade está ligado a fatores genéticos muitas vezes.

Por isso a questão da organização é tão primordial para o sucesso na vida acadêmica já que sem ela pode-se acarretar em uma baixa na produtividade e assim como no desenvolvimento de software e projetos, produtividade é a palavra chave para o sucesso.Uma das ferramentas mais importantes e conhecidas de organização de trabalho e tarefas é sem sombra de dúvidas o Kanban e segundo Wakode (2015) seus princípios são:

- Visualização do trabalho
- Limitar o trabalho em andamento
- Foco no fluxo
- Melhoria Contínua

Com este tipo de metodologia ágil pode-se trazer o controle de tarefas e afazeres de um estudante para um board e assim também utilizá-lo para seu benefício no dia a dia.

2.1.4 Metodologias ágeis

As metodologias ágeis são um amplo guarda-chuva de crenças de desenvolvimento de software. É uma estrutura conceitual para engenharia de software que começa com uma fase de planejamento inicial e segue o caminho para a fase de implantação com interações iterativas e incrementais durante todo o ciclo de vida do projeto. O objetivo inicial dos métodos ágeis é reduzir a sobrecarga no processo de desenvolvimento de software com a capacidade de adotar mudanças sem arriscar o processo ou sem retrabalho excessivo (ALSAQQA, 2020).

O manifesto ágil possui uma série de valores e princípios que são a base para o desenvolvimento dos processos que os guiam. A começar pelos valores que são:

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

Além disso o manifesto possui 12 princípios complementares que fortificam a mentalidade ágil:

- Princípio 1: satisfazer o cliente através da entrega contínua
- Princípio 2: Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento
- Princípio 3: Entregar frequentemente software funcionando
- Princípio 4: Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto

- Princípio 5: Construir projetos em torno de indivíduos motivados
- Princípio 6: O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face
- Princípio 7: Software funcionando é a medida primária de progresso
- Princípio 8: Os processos ágeis promovem desenvolvimento sustentável
- Princípio 9: Contínua atenção à excelência técnica e bom design aumenta a agilidade
- Princípio 10: Simplicidade – A arte de maximizar a quantidade de trabalho não realizado – é essencial
- Princípio 11: As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis
- Princípio 12: Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

A facilidade de visualização das tarefas e sensação de progresso é um fator importante, a aplicabilidade destas metodologias é muito útil para aqueles que buscam uma organização em seu ambiente de trabalho e se encaixa perfeitamente em um ambiente acadêmico.

Algumas das metodologias ágeis mais utilizadas hoje no mercado são:

Kanban: O Kanban é uma técnica de gerenciamento de projetos que foi desenvolvida no Japão no final da década de 1950. O termo "kanban" é uma palavra japonesa que significa "cartão visual". A técnica foi criada como uma forma de melhorar a eficiência e a produtividade na produção de peças de automóveis pela Toyota. A ideia por trás do Kanban é simples: criar um sistema visual que permita aos funcionários ver o que precisa ser feito e quando precisa ser feito. Isso é feito através de cartões ou tabelas que são colocados em uma parede ou painel de forma que todos possam ver. Cada cartão representa um item ou tarefa específica que precisa ser concluída.

Foi originalmente desenvolvido como uma técnica para a produção em massa, mas hoje em dia é amplamente utilizado em uma variedade de indústrias e ambientes de trabalho. É especialmente útil para projetos que envolvem muitas tarefas pequenas e independentes, como o desenvolvimento de software. Para usar o Kanban, os funcionários criam uma lista de tarefas e as dividem em três categorias: prontas para serem iniciadas, em

andamento e concluídas. Cada tarefa é então representada por um cartão ou item em uma tabela é movida através dessas três categorias conforme avança. Isso permite que todos vejam o progresso do projeto e saibam o que precisa ser feito a seguir.

Inclui o conceito de "limite de trabalho em andamento" (WIP, na sigla em inglês), que se refere à quantidade máxima de tarefas que podem estar em andamento ao mesmo tempo. Isso ajuda a evitar o sobrecarregamento de trabalho e a garantir que todas as tarefas sejam concluídas de maneira otimizada. É uma técnica de gerenciamento de projetos altamente visual e intuitiva que tem sido amplamente adotada em todo o mundo. Além disso, o Kanban é altamente adaptável e pode ser facilmente ajustado para atender às necessidades de uma ampla variedade de projetos e ambientes de trabalho. (CORONA, 2013).

Scrum: O Scrum surgiu nos anos 1980 como um framework para gerenciamento de projetos de software. Foi desenvolvido por Hirotaka Takeuchi e Ikujiro Nonaka, que estudavam como os times de desenvolvimento de software poderiam se adaptar rapidamente a mudanças e entregar valor contínuo aos clientes. Takeuchi e Nonaka observaram que os times de desenvolvimento de software tradicionais, que seguiam metodologias de linha de montagem, tendiam a ser inflexíveis e lentos para se adaptar a mudanças.

Eles propuseram uma nova abordagem, baseada em iterações curtas e entrega contínua de valor, que se tornaria conhecida como Scrum. Se baseia no modelo de desenvolvimento ágil, que enfatiza a adaptabilidade, a entrega contínua de valor e a colaboração com o cliente. Divide-se em três papéis principais: o Scrum Master, o Product Owner e o time Scrum. O Scrum Master é responsável por garantir que o time Scrum esteja seguindo as regras e práticas do Scrum, enquanto o Product Owner é responsável por definir o que o time Scrum deve fazer e o time Scrum é responsável por realizar o trabalho.

O ciclo de trabalho do Scrum é dividido em iterações curtas chamadas "sprints". Em cada sprint, o time Scrum se reúne para planejar o trabalho a ser realizado, executar o trabalho e revisar o progresso. Isso permite que o time se adapte rapidamente a mudanças no projeto e forneça valor contínuo para o cliente. É amplamente utilizada em projetos de software e é conhecida por sua flexibilidade e eficiência, especialmente útil em projetos de grande escala ou de alto nível de incerteza, onde a adaptabilidade é crucial. (PEREIRA, 2007).

Extreme Programming (XP): Foi desenvolvida por Kent Beck no final da década de 1990. Beck trabalhava como programador e líder de projeto em diversos projetos de software e percebeu que

as metodologias tradicionais de gerenciamento de projetos tendiam a ser inflexíveis e lentas para se adaptar a mudanças. Inspirado pelo modelo de desenvolvimento ágil, que enfatiza a adaptabilidade, a entrega contínua de valor e a colaboração com o cliente, Beck desenvolveu o Extreme Programming como uma forma de tornar os projetos de software mais eficientes e adaptáveis.

Beck publicou o livro "Extreme Programming Explained: Embrace Change" em 1999, que descreveu os princípios e práticas do XP como são conhecidos hoje. A metodologia XP se divide em duas fases principais: o planejamento e a refatoração. Durante a fase de planejamento, o time de desenvolvimento se reúne com o cliente para definir os objetivos e as funcionalidades do projeto. Na fase de refactoring, o time implementa e testa as funcionalidades de acordo com os objetivos definidos. É caracterizada por seus 12 princípios:

- O cliente deve sempre estar presente.
- O projeto deve ser desenvolvido em iterações curtas.
- O projeto deve ser planejado com precisão.
- Mudanças são esperadas, bem-vindas e gerenciadas.
- O projeto é desenvolvido em pares.
- O projeto é baseado em testes.
- O projeto é baseado em comunicação clara e concisa.
- A simplicidade é valorizada.
- O projeto é gerenciado com transparência.
- O projeto é gerenciado com feedback constante.
- O projeto é gerenciado com a qualidade como principal objetivo.
- O projeto é gerenciado com a colaboração como principal objetivo.

Baseia-se em práticas ágeis, como a programação em pares, o teste de unidade e o teste integrado. A metodologia XP é amplamente utilizada em projetos de software e é conhecida por sua eficiência e flexibilidade. Ela é especialmente útil em projetos de pequena escala ou de alto nível de incerteza, onde a adaptabilidade é crucial. (TELES, 2005).

3 SOLUÇÕES SIMILARES

O tema organização pessoal e empresarial é recorrente, por isso diversos são os sistemas e aplicativos que tentam suprir essa necessidade por ordem nas tarefas diárias dos indivíduos.

A seguir são mostradas algumas destas soluções tanto no âmbito de uma empresa quanto para o uso pessoal. Cada um destes sistemas possuem suas

funcionalidades particulares porém sempre com o mesmo intuito. Alguns dos critérios mais importantes a serem considerados incluem:

- Eficiência: é importante escolher uma solução que seja eficiente e otimize o tempo e os esforços do usuário. Isso inclui considerar se a solução é fácil de usar e se possui recursos que facilitam a realização de tarefas.
- Customização: é importante avaliar se a solução pode ser personalizada de acordo com as necessidades e objetivos específicos do usuário. Isso inclui considerar se é possível adicionar ou remover recursos, bem como se é possível ajustar as configurações e as preferências do usuário.
- Recursos: é fundamental avaliar os recursos disponíveis na solução e verificar se eles são adequados para atender às necessidades do usuário. Isso inclui considerar se a solução possui recursos avançados, como integrações com outras ferramentas ou recursos de automação.
- Usabilidade: é importante avaliar a usabilidade da solução, ou seja, sua facilidade de uso. Isso inclui considerar se a interface é intuitiva e fácil de entender, bem como se a solução possui tutoriais ou recursos de ajuda disponíveis.

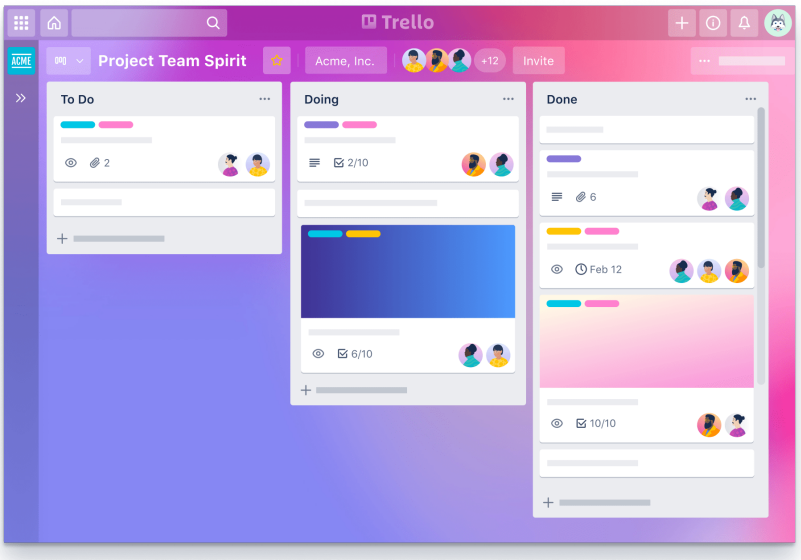
As aplicações a seguir foram encontradas através de pesquisas em artigos especializados no tema de organização de tarefas e assim escolhidos baseados nas funcionalidades que fazem sentido para um estudante.

3.1 Trello

Ferramenta visual que capacita equipes a gerenciar qualquer tipo de projeto, fluxo de trabalho ou rastreamento de tarefas. Permite adicionar arquivos, listas de verificação ou até mesmo automação.

Em sua essência, o Trello conta com os princípios dos quadros de projetos Kanban para visualizar fluxos de trabalho, fornecendo aos gerentes e membros da equipe uma visão geral simples de um projeto do início ao fim.

Figura 1 - Board do trello



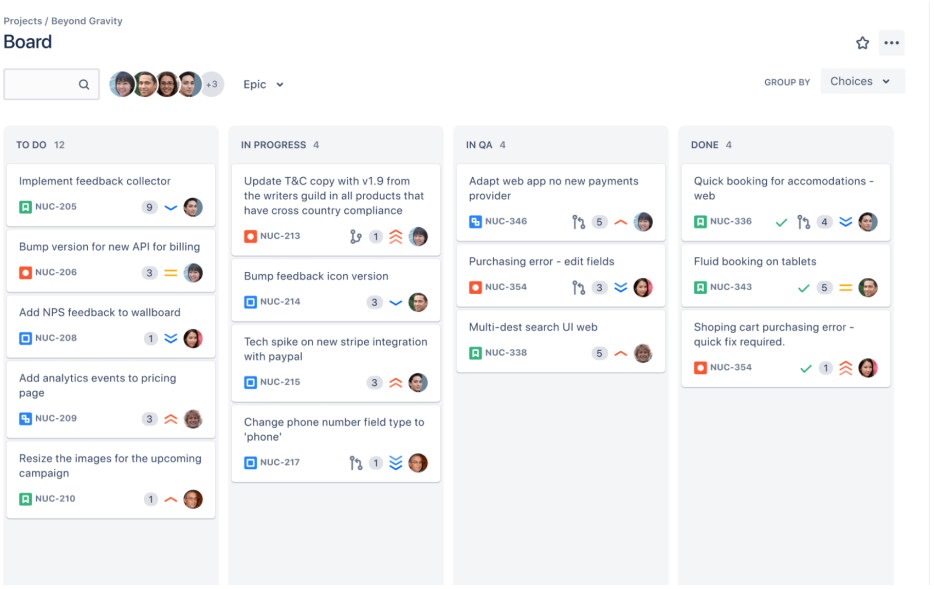
Fonte:https://tecnoblog.net/responde/trello-nao-permite-mais-criar-um-board-fora-de-um-time/

Outra característica é sua facilidade de uso e versatilidade: o aplicativo geralmente também é usado para uso pessoal, incluindo planejamento de férias a casamentos. Nesse sentido, difere do software de gerenciamento de projetos completo, priorizando uma funcionalidade leve e acessibilidade em um amplo conjunto de recursos (GLITCH, 2011).

3.2 Jira

O Jira Software faz parte de uma família de produtos desenvolvidos para ajudar equipes de todos os tipos a gerenciar o trabalho. Originalmente, o Jira foi projetado como um rastreador de bugs e problemas. Mas hoje, o Jira evoluiu para uma poderosa ferramenta de gerenciamento de trabalho para todos os tipos de casos de uso, desde requisitos e gerenciamento de casos de teste até desenvolvimento ágil de software.

Figura 2 - Board do Jira



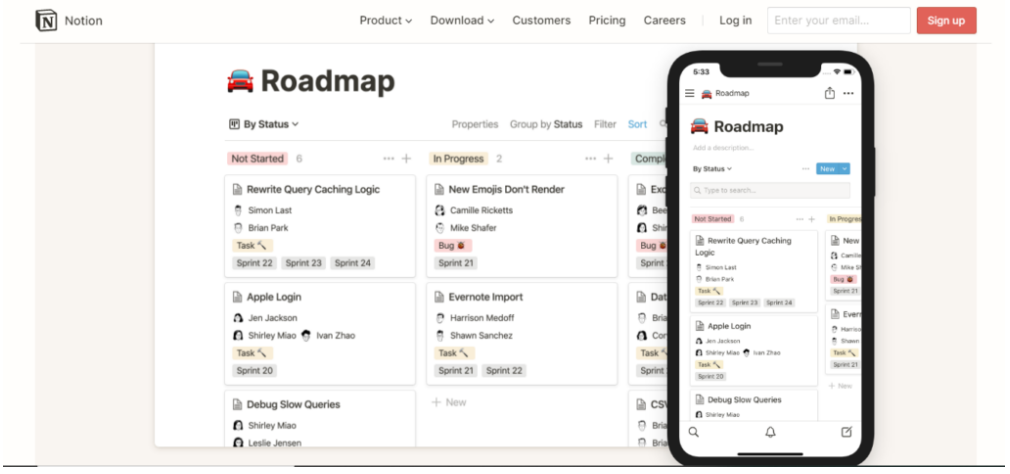
Fonte: https://www.atlassian.com/br/software/jira/features

A plataforma aproveita todos os tipos de habilidades de gerenciamento de projetos, incluindo desenvolvimento de software, gerenciamento ágil de projetos, rastreamento de bugs, gerenciamento de scrum, gerenciamento de conteúdo, marketing, gerenciamento de serviços profissionais e muito mais (ATLASSIAN, 2004).

