CronoClick: Sua solução Web para gerenciamento de horários

Dilvonei Alves Lacerda Junior[1], **Gabriel de Paula Meira**[1]

[1]Universidade Federal de São João Del Rei (UFSJ)
São João del-Rei – MG – Brasil

{dilvoneialveslacerdajunior, gabriel.meira.2004}@aluno.ufsj.edu.br

Abstract. *We will explore a time management solution through a software designed to register and organize activities at specific times. Faced with the common need to balance multiple responsibilities and avoid schedule conflicts, the development of this software was driven by the demand for a tool that would simplify the scheduling process and provide a clear view of future commitments.*

Resumo. *Exploraremos uma solução de gerenciamento de horários por meio de um software projetado para cadastrar e organizar atividades em horários específicos. Diante da necessidade comum de equilibrar múltiplas responsabilidades e evitar conflitos de horários, o desenvolvimento desse software foi impulsionado pela demanda por uma ferramenta que simplificasse o processo de agendamento e oferecesse uma visão clara dos compromissos futuros.*

1. Visão Geral

Inicialmente, é importante ressaltar o papel fundamental dos softwares na atualidade. A exemplo dos variados programas e aplicativos que executam tarefas específicas em dispositivos eletrônicos, tornando diversas atividades mais eficientes e acessíveis.

Uma problemática muito comum do cotidiano das pessoas é a necessidade de realizar o planejamento e gerenciamento dos afazeres, que muitas vezes passa pela falta de uma ferramenta específica de fácil utilização que propicie todas as funcionalidades em um único ambiente. Partindo do conhecimento desses obstáculos, o software em questão tem como missão proporcionar aos usuários, com foco em estudantes, uma maneira intuitiva e prática de gerenciar seu tempo, permitindo a otimização da agenda e o aumento da produtividade.

Com a definição clara da funcionalidade e propósito do sistema, é essencial detalhar esses requisitos de forma precisa e completa. Isso garantirá que o sistema desenvolvido atenda às expectativas dos usuários e partes interessadas, fornecendo uma solução eficaz e de alta qualidade.

2. Requisitos

Os requisitos são especificações detalhadas que descrevem as funcionalidades, características e restrições que um sistema deve ter para atender às necessidades dos usuários e das partes interessadas.

Ter requisitos bem definidos é crucial para o sucesso do projeto de software, já que para alcançar um resultado satisfatório é necessário primeiramente especificar o que está

sendo esperado. Essas especificações detalhadas servem como guia para todo o desenvolvimento, testes e implementação do sistema, fornecendo uma base sólida com métricas para avaliar o progresso e a qualidade do trabalho realizado.

Os requisitos foram definidos primeiramente com base em histórias de usuário e depois transformados em casos de uso, ambos detalhados a seguir. Essas abordagens fornecem uma estrutura detalhada para a concepção das funcionalidades, interações do usuário e cenários de uso esperados.

2.1. Histórias

As histórias de usuário são uma abordagem essencial para especificar requisitos em projetos de desenvolvimento de software, particularmente dentro de metodologias ágeis como o Scrum e o Extreme Programming (XP). Elas são escritas em uma linguagem simples e acessível, seguindo a estrutura:

```
"Como um [tipo de usuário], eu gostaria de [realizar uma ação]"
```

Essa simplicidade facilita a compreensão por parte de todas as partes interessadas e permite uma priorização eficaz com base no valor que agregam ao usuário e aos objetivos do projeto. Além disso, as histórias promovem *feedback* contínuo do cliente durante o desenvolvimento do software, possibilitando ajustes e melhorias ao longo do processo. Em suma, as histórias de usuário são uma ferramenta valiosa para capturar e comunicar os requisitos de forma ágil, propiciando a colaboração entre equipe e cliente.

2.1.1. Histórias de um usuário não-autenticado

Os usuários foram divididos em duas classes: autenticado e não-autenticado. Optou-se por não permitir que qualquer usuário pudesse interagir com as funcionalidades principais do sistema caso não tenha feito a autenticação, obrigando a realização do cadastro no primeiro acesso e o login nos demais. Dessa forma, apenas duas histórias estão vinculadas a essa classe, sendo descritas abaixo.