3.3 Notion

Notion é uma ferramenta de produtividade e uma ferramenta de gerenciamento de projetos para ajudar na organização. Possui muitas ferramentas de produtividade, como Tarefas, Notas, Cadernos e Lembretes. Também possui Tabelas, Bancos de Dados. Ajuda pequenas e grandes organizações a coordenar prazos, objetivos e atribuições.

Figura 3 - Página do Notion



Fonte: https://www.apptuts.net

Permite construir um wiki pessoal com infinitas camadas de conteúdo, planejar usando uma visualização kanban, um calendário ou uma visualização de lista simples.

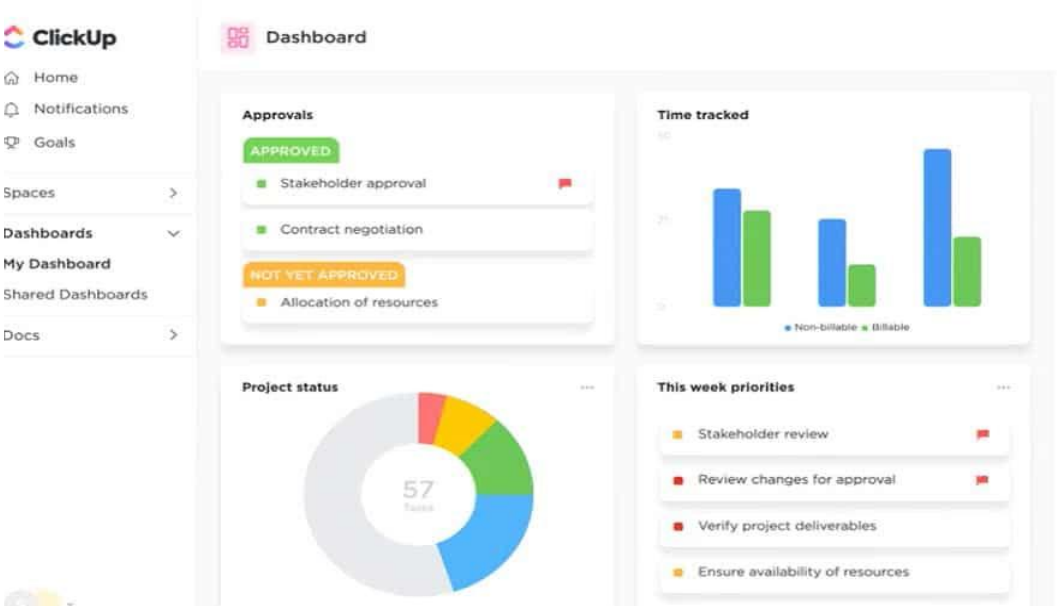
Fornece funcionalidade de equipe para colaboração em tempo real e permite que as equipes compartilhem, comentem e atribuem tarefas e lembretes. Assim como indivíduos e profissionais podem usar o Notion, as equipes também podem (NOTION, 2016).

3.4 Click Up

A clickUp é uma plataforma de colaboração e gerenciamento de projetos para times de pequena ou grande escala, possui diversas funcionalidades, entre elas ferramentas de comunicação, atribuições e status de tarefas, alertas, chat. Permite que os usuários documentem bugs ou anote atas de reuniões com o ClickUp Docs e editem em tempo real com outras pessoas, além disso Possui integração com diversas outras ferramentas como: Github, Slack, GitLab, GoogleDrive, Outlook, Figman, Google Calendar.

Os projetos podem ser visualizados em um painel Agile ou organizados pelo responsável. O fluxo de atividades exibe as tarefas à medida que são criadas e concluídas em tempo real.

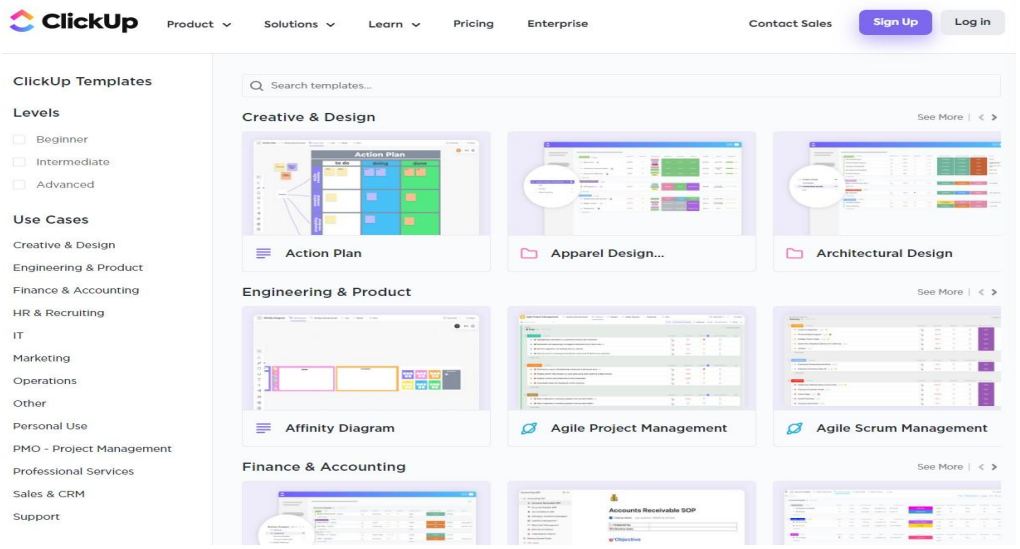
Figura 4 - Dashboard do ClickUp



Fonte: https://clickup.com/

Há também a possibilidade de importar templates através do *template center*, uma biblioteca de modelos prontos para uso. Você pode acessar o Template Center de várias maneiras para localizar modelos para localizar um modelo criado por membros do seu espaço de trabalho ou pela comunidade ClickUp. Os modelos podem ser mantidos privados ou compartilhados com pessoas específicas ou equipes (CLICKUP, 2017)

Figura 5 - Página de templates da ClickUp



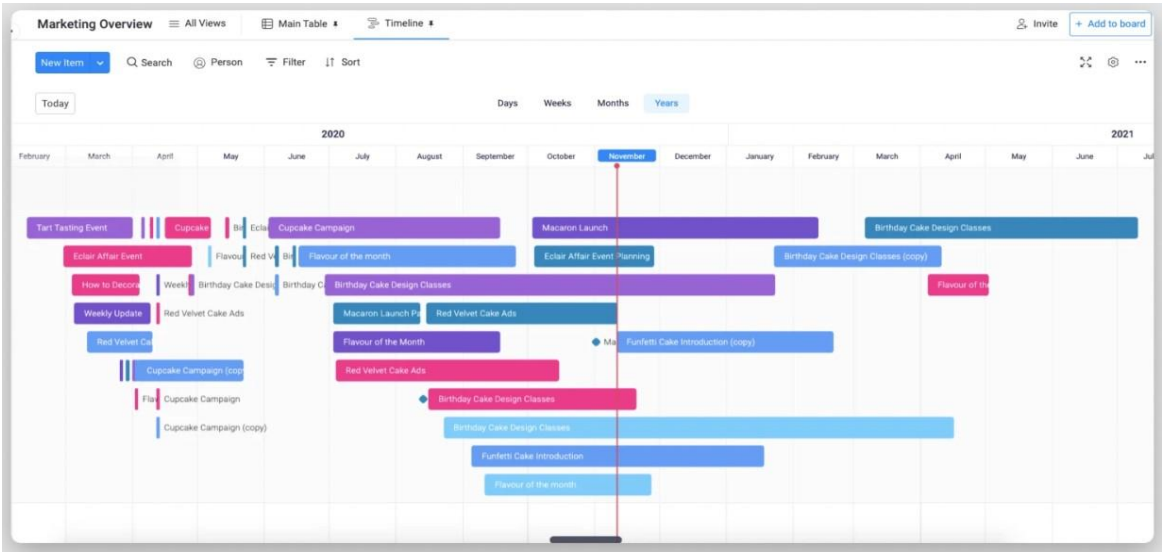
Fonte: https://clickup.com/

3.5 Monday.com

É uma ferramenta de gerenciamento de projetos, oferece uma rica visualização dos projetos com base nas preferências do usuário, incluindo Kanban, Gantt, linha do tempo e exibições de calendário. Se a equipe quiser ter uma visão geral ou quer ver as tarefas diárias, pode-se facilmente alternar as exibições, detalhar ou diminuir o zoom o quanto precisar, além de criar exibições personalizadas para reunir todos os dados e informações em um só lugar.

Similar ao ClickUp há também como adicionar templates para diferentes categorias de negócio, como design, desenvolvimento de software, RH, marketing, vendas entre outras.

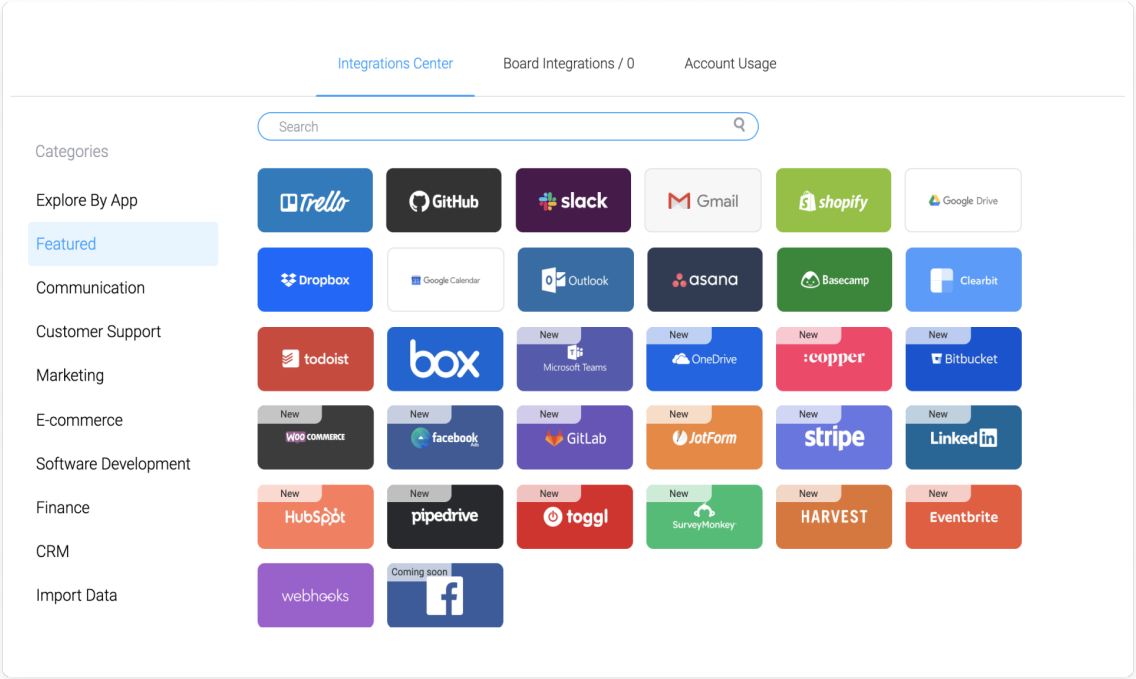
Figura 6 - Timeline da Monday.com



Fonte: https://monday.com

Outra funcionalidade é a integração com as ferramentas e aplicativos, a qual potencializa os processos de gerenciamento de projetos e assim cria automações de fluxo de trabalho para eliminar tarefas administrativas repetitivas. Alguns dos softwares que são possíveis de integrar são: Slack, Zoom, Shopify, Google Calendar e dezenas de outras ferramentas (Monday.com, 2017).

Figura 7 - Página de integrações da monday.com



Fonte: https://monday.com

3.6 Kanban Tool

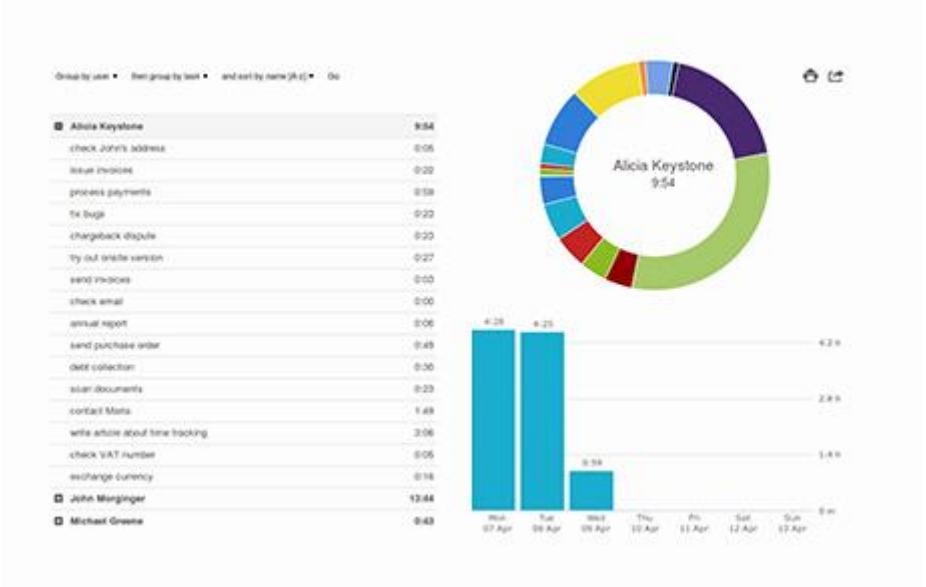
É um aplicativo para gerenciamento de projetos e tarefas que utiliza a metodologia Kanban. Pode-se criar um quadro Kanban personalizado e permitir o acesso a ele para os membros de uma equipe de trabalho, para que todos possam acompanhar o andamento das tarefas. Um quadro Kanban deve consistir em pelo menos três colunas: uma lista de pendências, na qual deve-se reunir todas as tarefas que precisam ser executadas, uma em andamento e uma coluna de tarefas concluídas - onde são colocadas todas as tarefas concluídas e de onde podem ser arquivadas para servir a um propósito analítico. Pode haver quantas colunas forem necessárias, cada uma representando um estágio específico do processo de trabalho específico.

Também é possível identificar e eliminar problemas com análises e métricas Kanban. Fazendo assim um melhor planejamento, monitoramento e consequentemente melhorando o desempenho do projeto utilizando o diagrama de fluxo cumulativo e o relatório de duração de ciclo. (KANBAN TOOL, 2012).

Figura 8 - Board do Kanban Tool

Fonte: https://kanbantool.com

Figura 9 - Análise métrica do Kanban Tool



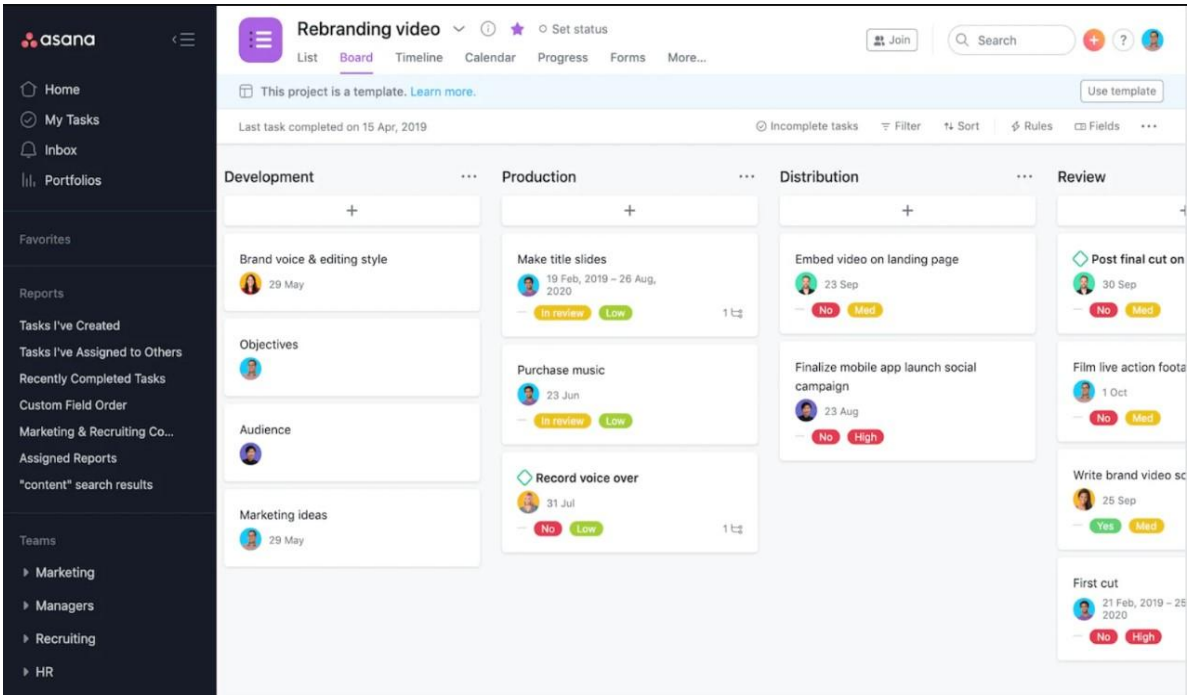
Fonte: https://kanbantool.com

3.7 Asana

Com Asana, usuários podem visualizar projetos e gerenciar tarefas de várias maneiras diferentes, incluindo quadros Kanban, listas, calendários, portfólios, cargas de trabalho e cronogramas, semelhante aos seus concorrentes ClickUp e Monday.com. Cada membro da equipe pode gerenciar tarefas da maneira que funciona melhor para eles, permitindo que sejam o mais produtivos possível. A alternância entre visualizações também oferece aos gerentes de projeto uma perspectiva geral sobre a posição do projeto e pode-se criar painéis de projeto personalizáveis que são atualizados em tempo real.

É possível configurar regras e ações personalizadas utilizando o construtor gráfico de fluxo de trabalho da Asana. Pode-se usá-lo para configurar automações básicas, como datas de vencimento em cascata, atribuir uma tarefa à próxima pessoa no fluxo ou alertar a equipe no Slack quando um projeto é concluído, até mesmo os fluxos de trabalho mais complexos que abrangem dezenas de usuários, ferramentas e tarefas.

Figura 10 - Board da Asana



Fonte: https://asana.com

3.8 Discussões sobre as ferramentas

As ferramentas apresentadas possuem uma grande quantidade de funcionalidades e cumprem o que propõem, porém algumas delas pecam na questão da usabilidade, como é o caso do jira, Monday.com, ClickUp, pois pensando em um único usuário e sendo este um estudante a maioria destas funcionalidades não serão utilizadas e são pensadas na maioria das vezes para times. O board Kanban em si destas aplicações é bastante usual, mas outras funcionalidades se perdem em meio a um tela não muito amigável em termos de

UX. O notion por sua vez tem um visual bem minimalista, mas pelo mesmo motivo do Jira, apresenta alguns aspectos complexos em relação às suas funcionalidades.

O trello e Notion em comparativo com o Jira aparenta ser muito mais intuitivo e simples para ser utilizado e isso se dá pelo fato de não ser tão focado em um meio empresarial, porém ainda sim é muito focado em times, o que não dá uma atenção maior para funcionalidades mais relacionadas a apenas uma pessoa e suas tarefas particulares de estudante. Já o Kanban Tool de todos os softwares apresentados é o que mais se aproxima de uma boa UX, é fácil e simples de ser utilizado e cumpre bem o que faz, porém é focado somente a quadros Kanban e sendo assim não apresenta outras funcionalidades comparado as outras aplicações.

4 Desenvolvimento do projeto

Para este projeto, as tecnologias a serem utilizadas foram pensadas levando em conta sua performance e praticidade. O aplicativo será multiplataforma entre web e *mobile*, por isso é importante a utilização de tecnologias que facilitem o desenvolvimento.

4.1 Tecnologias aplicáveis

Abaixo serão apresentadas algumas tecnologias que são utilizadas para o desenvolvimento do projeto.

4.1.1 ReactJs

O ReactJS[7] foi criado pela empresa Facebook (hoje Meta) que precisava de uma biblioteca para o desenvolvimento de seus produtos. Em 2013 essa biblioteca foi colocada à disposição de outros desenvolvedores e, de lá pra cá, se tornou uma das principais bibliotecas de desenvolvimento em JavaScript.

Por isso, com o intuito de facilitar a criação das interfaces do sistema web em conjunto com a biblioteca de componentes ant design, a qual fornece uma ampla variedade de componentes pré-projetados, como botões, formulários, tabelas e caixas de seleção, que podem ser facilmente incorporados em aplicativos da web. O uso da biblioteca ant design foi considerado para proporcionar uma aparência consistente de componentes tanto na web quanto no mobile, agilizando o processo de desenvolvimento.

React é uma biblioteca Javascript para criação de interfaces. O principal conceito desta tecnologia está nos chamados componentes, que nada mais são do que dividir a interface em diferentes pedaços, criando assim uma melhor abstração e principalmente organização no código do projeto. Outro conceito importante são os estados que cada componente pode ter, eles servem para guardar algum tipo de dado para ser utilizado depois.

---

[7] Disponível: https://pt-br.reactjs.org/

Portanto, por ser fácil e intuitiva de ser utilizada e com uma comunidade bem ativa, foi escolhida React para ser a tecnologia principal para a construção das interfaces do projeto (WALKE, 2013).

4.1.2 Styled Components

Styled Components é uma biblioteca de estilo para React que permite escrever código CSS (Cascading Style Sheets) em componentes de React. Isso significa que os estilos são escritos diretamente no código JavaScript e podem ser aplicados aos elementos HTML (HyperText Markup Language) que o componente renderiza. Uma das principais vantagens de usar Styled Components é que permite manter os estilos bem organizados e anexados aos componentes relevantes. Isso pode ajudar a tornar o código mais legível e fácil de manter, especialmente em projetos maiores. Outra vantagem é que é possível aplicar estilos dinâmicos aos componentes, o que é útil em situações em que os estilos precisam mudar com base em propriedades ou estado do componente (STYLED COMPONENTS, 2015).

4.1.3 Redux

Além disso, será usada a biblioteca Redux[8] para gerenciamento de estados globais, o qual é de suma importância para a organização do código. Redux é uma biblioteca JavaScript de código aberto usada para gerenciar o estado do aplicativo, é amplamente utilizado com React, mas pode ser usado com qualquer outra biblioteca ou framework. É especialmente útil em aplicações de tamanho médio a grande, onde o gerenciamento do estado da aplicação pode se tornar complexo e desafiador. Permite que componentes React leiam dados de um Redux Store e enviem Actions para o Store para atualizar dados. O Redux ajuda os aplicativos a serem dimensionados, fornecendo uma maneira sensata de gerenciar o estado por meio de um modelo de fluxo de dados unidirecional. React Redux é conceitualmente simples, os estados são salvos na “Redux store” e então verifica-se os dados que seu componente deseja foram alterados e os renderiza novamente (ABRAMOV, 2015).

4.1.4 NodeJs

Para o backend será utilizado o NodeJs[9] outra tecnologia do ecossistema javascript que irá ajudar a criar aplicativos de rede rápidos e escaláveis. O modelo de encadeamento único com loop de eventos é muito útil e permite lidar com várias solicitações de clientes. Essa é uma das principais vantagens de usar o NodeJs para aplicações web. Além disso, a sintaxe javascript comum entre front-end e back-end é outro atrativo para a escolha da tecnologia. Portanto, a principal vantagem do uso desta tecnologia para este projeto se dá

[8] Disponível em: https://redux.js.org/
[9] Disponível em: https://nodejs.org/en/

pelo fato da syntaxe javascript, facilitando muito o desenvolvimento da aplicação (DAHL, 2009).

4.1.5 Postgresql

PostgreSQL[10] é um sistema de gerenciamento de banco de dados relacional de código aberto (open source). Foi desenvolvido como um fork do projeto POSTGRES, que foi iniciado na Universidade de Califórnia, Berkeley, nos Estados Unidos, em 1986. Desde então, tem se tornado um dos sistemas de gerenciamento de banco de dados mais populares e avançados do mundo. PostgreSQL é conhecido por sua robustez, flexibilidade e escalabilidade. Suporta muitos tipos de dados diferentes, incluindo texto, números, geometria e tipos de dados de data/hora. Também oferece uma ampla variedade de ferramentas de análise de dados, incluindo suporte para consultas SQL avançadas, índices e triggers. Além disso, o PostgreSQL é altamente compatível com padrões da indústria e é amplamente utilizado em aplicativos empresariais, científicos e governamentais ao redor do mundo. Ele é compatível com muitos sistemas operacionais diferentes, incluindo Linux, MacOS e Windows, e é compatível com muitas linguagens de programação (STONEBRAKER, 1986).

4.2 Requisitos do projeto

4.2.1 Tabelas de requisitos

A seguir será apresentado os requisitos do sistema, baseado nas pesquisas realizadas e das necessidades dos estudantes.

Tabela 2 - Tabela de requisitos

Ident	Requisito	
R1	Logar e deslogar do sistema	**Obrigatório**
R2	Cadastrar disciplinas: cada disciplina terá um id, nome e uma cor	**Obrigatório**
R3	Remover disciplinas	**Obrigatório**
R4	Cadastrar uma tarefa: cada tarefa terá um id, titulo e descrição	**Obrigatório**
R5	Editar uma tarefa	**Obrigatório**
R6	Adicionar uma tarefa no calendário: para cada disciplina poderá ser adicionada uma tarefa com uma data e hora para sua finalização	**Obrigatório**

[10] Disponível em: https://www.postgresql.org

R7	Exibir as tarefas das disciplinas no board: as tarefas das disciplinas poderão ser arrastadas em um board com os seguintes estados: “Tarefas”, “Fazendo”, “Concluído”	**Obrigatório**
R8	Filtrar board das disciplinas: O usuário irá poder filtrar por disciplina, prioridade e tempo de término (30 dias, 15 dias, 7 dias)	**Obrigatório**
R9	Remover tarefa do board	**Obrigatório**
R10	Exibir chat para comunicação entre alunos: O sistema terá um meio de comunicação via texto.	**Desejável**
R11	Visualizar detalhes das disciplinas	**Obrigatório**
R12	Visualizar disciplinas	**Obrigatório**
RN1	Linguagem Javascript	**Obrigatório**
RN2	Idioma: Português	**Obrigatório**
RN3	Banco de dados: Postgresql	**Obrigatório**
RN4	IDE de desenvolvimento: Visual Studio Code	**Obrigatório**

Fonte: Elaboração do autor, 2022

4.2.2 Histórias de usuário

- Como **estudante**, eu gostaria de conseguir **organizar** melhor minhas **tarefas** da faculdade para que eu consiga obter um **melhor resultado** nas minhas **notas**.

- Para ter um **melhor proveito** do semestre eu como **estudante** quero ter um **calendário organizado** com as **tarefas** que preciso fazer.

- Como **estudante**, gostaria de ter um meio em que possa fazer **anotações** em texto rico das disciplinas e poder **compartilhá-las** com colegas a fim de ter **resumos** melhor elaborados.

- Para conseguir melhor **visualizar** as minhas **tarefas**, eu como **estudante** quero ter um **board** estilo **kanban** para que assim tenha uma melhor noção de **progresso** das minhas tarefas.

4.2.3 Mockups

Uma primeira iteração com os requisitos listados no item 4.2 permitiu gerar as propostas de interfaces apresentadas abaixo, com suas respectivas funcionalidades. Tais telas poderão ser revisadas/alteradas à medida que forem sendo desenvolvidas as funcionalidades definitivas.

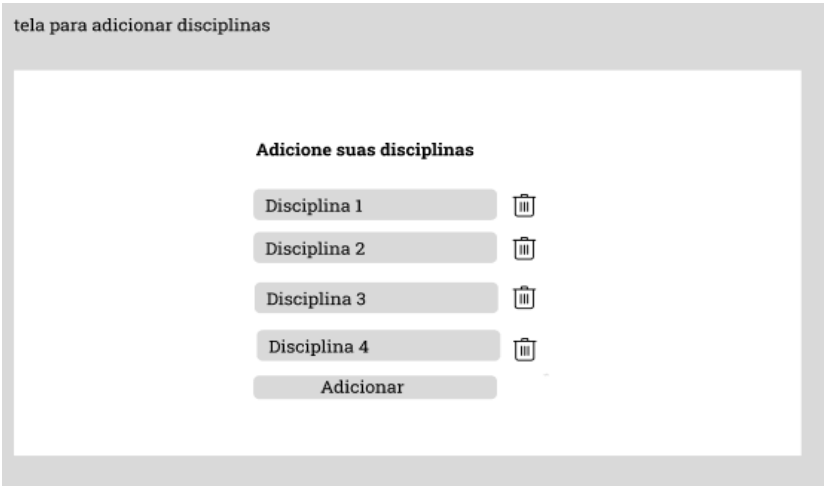
Tela de login (Requisito R1)

Figura 11 - Mockup tela de login

Fonte: Elaboração do autor, 2022

Tela para adicionar as disciplinas (Requisito R2)

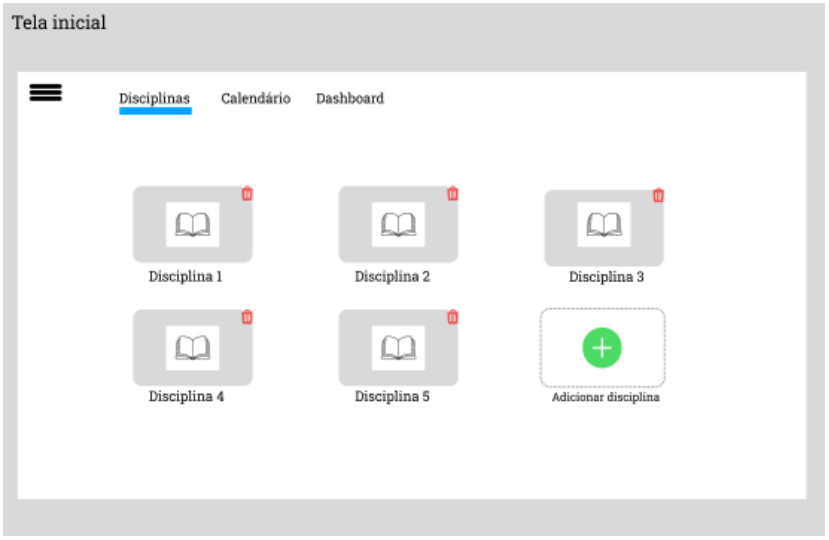
Figura 12 - Mockup para adicionar as disciplinas



Fonte: Elaboração do autor, 2022

Tela das disciplinas (Requisito R12)