História: Como um usuário não-autenticado, eu gostaria de me cadastrar.

Critérios de Aceitação:

1. O usuário deve informar nome, senha e email.

História: Como um usuário não-autenticado, eu gostaria de fazer login.

Critérios de Aceitação:

1. O formulário para login deve exigir nome e senha.
2. O sistema deve manter o usuário autenticado por um mês, contando a partir de cada novo acesso.

2.1.2. Histórias de um usuário autenticado

Após conceder o acesso para o usuário, este poderá interagir com o sistema para iniciar sua experiência. A primeira funcionalidade está relacionada com o cronograma semanal e suas disciplinas, o qual deve receber as informações e organizá-las em um quadro gráfico.

História: Como um usuário autenticado, eu gostaria de criar um cronograma semanal com as disciplinas, definindo nome, professor, e horários em que ocorrem.

Critérios de Aceitação:

1. O formulário deve indicar quais campos são obrigatórios.

História: Como um usuário autenticado, eu gostaria de visualizar minha semana em um quadro de horários contendo as disciplinas posicionadas nos seus horários.

Critérios de Aceitação:

1. O quadro de horários deve se ajustar dinamicamente aos horários inseridos.
2. O usuário deve poder decidir mostrar ou ocultar os dias do fim de semana.
3. Cada ocorrência de uma disciplina no cronograma deve mostrar seu nome e nome do professor.

Também deve ser possível o registro de tarefas, eventos pontuais que ocorrem em data e hora específicas, estando vinculadas a uma disciplina.

História: Como um usuário autenticado, eu gostaria de criar uma tarefa, definindo nome e data de entrega, além de poder adicionar anotações a qualquer momento.

Critérios de Aceitação:

1. O formulário deve indicar quais campos são obrigatórios.
2. As tarefas adicionadas devem ser adicionadas em uma lista de fácil visuação.

Ademais, outras funcionalidades se fazem presentes para agregar valor ao software, como portabilidade, disponibilidade e segurança das informações.

História: Como um usuário autenticado, eu gostaria de salvar as alterações no meu cronograma e poder visualizá-las em qualquer lugar.

Critérios de Aceitação:

1. Os dados do usuário devem ser salvos em um banco de dados.
2. Deve ser indicado quando há alterações não salvas, alertando o usuário caso tente fechar a página.
3. As interfaces devem ser responsivas, adaptando a qualquer dispositivo.

História: Como um usuário autenticado, eu gostaria de exportar um cronograma e tê-lo como arquivo.

Critérios de Aceitação:

1. Não deve haver perda das informações do cronograma.
2. O formato do arquivo deve ser conveniente para impressão.

2.2. Casos de uso

Casos de uso são representações detalhadas de interações entre um sistema e seus usuários para alcançar um objetivo específico. Eles descrevem como os usuários interagem com o sistema em diferentes cenários e fornecem uma visão clara das funcionalidades do software. A importância dos casos de uso reside na sua capacidade de capturar requisitos de forma precisa e compreensível, fornecendo uma base sólida para o design, desenvolvimento e teste do sistema.

A Figura 1 traz o diagrama de casos de uso projetado para o software desenvolvido, sendo detalhado posteriormente.