Figura 13 - Mockup da tela de disciplinas



Fonte: Elaboração do autor, 2022

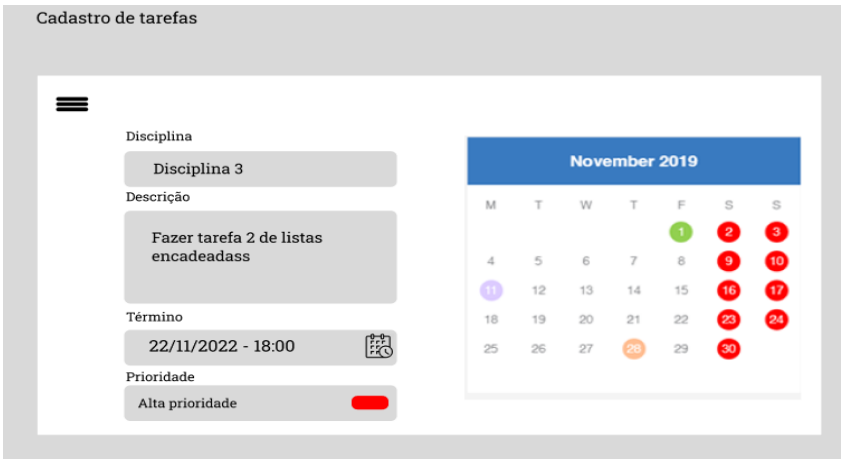
Tela de detalhes de uma disciplina (Requisito R11)

Figura 14 - Mockup da tela detalhes de uma disciplina

Fonte: Elaboração do autor, 2022

Cadastro de tarefas no calendário (Requisito R6)

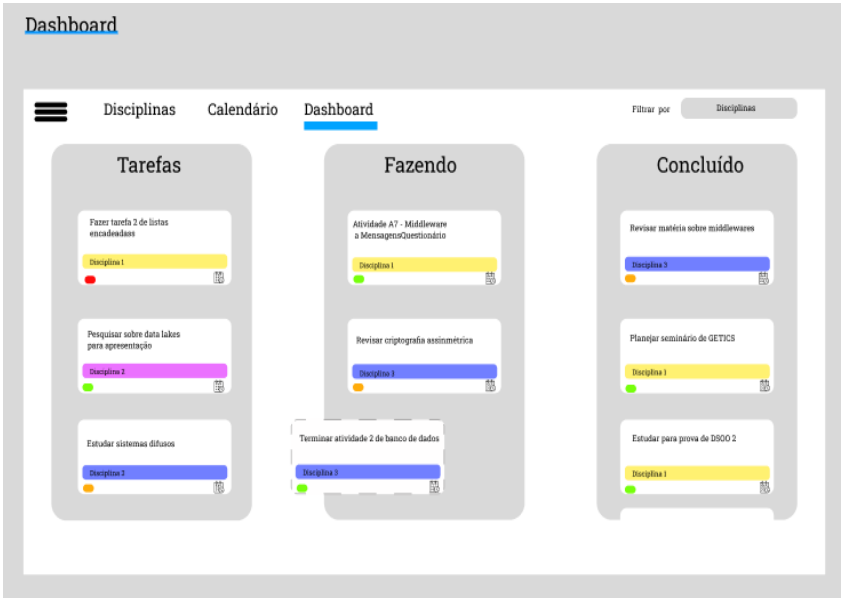
Figura 15 - Mockup da tela de cadastro de uma tarefa no calendário



Fonte: Elaboração do autor, 2022

Board (Requisito R7)

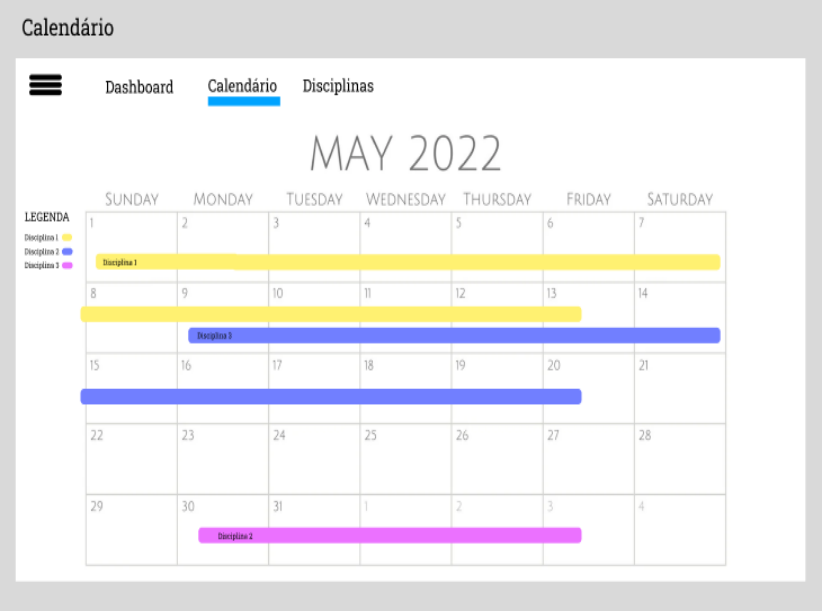
Figura 16 - Mockup da tela de board



Fonte: Elaboração do autor, 2022

Calendário (Requisito R6)

Figura 17 - Mockup da tela de calendário



Fonte: Elaboração do autor, 2022

4.3 Proposta de projeto

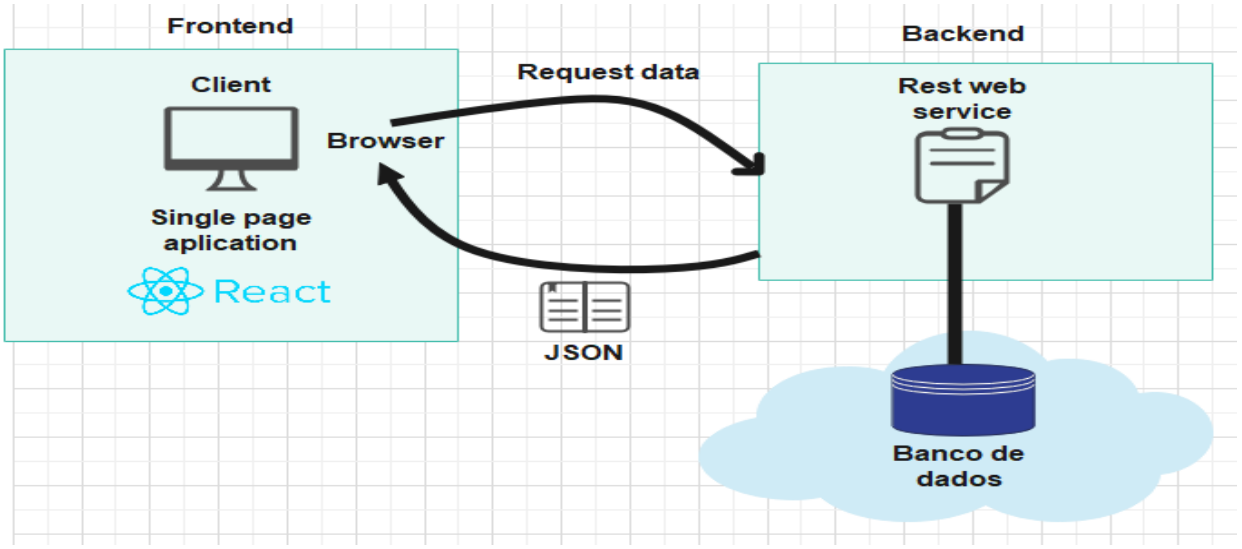
A seguir será apresentada uma proposta de como será executado o projeto em termos arquiteturais de software.

4.3.1 Arquitetura e tecnologias

À luz dos requisitos registrados no item 5.2.1, foi modelada a seguinte arquitetura para dar suporte a um sistema como o proposto:

Aplicação Web

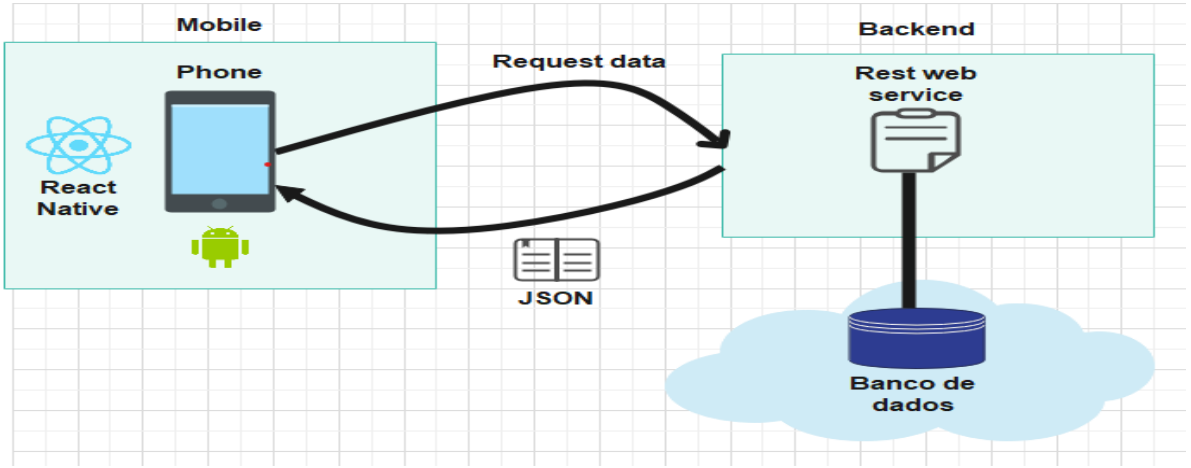
Figura 18 - Arquitetura da aplicação web



Fonte: Elaboração do autor, 2022

Aplicação Mobile

Figura 19 - Arquitetura da aplicação mobile



Fonte: Elaboração do autor, 2022

Pode-se observar na imagem acima, uma arquitetura de cliente servidor, onde o frontend será um Single Page Application (SPA) e o backend utilizará Representational State Transfer (REST) para a construção dos webservices que por sua vez “entregará” os dados no formato de JavaScript Object Notation (JSON) para o Cliente.

Já na aplicação mobile haverá uma arquitetura muito parecida onde a comunicação será feita de forma igual a da web através de webservices e também recebendo um JSON como resposta.

4.4 FRONTEND

O front-end da aplicação foi desenvolvido utilizando o editor de código Visual Studio e utilizando a linguagem Javascript juntamente com a biblioteca para criação de interfaces ReactJs pois essa tecnologia apresenta diversas vantagens como performance, velocidade, usabilidade, fácil de se aprender, além de ter uma grande comunidade e apresentar a maior porcentagem de frameworks mais amados pelos desenvolvedores segundo pesquisa[11] realizada pelo site stack-overflow. Para facilitar a criação dos componentes, foi utilizado a biblioteca ant design e styled components para facilitar na estilização.

Para facilitar na explicação de cada desafio técnico tido, será apresentada uma separação em cada tela do sistema:

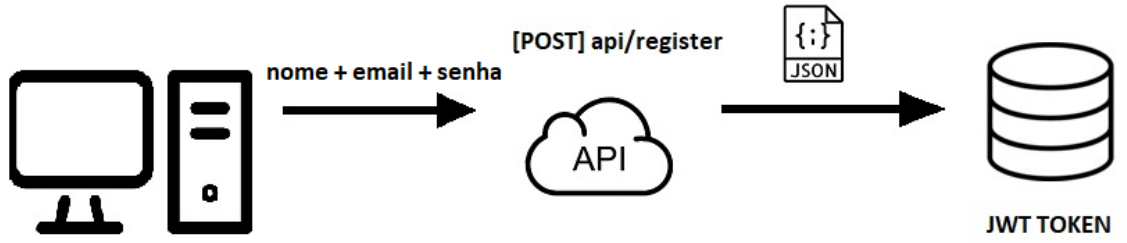
- Tela de Login
- Tela de Board
- Tela de Disciplina
- Tela de Anotações
- Tela de agenda

4.4.1 Tela de Login e Cadastro

O login é realizado através de uma interface simples contendo os campos de email e senha, o mesmo para o cadastro porém com nome, email e senha, não foi feita nenhuma verificação de confirmação de email, algo comum entre as aplicações, já que segurança não é o foco deste trabalho e sim foi feito algo mais simplificado a fim de obter um MVP (Minimum Viable Product). Abaixo pode-se observar um esquema como funciona a lógica no frontend para que o usuário consiga logar na plataforma:

[11] Disponível em: https://survey.stackoverflow.co/2022/#technology

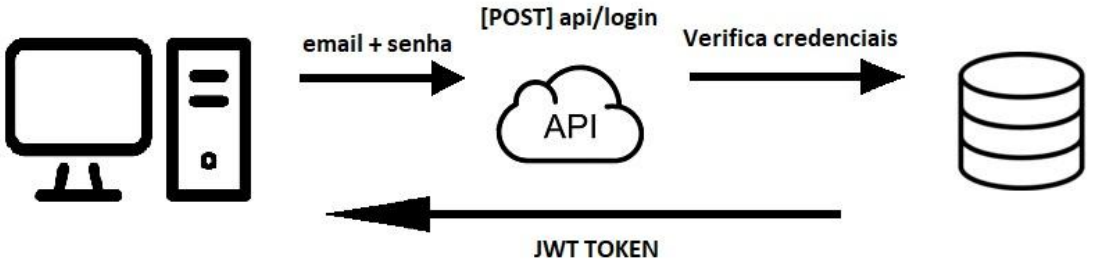
Figura 20 - Esquema do processo de cadastro



Fonte: Elaboração do autor, 2022

Como pode-se observar na imagem acima o usuário insere o nome, email e senha, estes são enviados para a api que é salvo no formato JSON para o banco de dados, por fim utilizado o algoritmo de criptografia, um JWT (JSON Web Token) é gerado e salvo no banco de dados.

Figura 21 - Esquema do processo de login

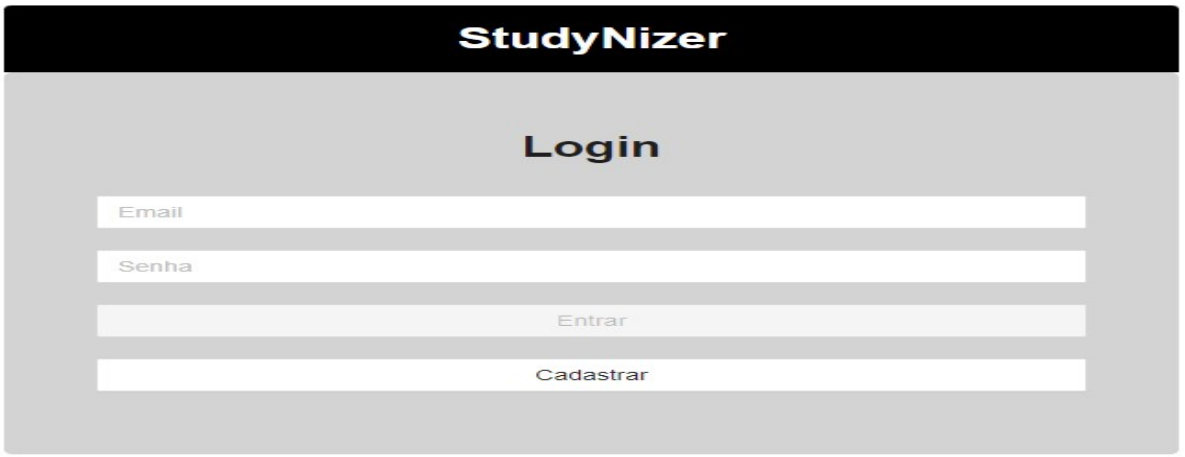


Fonte: Elaboração do autor, 2022

De maneira similar para o login, o usuário insere o email e senha, a api verifica as credenciais, se forem válidas, um JWT token é enviado para o usuário através de uma requisição HTTP (Hypertext Transfer Protocol). Para que o frontend identifique se este possui o JWT. Foi utilizado o local storage do navegador e assim feita uma verificação através das rotas da aplicação para que ele seja redirecionado para a primeira tela principal do sistema. Como é mostrado no **anexo 1,** após a requisição ser feita é salvo no local storage através da chave *@StudyNizer:userSession* que será utilizada em seguida para o redirecionamento de rota da aplicação. No arquivo *Router* em **anexo 2** observa-se que o token é pego através do local storage e assim enviado para o componente AuthLayout em **anexo 3** que é responsável por fazer a verificação da disponibilidade do Token no frontend e também o redirecionamento para a tela principal da aplicação.