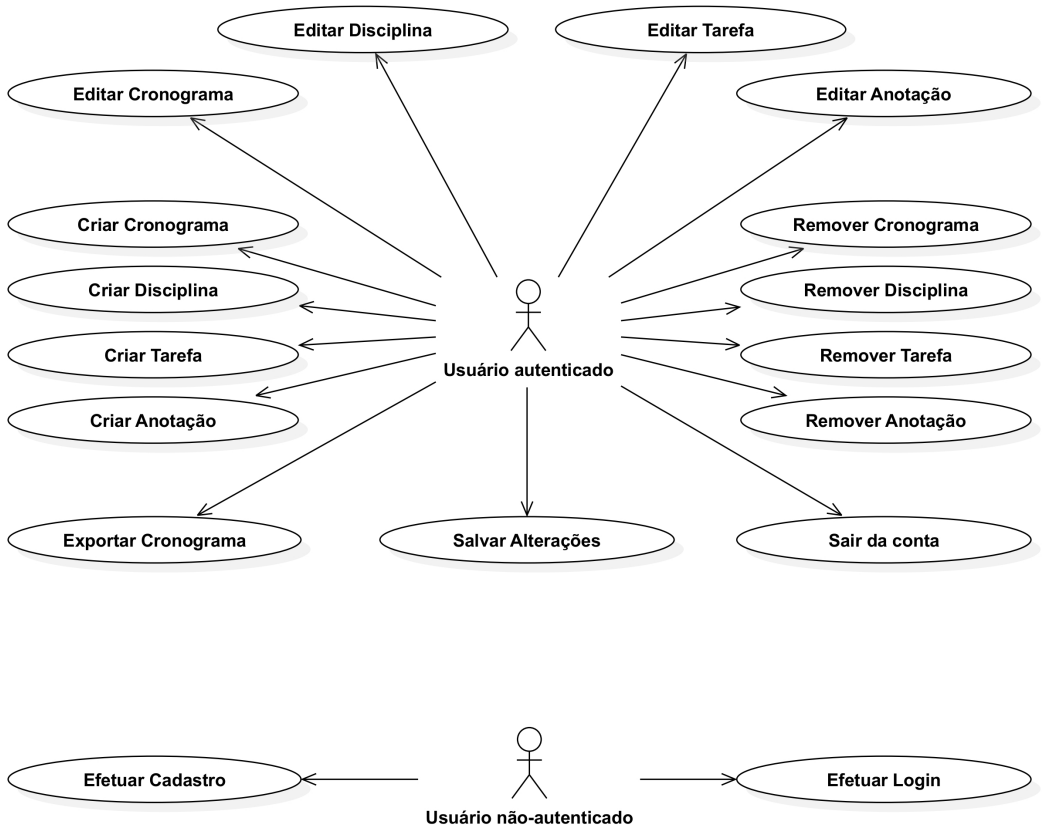

Figura 1. Diagrama de Casos de Uso

2.2.1. Casos de uso de um usuário não-autenticado

A definição dos casos de uso para usuários que não estão autenticados torna-se simples devido ao fato da utilização das funcionalidades do sistema ser limitada somente a usuários autenticados, portanto, existem apenas dois casos: Cadastro e Login.

Nome: Efetuar Cadastro
Ator: Usuário não-autenticado
Fluxo normal:
1. Usuário insere email, nome e senha.
2. Sistema valida os dados, evitando campos vazios.
3. Sistema cadastra o usuário na base de dados.
4. Sistema redireciona Usuário para a aplicação.

Extensões:
2. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Efetuar Login
Ator: Usuário não-autenticado
Fluxo normal:
1. Usuário insere nome e senha.
2. Sistema valida as credenciais.
3. Sistema salva o token com tempo de expiração de um mês na máquina do usuário para autenticações futuras.
4. Sistema redireciona Usuário para a aplicação.

Extensões:
2. Credenciais incorretas, Sistema solicita ao Usuário que tente novamente.

2.2.2. Casos de uso de um usuário autenticado

No diagrama da Figura 1, destaca-se a existência de um padrão nos casos de uso para usuários autenticados. Cada entidade do sistema que integra as funcionalidades principais - cronograma, disciplina, tarefa e anotação - possui três casos: Criar, Editar e Remover.

Nome: Criar Cronograma
Ator: Usuário autenticado
Fluxo normal:
1. Usuário insere nome da cronograma.
2. Sistema valida os dados, evitando que o nome seja vazio.
3. Sistema adiciona o novo cronograma na listagem.
4. Usuário é redirecionado para o novo cronograma vazio.

Extensões:
2. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Criar Disciplina
Ator: Usuário autenticado
Fluxo normal:

1. Usuário insere nome da disciplina, professor e horários que ocorre.
2. Sistema valida os dados, evitando campos obrigatórios vazios.
3. Sistema adiciona a nova disciplina no cronograma.
4. A disciplina é exibida no cronograma para o Usuário.