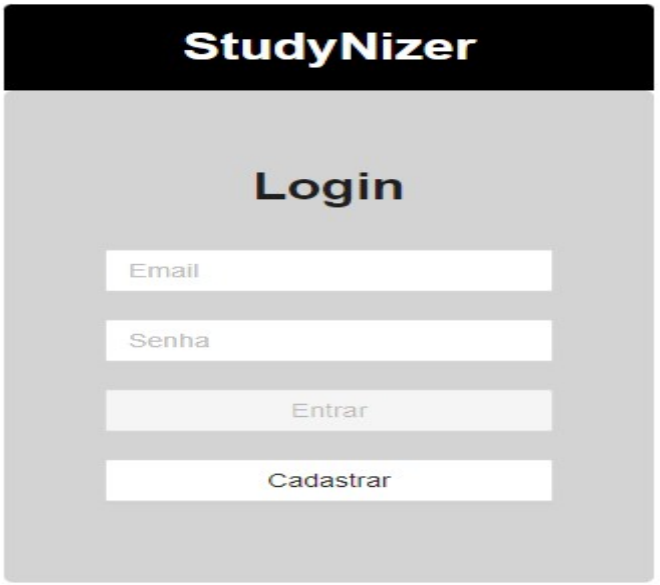
Abaixo a tela de login na versão para web desktop e mobile.

Figura 22 - Tela de login do StudyNizer



Fonte: Elaboração do autor, 2022

Figura 23 - Tela de login do StudyNizer versão mobile

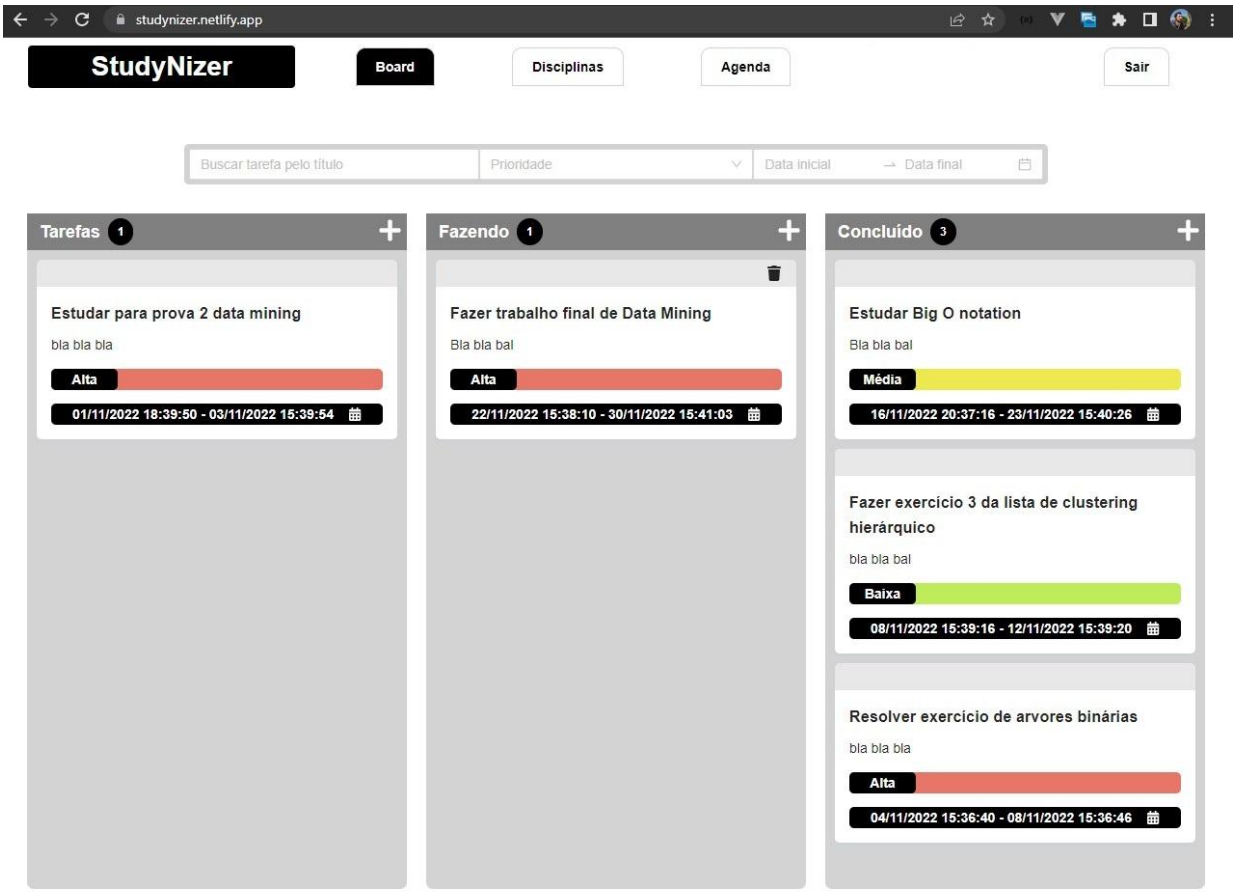


Fonte: Elaboração do autor, 2022

4.4.2 Tela de Board

Uma vez que o usuário tem suas credenciais validadas este é redirecionado para o board, onde ele irá fazer o acompanhamento das tarefas. A tela foi desenvolvida utilizando a biblioteca *react-beautiful-dnd*, criada pela empresa atlassian, a qual facilita a escrita do código para a funcionalidade de *drag-and-drop,* que é fundamental para o quadro Kanban. Nesta tela é feito um CRUD (Create, Read, Update and Delete) completo, o usuário pode adicionar, editar, deletar e listar tarefas, além de filtrar os cards pelo título, nível de prioridade e data. A seguir são apresentadas as telas desta parte do sistema:

Figura 24 - Tela de board do StudyNizer



Fonte: Elaboração do autor, 2022

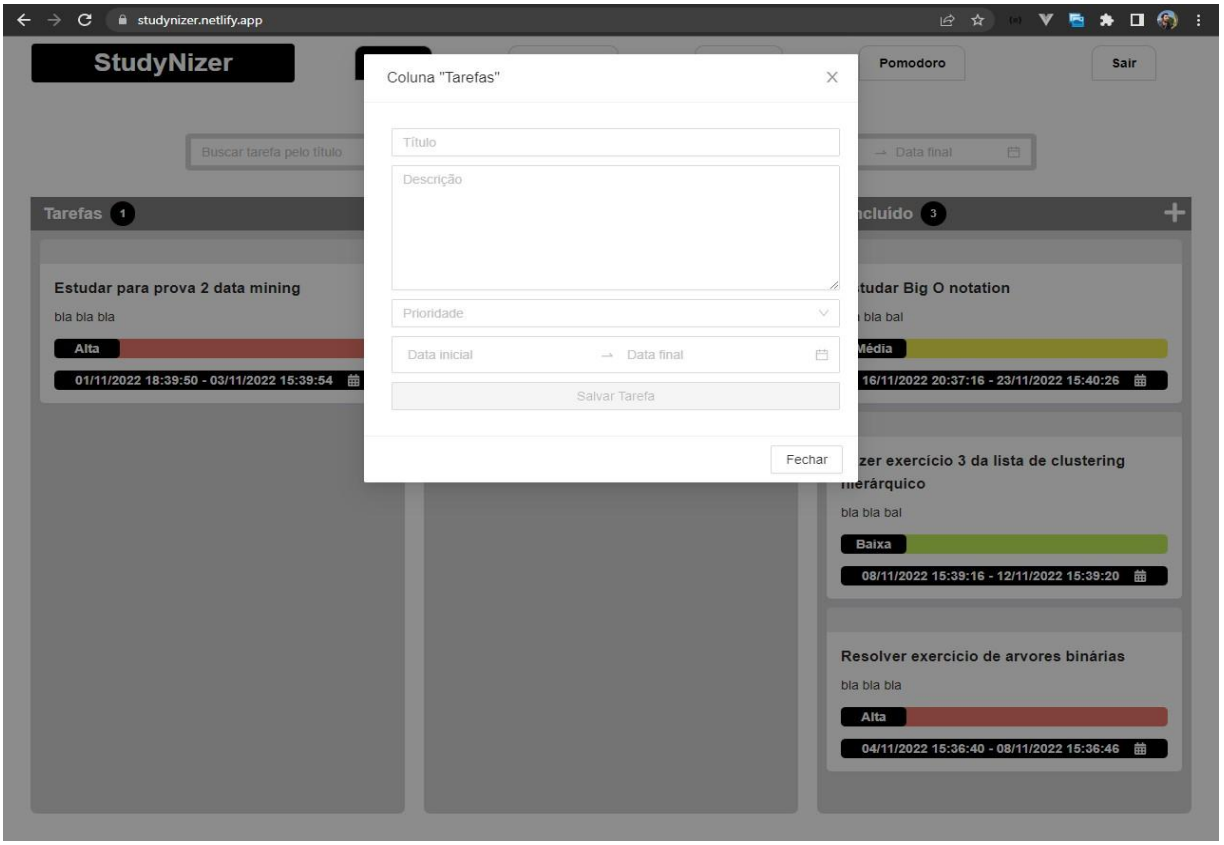
O board apresenta três colunas de nomes intuitivos para cada status de progresso , que são:

- Tarefas: Representa os cards que o usuário precisa fazer mas ainda não deu início.
- Fazendo: Representa os cards que o usuário começou a fazer.
- Concluído: Representa os cards que o usuário finalizou e serve também como um histórico.

Ao clicar no ícone de “adição” no topo da coluna um *modal* se abre com os campos: Título, Descrição, Prioridade e Data de início e Fim para serem atribuídos a uma tarefa. Ao passar o mouse por cima de uma card um tooltip

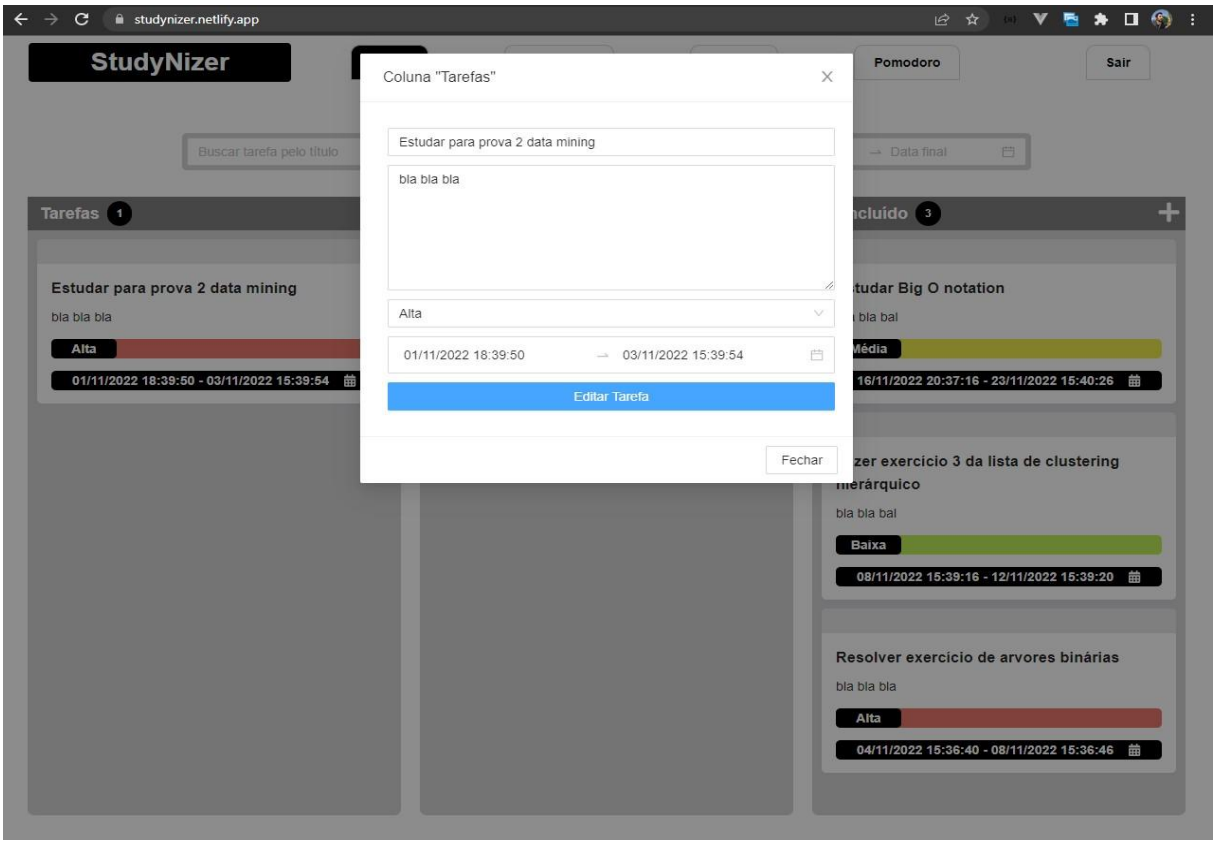
aparece com uma mensagem “Clique para ver detalhes desta tarefa”, com os dados pré-preenchidos do card selecionado e assim o usuário consegue editar as informações se necessário.

Figura 25 - Modal para adicionar uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

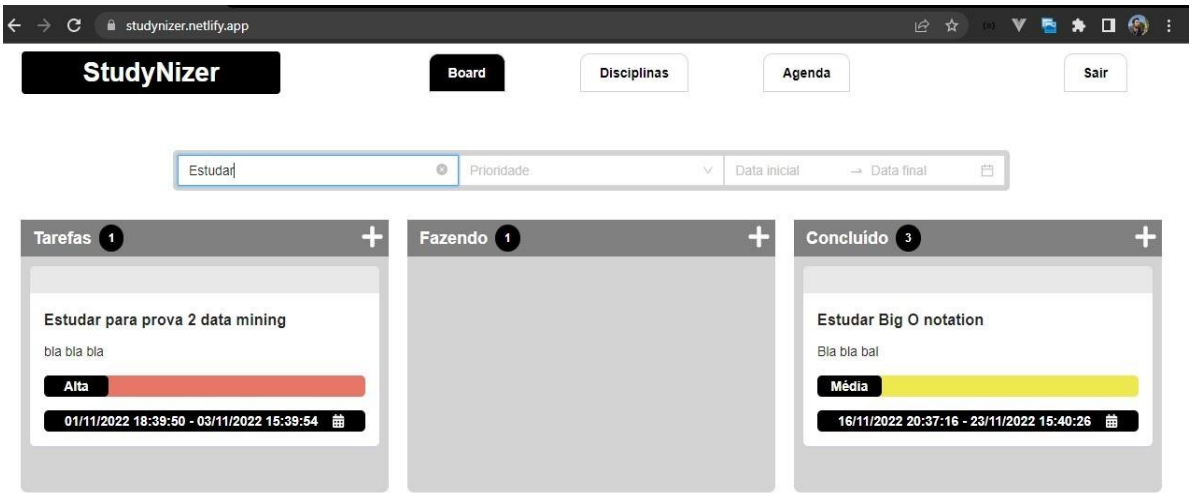
Figura 26 - Modal para editar uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

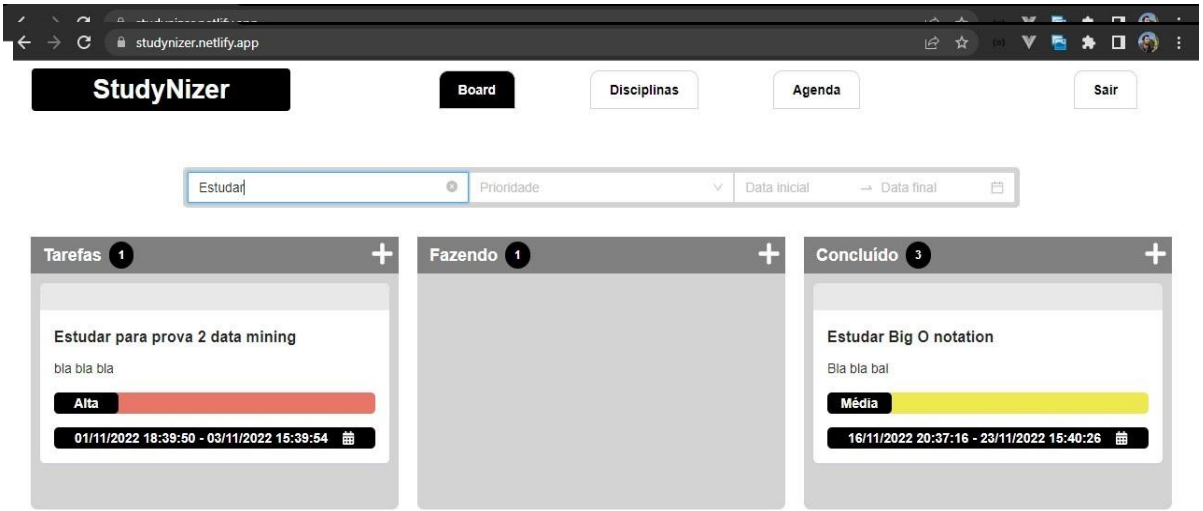
As próximas imagens são referentes a funcionalidade de filtro dos cards onde é possível filtrar pelo título, prioridade e data.

Figura 27 - Filtro por título de uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

Figura 28 - Filtro por data de uma tarefa do StudyNizer



Fonte: Elaboração do autor, 2022

O código para realizar a função de *drag-and-drop* pode ser separada em dois momentos, o primeiro sendo a mudança de um card de uma coluna para outra e assim salvando no estado da aplicação, foi utilizado a api de hook do react *useState* que serve para “salvar na memória" uma certa informação e poder capturá-la ou modificá-la dependo da necessidade. Como pode-se observar no **anexo 3**, tem-se uma estrutura *json* em um formato chave valor

para cada coluna. Na função *handleDragEnd* em **anexo 4** é onde é feita a lógica para identificar onde deve-se posicionar os cards e salvá-los no estado, utilizando o *setState.*

O segundo momento seria salvar no banco de dados, no backend existe três *endpoints* para cada coluna:

- [POST] /board-tasks-todo
- [POST] /board-tasks-doing
- [POST] /board-tasks-completed

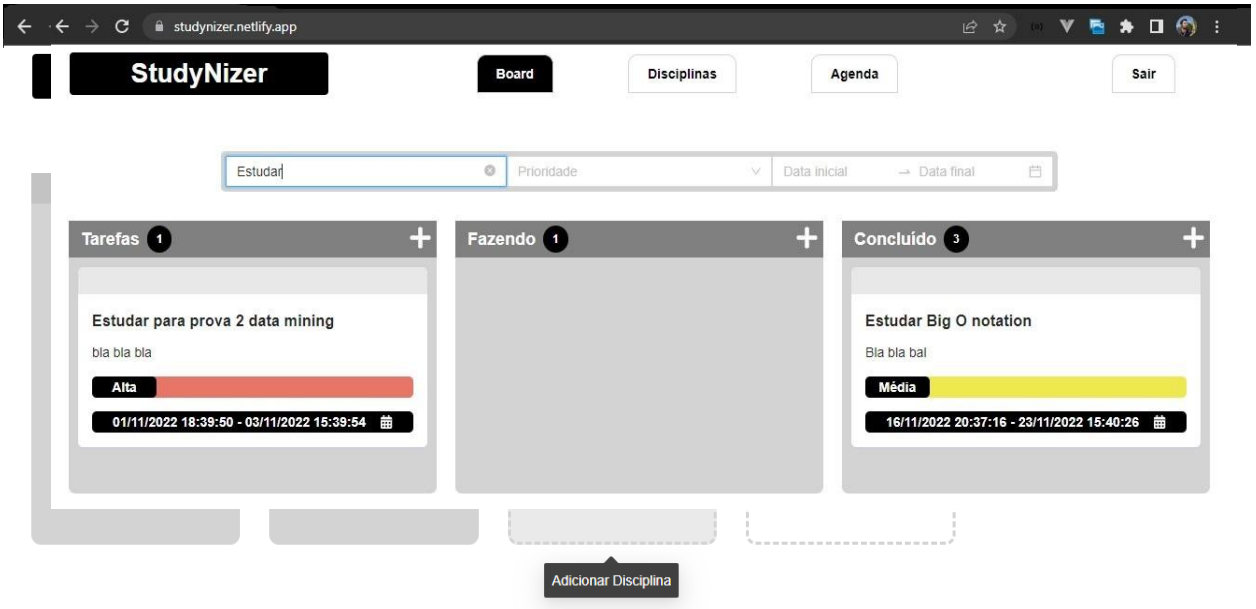
Foi realizado uma lógica no frontend para identificar a partir da detecção do *hover* do mouse do usuário qual coluna deve-se inserir o card selecionado e assim acionar o enpoint para salvar no banco de dados, como pode ser observado no **anexo 5**.

4.4.3 Tela de Disciplina

A tela de disciplina é onde o usuário irá registrar o nome das matérias para em seguida ter os resumos de cada uma. As funcionalidades desta tela permitem a adição de uma disciplina, excluir, importar e exportar os arquivos com seus respectivos resumos para que um usuário possa compartilhar seus resumos com um colega.

Conforme as disciplinas são adicionadas, estas são apresentadas em forma de cards, como pode-se observar abaixo também há dois botões para adicionar e importar uma disciplina. Ao passar o mouse sobre um dos cards fica visível para o usuário dois ícones, um de exclusão e outro para exportar a disciplina.

Figura 29 - Tela de disciplinas do StudyNizer



Fonte: Elaboração do autor, 2022

Figura 30 - modal para adicionar uma disciplina do StudyNizer

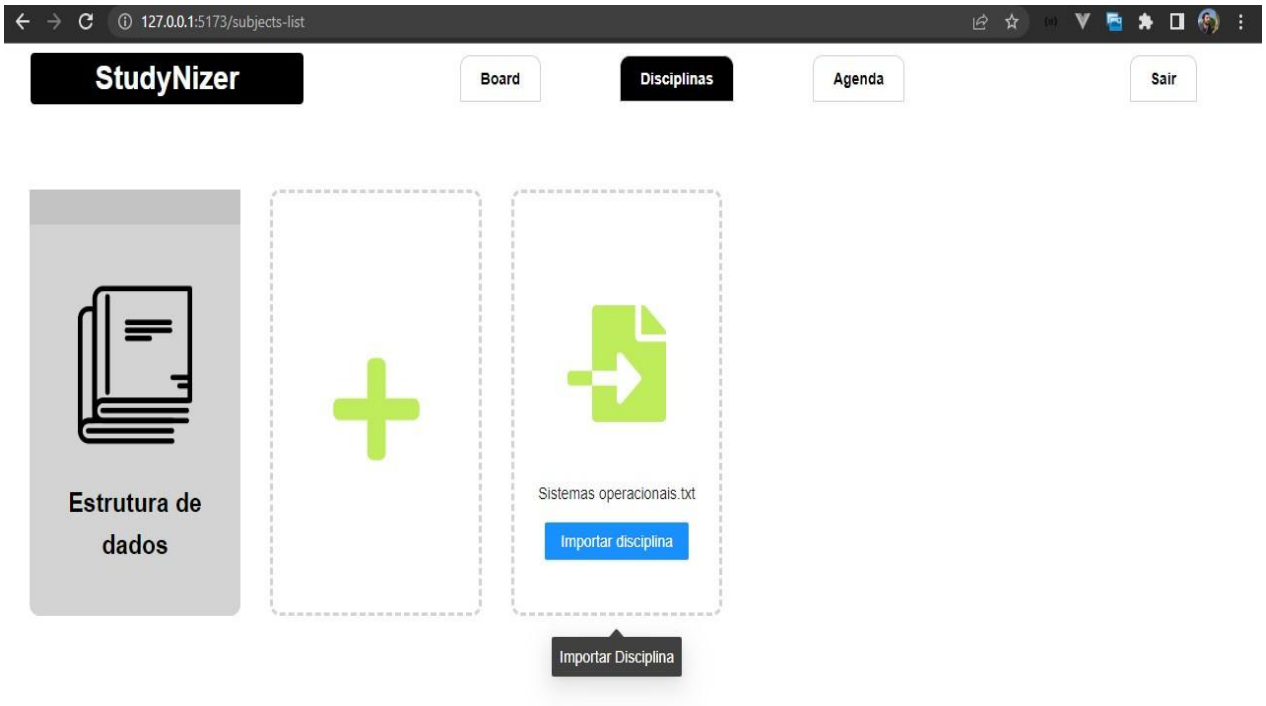
Fonte: Elaboração do autor, 2022

Figura 31 - Exportando uma disciplina

Fonte: Elaboração do autor, 2022

Na imagem abaixo mostra um arquivo em que o usuário selecionou de seu computador para ser importado e em seguida ele terá todos os resumos feitos por um outro usuário.

Figura 32 - Importando uma disciplina



Fonte: Elaboração do autor, 2022

O desafio técnico maior desta etapa foi sem dúvidas a lógica para importar e exportar as disciplinas, foi utilizado a biblioteca *file-saver* que é responsável por salvar arquivos no formato txt. No **anexo 6** podemos ver como isto se faz presente no código.

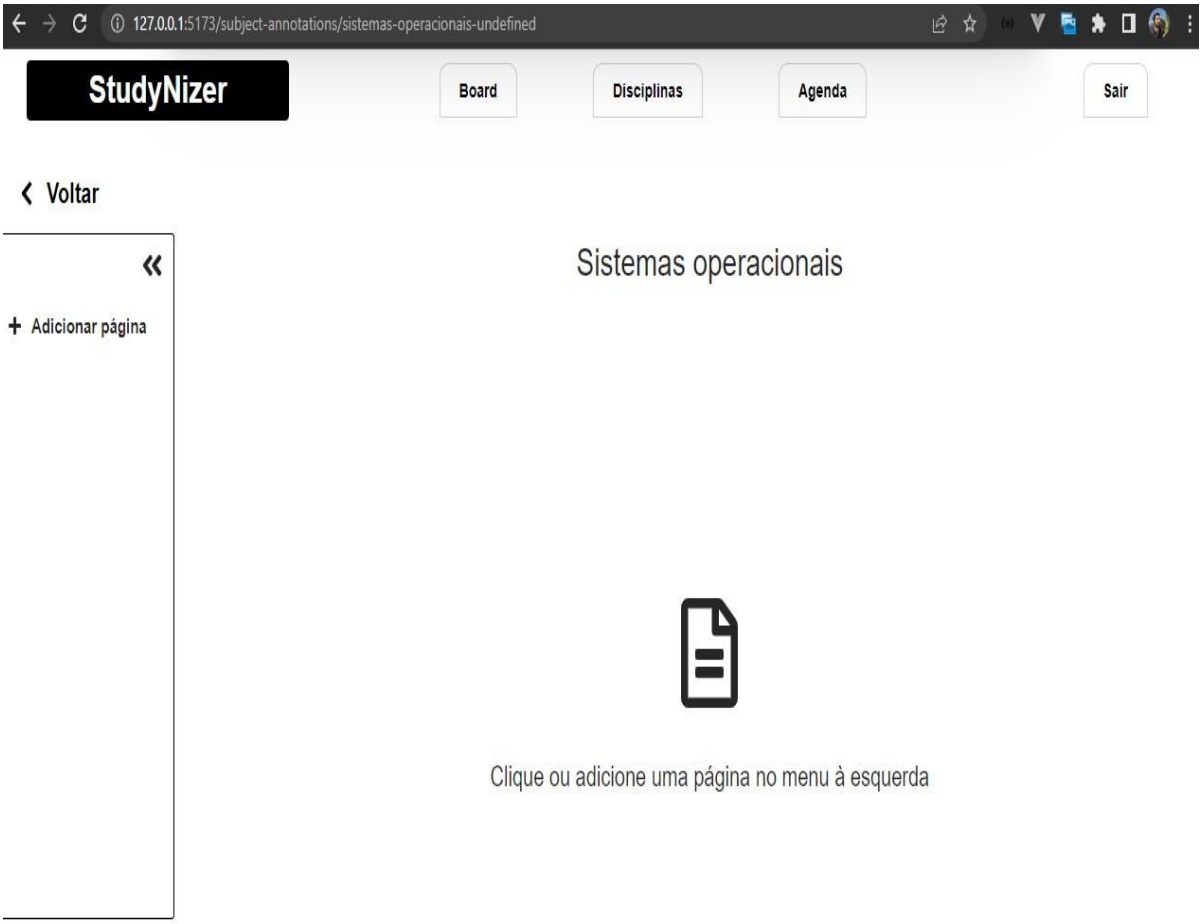
Para a exportação é feita uma chamada ao endpoint [GET] /user/markdown/:id, através deste tem-se o conteúdo das anotações de cada disciplina e com isso foi convertido a resposta da requisição para o formato *blob,* que será usado como primeiro parâmetro na função *FileSaver.saveAs* e como segundo o nome do arquivo.

Na parte da importação, como pode ser observado no **anexo 7**, foi realizada uma chamada para o endpoint [POST] /user/markdownImport que irá receber o conteúdo do arquivo exportado, também foi feita uma verificação para saber se a disciplina já foi importada.

4.4.4 Tela de Anotações

Ao clicar em uma matéria na tela de disciplina, o usuário é redirecionado para a tela de anotações, nesta o aluno poderá adicionar uma página para um determinado resumo, em seguida tem disponível uma área com texto rico. A configuração inicial desta seção indica o que o usuário deve fazer, neste caso adicionar uma página no menu lateral.

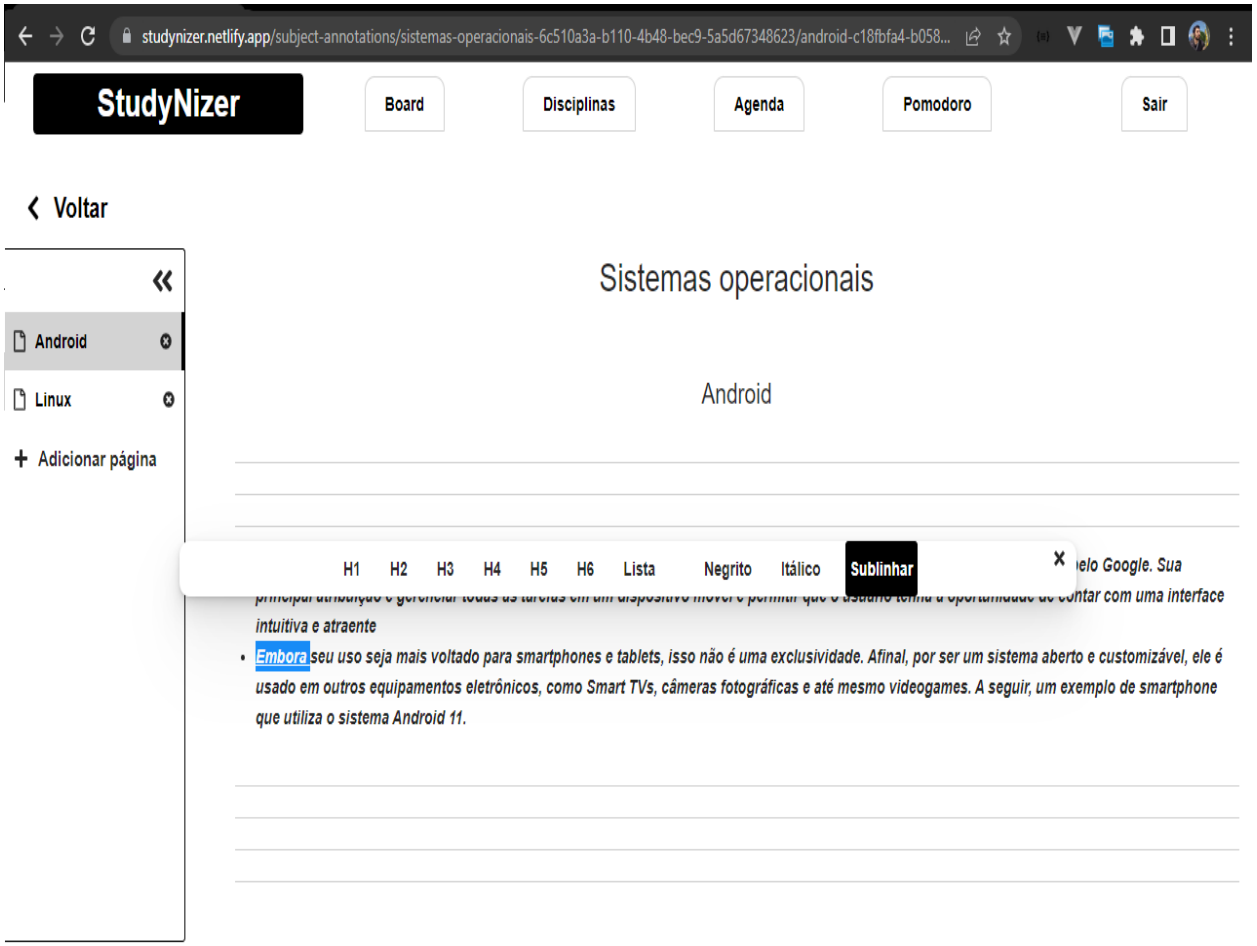
Figura 33 - Tela de anotações do StudyNizer



Fonte: Elaboração do autor, 2022

Adicionada a página, obtém-se agora disponível uma área de texto rico para escrever, ao selecionar uma linha, surge um painel para o usuário customizar seus resumos com as opções para aumentar o tamanho da fonte, criar listas, negrito, itálico e sublinhado, ao digitar este painel some para não interferir na visão enquanto o usuário digita.

Figura 34 - Caixa de texto rico



Fonte: Elaboração do autor, 2022

Foi utilizada a biblioteca *draft-js* como pode-se observar no **anexo 8,** para o texto rico, o componente <Editor /> representa o campo de texto e com a propriedade editorState capturamos o conteúdo escrito pelo usuário na função handleChangeEdtior é onde é feita a lógica para salvar no banco de dados.

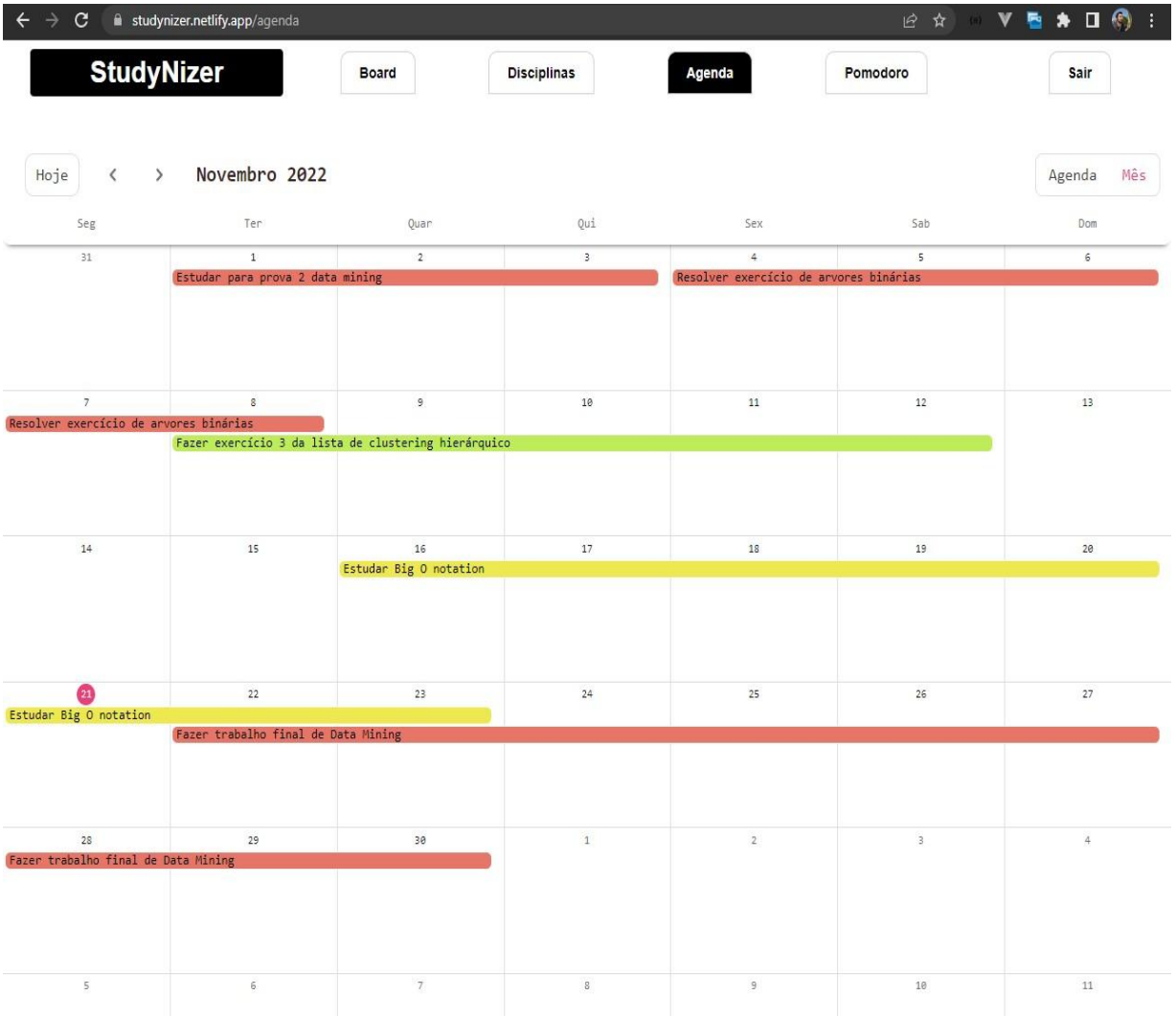
No código mostrado do **anexo 9,** o conteúdo do texto é passado para uma função converRaw da biblioteca *draft-js,* isso irá converter as informações para um formato de texto, para ser aceito pelo banco de dados.

É feita uma chamada para uma função de *debounce* que é comumente utilizada em situações onde precisa-se salvar o conteúdo digitado pelo usuário apenas uma vez quando este terminar de digitar, neste caso após um segundo. Desta forma prevenimos chamadas abundantes para a api [PUT] /user/markdown/:id, onde é feito o registro do conteúdo do texto no banco de dados como observado no **anexo 10**.

4.4.5 Tela de Agenda

Ao adicionar uma tarefa na tela de board, estas aparecerão automaticamente na agenda, a qual possui dois modos de visualização, “agenda” e “Mês”, No modo agenda as tarefas são listadas em formato vertical com suas respectivas datas, horários, nível de prioridade representado por cores e o título.

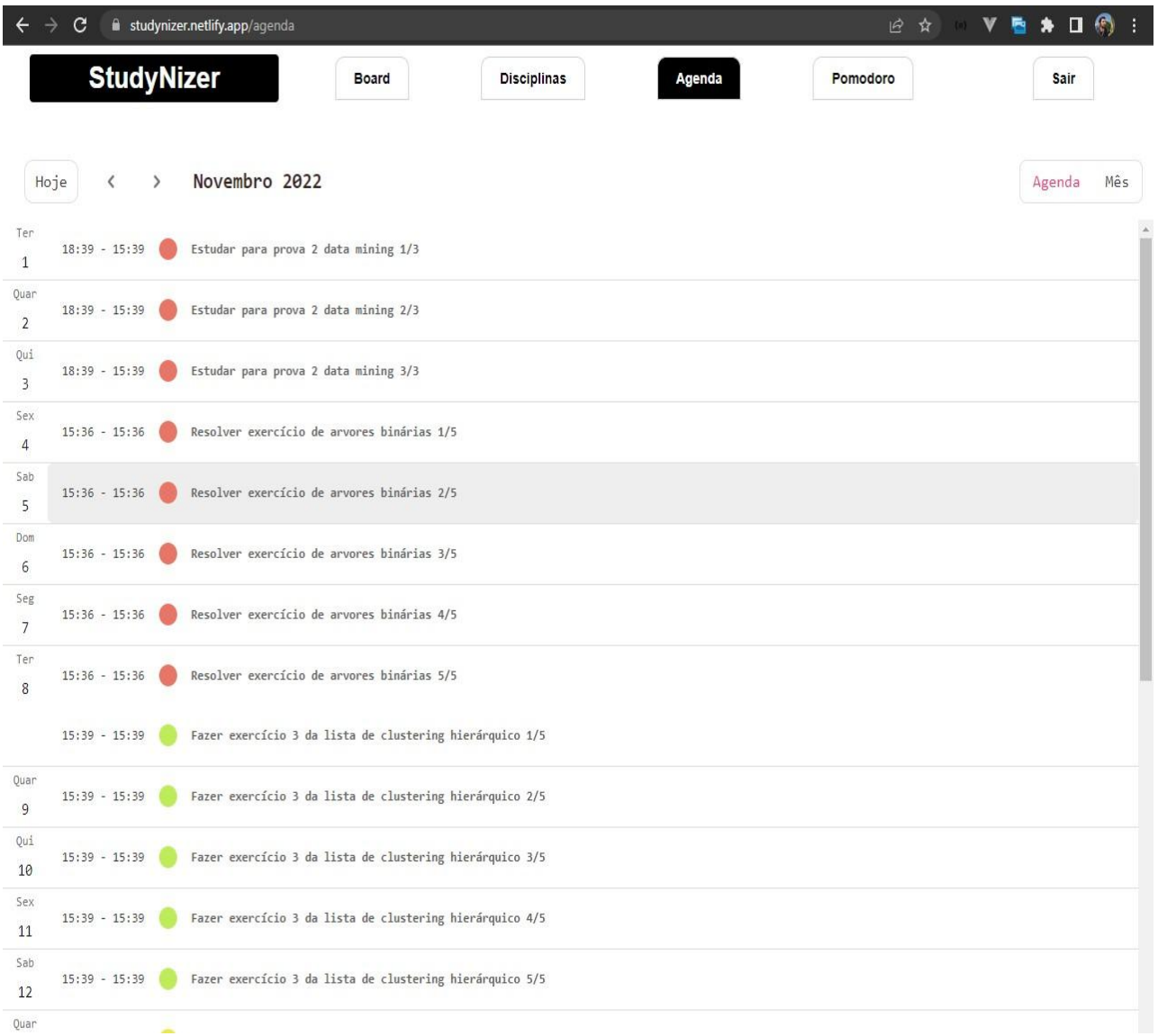
Figura 35 - Tela de calendário do StudyNizer



Fonte: Elaboração do autor, 2022

Já no modo de mês observa-se uma visualização em modo de calendário, representada por linhas, desta forma o usuário tem uma visão melhor das tarefas a serem feitas ao decorrer dos dias.

Figura 36 - Tela de agenda do StudyNizer



Fonte: Elaboração do autor, 2022

Foi utilizada a biblioteca *kalend* para a implantação do calendário, uma de suas propriedades deste componente é o *events* onde é passado as tarefas e assim listados para o usuário como observado no código do **anexo 11**.

Para disponibilizar as tarefas no calendário foi realizada três chamadas aos endpoints:

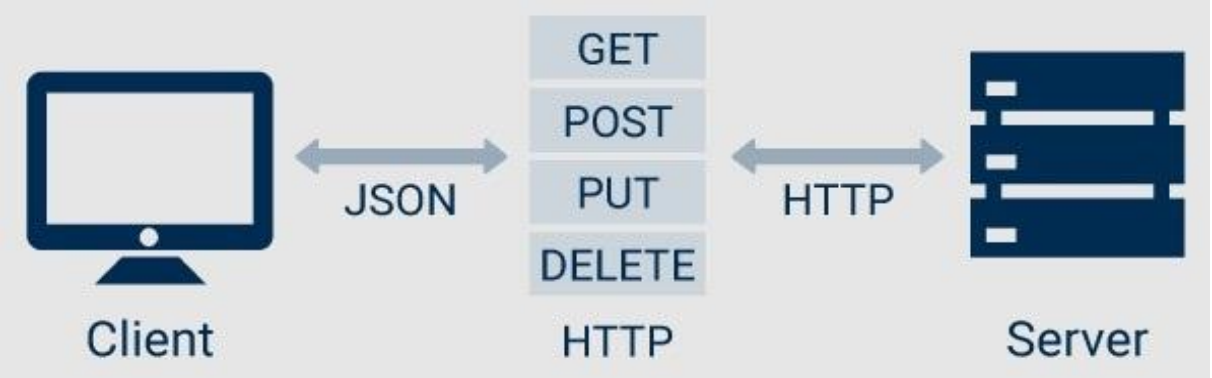
- [GET] /user/board/tasks-todo
- [GET] /user/board/tasks-doing
- [GET] /user/board/tasks-completed

Em seguida foi passado a resposta destas requisições para a função *generateDemoEvents* que é responsável por preencher as tarefas no calendário como mostrado no **anexo 12**.

4.5 BACKEND

Para o backend da aplicação, foram utilizadas ferramentas que são hoje populares no mercado e além de tudo performáticas, além de usar Rest como arquitetura de software. As API's (Application Programming Interface) vêm crescendo rapidamente. Os desenvolvedores passaram do SOAP (Simple Object Access Protocol) ou RPC (Remote Procedure Call) para implantar o REST (Representational State Transfer), como o meios para os consumidores usarem seus serviços. Isto é corroborado por grandes sites como Google, Facebook ou Twitter, que agora estão implantando serviços REST para fornecer acesso fácil a seus recursos. Esta arquitetura fornece padrões entre sistemas de computador na web, tornando mais fácil para os sistemas se comunicarem uns com os outros. Os sistemas compatíveis com REST, geralmente chamados de sistemas RESTful, seguem os "cinco verbos RESTFUL" ou os "cinco princípios RESTFUL" para garantir a integridade e a interoperabilidade dos dados trocados entre clientes e servidores. Um sistema não é RESTFULL se não seguir os cinco princípios. Por exemplo, se um sistema mantém o estado entre as requisições ou não usa verbos HTTP para operar sobre recursos, ele não é RESTFUL. (NEUMANN, 2018).

Figura 37 - Esquema do fluxo Cliente servidor



Fonte: https://holdertech.com.br

Outra tecnologia utilizada foi o NodeJs que por ser escrita em javascript facilita a implementação do projeto principalmente para desenvolvedores com experiência em front-end, é muito utilizada entre diversas startups e também empresas de grande porte, Possui uma arquitetura orientada a eventos capaz de implementar eventos de E/S assíncrona (é uma forma de processamento de entrada/saída que permite que outro processamento continue antes da transmissão). Destina-se a otimizar a escalabilidade em aplicações web (DAHL, 2009).

Para a comunicação com o banco de dados foi utilizado o postgresql, que é um sistema de banco de dados objeto-relacional de código aberto que usa e estende a linguagem SQL. Conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para oferecer consistentemente soluções inovadoras e de alto desempenho, é executado em todos os principais sistemas operacionais. (STONEBRAKER, 1986).

Para a interação e testes com as APIs foi utilizado um aplicativo desktop chamado Insomnia[12], uma tecnologia multi plataforma gratuita que facilita a interação e o design de API's baseadas em HTTP. O Insomnia combina uma interface fácil de usar com funcionalidades avançadas, como auxiliares de autenticação, geração de código e variáveis de ambiente, é primordial a utilização de tal aplicação para ajudar a descobrir bugs, anomalias ou discrepâncias do comportamento esperado de uma API durante o desenvolvimento (KONG, 2017).

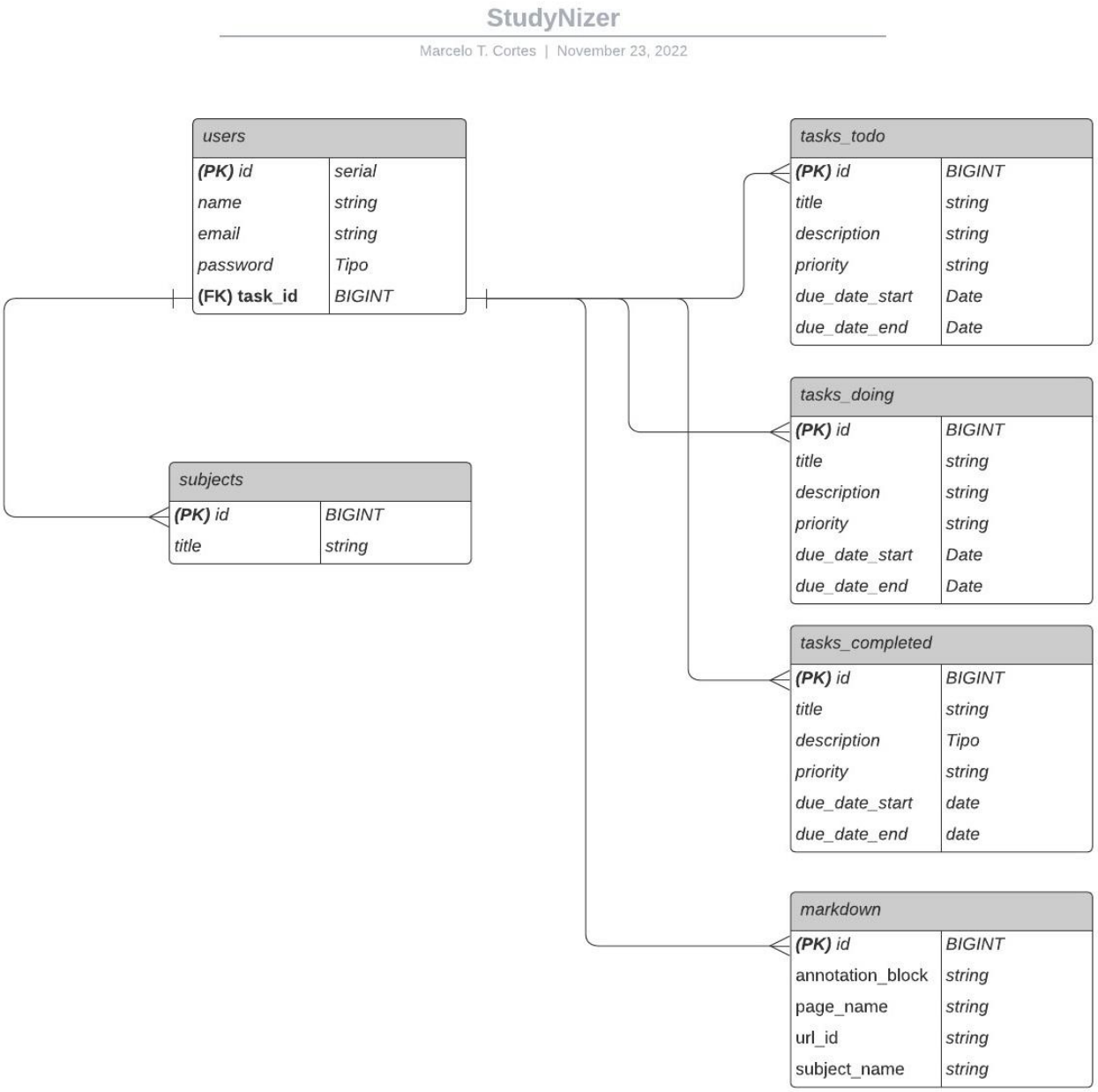
A modelagem do banco de dados se fez com as seguintes tabelas:

- users: Responsável por guardar os usuários.
- subjects: Responsável por guardar as disciplinas de cada usuário..
- tasks_todo: Responsável por guardar as tarefas que precisam ser feitas..
- tasks_doing: Responsável por guardar as tarefas que estão sendo feitas..
- tasks_completed: Responsável por guardar as tarefas que foram feitas.
- markdown: Responsável por guardar o conteúdo dos resumos das disciplinas.

[12] Disponível em: https://insomnia.rest

Como pode-se observar a tabela de usuário possui um relacionamento de um para muitos com os demais através de sua chave estrangeira com a chave primária.

Figura 38 - Diagrama de banco de dados do StudyNizer



Fonte: Elaboração do autor, 2022

5 Tabela comparativa das funcionalidades das aplicações

A seguir é apresentada uma tabela comparativa entre as aplicações exploradas onde o principal objetivo é analisar a viabilidade destas ferramentas serem pensadas especificamente para um estudante, englobando critérios como UX e UI, facilidade de uso e funcionalidades que façam sentido para este tipo de usuário.

Tabela 3 - Tabela comparativa entre as plataformas

Aplicação	Fácil de usar para um único usuário?	É pensado para utilização de um estudante e possui funcionalidades relacionadas com esta área?	Possui uma boa UX e UI?	Permite fazer anotações?	Possui quadro Kanban?	Possui Calendário?
Trello	Sim	Sim	Não	Não	Sim	Sim
Jira	Não	Não	Não	Sim	Sim	Sim
Notion	Sim	Sim	Sim	Sim	Sim	Sim
ClickUp	Não	Não	Não	Sim	Sim	Sim
Monday.com	Não	Não	Não	Sim	Sim	Sim

Kanban Tool	Sim	Sim	Sim	Não	Sim	Sim
Asana	Não	Não	Não	Sim	Sim	Sim
StudyNizer	Sim	Sim	Sim	Sim	Sim	Sim

Fonte: Elaboração do autor, 2022

Analisando a tabela acima percebe-se que nenhuma das aplicações preenche todos os requisitos, já o software deste trabalho possui todas essas condições, se enquadrando desta forma em uma ferramenta pensada exclusivamente para um estudante dando assim o suporte necessário para uma boa experiência durante sua vida acadêmica.

6 Conclusão

O intuito do presente trabalho foi mostrar que hoje no mercado existem diversas aplicações para organização pessoal com múltiplas funcionalidades, porém a maioria delas é pensada em grupos de trabalho, não sendo voltadas para apenas um único usuário, desta forma a dificuldade de utilização desses softwares atreladas a problemas de UI e UX faz com que o estudante possa ter dificuldade ao utilizar o sistema. Este estudo abordou a necessidade de juntar funcionalidades de kanban, anotações e calendário em uma única plataforma. Atualmente, existem plataformas excelentes para cada uma dessas funcionalidades, mas elas geralmente são oferecidas de forma isolada, o que requer que os usuários tenham várias contas diferentes e não tenham acesso a todas essas utilidades em um só lugar. A junção dessas funcionalidades em uma única plataforma permite com que os usuários tenham tudo o que precisam em um único local, aumentando a praticidade e a eficiência.

O protótipo desenvolvido pôde mostrar que uma plataforma que possui uma interface minimalista e com funcionalidades focadas para a vida estudantil, pode ser muito útil durante a vida acadêmica daqueles que porventura irão utilizá-la. As tecnologias também aplicadas fazem com que a aplicação seja performática, dando uma melhor experiência para os usuários.

Por fim, espera-se que este trabalho tenha conseguido dar uma ideia de como um software de organização deve ser aplicado às necessidades específicas de um estudante e assim fazer com que estes obtenham um melhor resultado ao longo de seus cursos.

Para trabalhos futuros algumas funcionalidades podem ser adicionadas:

- Integração com o moodle para que os alunos consigam importar as disciplinas que estão fazendo no momento.
- Desenvolvimento de uma aplicação móvel para dispositivos android e IOS.
- Acrescentar mais funcionalidades na plataforma como sistema de notificações, pomodoro, chat por texto e vídeo conferência.
- Ser possível adicionar colegas.
- Fóruns de discussões para disciplinas.

REFERÊNCIAS

BASSO, Cláudia *et al*. **Organização de tempo e métodos de estudo: Oficinas com estudantes universitários**.
Disponível em: http://pepsic.bvsalud.org/pdf/rbop/v14n2/12.pdf. Acesso em: 22 jul. 2022

COCKBURN, Alistair. **Agile software development: The people factor**. Disponível em: https://www.researchgate.net/publication/2955526_Agile_software_development_The_people_factor. Acesso em: 20 jul. 2022.

CEARÁ. SECRETARIA DE EDUCAÇÃO DO ESTADO DO CEARÁ. (ed.). **Gestão Organizacional**: Curso Técnico em administração. Curso Técnico em Administração. 2015. Escola Estadual de Educação Profissional. Disponível em: https://educacaoprofissional.seduc.ce.gov.br/images/material_didatico/administracao/administracao_gestao_organizacional.pdf. Acesso em: 22 jul. 2022

PERNAMBUCO, Ufpe: Universidade Federal de. **Rational Unified Process**. 2005. Disponível em: https://www.cin.ufpe.br/~gta/rup-vc/core.base_concepts/guidances/concepts/task_86A4917F.html. Acesso em: 22 jul. 2022.

MACCANN, Carolyn. **Strategies for success in education**: time management is more important for part-time than full-time community college students. Time management is more important for part-time than full-time community college students. 2010. Disponível em: https://www.sciencedirect.com/science/article/pii/S1041608011001786. Acesso em: 08 jul. 2022.

WAKODE, Rajat B.. **Overview on Kanban Methodology and its Implementation**. 2015. Disponível em: https://www.academia.edu/27885179/Overview_on_Kanban_Methodology_and_its_Implementation. Acesso em: 20 jul. 2022.

ALSAQQA, Samar. **Agile Software Development: Methodologies and Trends**. 2020. Disponível em: https://www.researchgate.net/profile/Samar-Alsaqqa/publication/342848746_Agile_Software_Development_Methodologies_and_Trends/links/5f09bcdfa6fdcc4ca45e36f0/Agile-Software-Development-Methodologies-and-Trends.pdf. Acesso em: 20 jul. 2022.

PEREIRA, Paulo. **Entendendo Scrum para Gerenciar Projetos de Forma Ágil**. 2007. Disponível em: https://www.academia.edu/28954250/Entendendo_Scrum_para_Gerenciar_Projetos_de_Forma_%C3%81gil. Acesso em: 22 jul. 2022.

ARBULU, Roberto. **KANBAN IN CONSTRUCTION**. 2003. Disponível em: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.875.4062&rep=rep1&type=pdf. Acesso em: 20 jul. 2022.

CORONA, Erika. **A Review of Lean-Kanban Approaches in the Software Development** ERIKA CORONA e FILIPPO EROS PANI. 2013. Disponível em: http://www.wseas.us/journal/pdf/information/2013/5709-110.pdf. Acesso em: 20 jul. 2022.

TELES, Vinícius Manhães. **UM ESTUDO DE CASO DA ADOÇÃO DAS PRÁTICAS E VALORES DO EXTREME PROGRAMMING**. 2005. Disponível em: https://www.ime.usp.br/~ale/Dissertacao.pdf. Acesso em: 22 jul. 2022.

NEUMANN, Andy. **Analysis of Public REST Web Service APIs.** 2018**.** Disponível em: https://ieeexplore.ieee.org/abstract/document/8385157. Acesso 20 nov. 2022.

BROOKE, John. **SUS: A Retrospective**. 2013. Disponível em: https://uxpajournal.org/sus-a-retrospective/ . Acesso 20 nov. 2022.

State of Agile Report**. State of Agile**. 2022. Disponível em: https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf. Acesso em: 22 dez. 2022.

GLITCH. **Trello**. 2011. Disponível em: https://trello.com/. Acesso em: 22 jul. 2022.

ATLASSIAN. **Jira**. 2004. Disponível em: https://www.atlassian.com/br/software/jira. Acesso em: 22 jul. 2022.

INC., Notion Labs. **Notion**. 2016. Disponível em: https://www.notion.so/. Acesso em: 22 jul. 2022.

Mango Technologies, Inc. **ClickUp.** 2017. Disponível em: https://clickup.com/. Acesso em: 15 nov. 2022.

monday.com Ltd**. Monday.com.** 2012. Disponível em: https://www.monday.com. Acesso em: 15 nov. 2022.

Shore Labs. **Kanban Tool.** 2012. Disponível em: https://kanbantool.com. Acesso em: 15 nov. 2022.

Asana, Inc. **Asana.** 2008. Disponível em: https://kanbantool.com. Acesso em: 15 nov. 2022.

WALKE, Jordan. **React**. 2013. Disponível em: https://reactjs.org/. Acesso em: 22 jul. 2022.

Vercel Inc. **Styled Components**. 2015. Disponível em: https://reactjs.org/. Acesso em: 20 dez. 2022.

ABRAMOV, Dan. **Redux**. 2015. Disponível em: https://styled-components.com/. Acesso em: 22 jul. 2022.

DAHL, Ryan. **NodeJs**. 2009. Disponível em: https://nodejs.org/en/. Acesso em: 22 jul. 2022.

STONEBRAKER, Michael. **PostgresSQL**. 1986. Disponível em: https://www.postgresql.org/. Acesso em: 22 jul. 2022.

KONG, Inc. **insomnia.** 2017**.** Disponível em: https://insomnia.rest/. Acesso em: 22 jul. 2022.

APÊNDICE B – CÓDIGO FONTE

Os códigos fontes deste trabalhos estão disponíveis em um repositório do github. É possível realizar o download dos códigos através das seguintes URLs abaixo:

Front-end: https://github.com/marcelotc/StudyNizer-frontend
Back-end: https://github.com/marcelotc/StudyNizer-backend