Extensões:

2. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Criar Tarefa
Ator: Usuário autenticado
Fluxo normal:

1. Usuário insere nome da tarefa, data de entrega e descrição.
2. Sistema valida os dados, evitando campos obrigatórios vazios.
3. Sistema adiciona a nova tarefa à lista.
4. A tarefa é exibida na lista para o Usuário.

Extensões:

2. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Criar Anotação
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe tarefa na lista.
2. Usuário insere o conteúdo e categoria da anotação.
3. Sistema valida os dados, evitando campos obrigatórios vazios.
4. Sistema adiciona a nova anotação à tarefa.
5. A tarefa é exibida com a nova anotação para o Usuário.

Extensões:

3. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Editar Cronograma
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe o cronograma na listagem.
2. Usuário insere o novo nome para o cronograma.
3. Sistema valida os dados, evitando que o nome seja vazio.
4. Sistema atualiza o nome do cronograma na listagem.
5. O cronograma é exibido ao Usuário com o novo nome.

Extensões:

3. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Editar Disciplina
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a disciplina no cronograma.
2. Usuário altera os dados já definidos previamente.
3. Sistema valida os dados, evitando campos obrigatórios vazios.
4. Sistema atualiza a disciplina no cronograma.
5. A disciplina é exibida ao Usuário com as alterações.

Extensões:

3. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Editar Tarefa
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a tarefa na lista.
2. Usuário altera os dados já definidos previamente.
3. Sistema valida os dados, evitando campos obrigatórios vazios.
4. Sistema atualiza a tarefa na lista.
5. A tarefa é exibida ao Usuário com as alterações.

Extensões:

3. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Editar Anotação
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a anotação na tarefa.
2. Usuário altera os dados já definidos previamente.
3. Sistema valida os dados, evitando campos obrigatórios vazios.
4. Sistema atualiza a anotação na tarefa.
5. A anotação é exibida ao Usuário com as alterações.

Extensões:

3. Dados inválidos, Sistema solicita ao usuário que conserte os erros.

Nome: Remover Cronograma
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe o cronograma na listagem.
2. Usuário seleciona a opção de deletar o cronograma.
3. Sistema atualiza a listagem de cronogramas.
4. A lista de cronogramas é exibida ao Usuário com as alterações.

Nome: Remover Disciplina
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a disciplina no cronograma.
2. Usuário seleciona a opção de deletar a disciplina.
3. Sistema atualiza o cronograma.
4. O cronograma é exibido ao Usuário com as alterações.

Nome: Remover Tarefa
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a tarefa na lista.
2. Usuário seleciona a opção de deletar a tarefa.
3. Sistema atualiza a lista.
4. A lista é exibida ao Usuário com as alterações.

Nome: Remover Anotação
Ator: Usuário autenticado
Fluxo normal:

1. Usuário escolhe a anotação na tarefa.
2. Usuário seleciona a opção de deletar a anotação.
3. Sistema atualiza a tarefa.
4. A tarefa é exibida ao Usuário com as alterações.

Uma funcionalidade casual de grande valor para o usuário é exportar seu cronograma, fazendo com que seja gerado um arquivo contendo todas as informações para auxiliar seu planejamento. Esse arquivo pode ser impresso e usado da forma que for conveniente, como imprimir uma versão física.

Nome: Exportar Cronograma
Ator: Usuário autenticado
Fluxo normal:

1. Usuário seleciona a opção de exportar o cronograma.
2. Sistema gera um arquivo gráfico contendo o cronograma completo.
3. Sistema salva o arquivo no computador do Usuário.

Todas as alterações nas entidades não devem ser processadas no exato momento em que são solicitadas, sendo estas salvas apenas quando o usuário solicitar tal ação. Essa medida faz com que menos recursos sejam gastos, principalmente em operações no banco de dados, melhorando a disponibilidade e eficiência do sistema.

Nome: Salvar Alterações
Ator: Usuário autenticado
Fluxo normal:
1. Usuário seleciona a opção de salvar as alterações.
2. Sistema salva as alterações na base de dados.
3. Sistema avisa que as alterações foram salvas com sucesso.

Extensões:
2. Falha na conexão com o banco de dados, tentar novamente em instantes.

Da mesma forma que o usuário realiza a autenticação para acessar as funcionalidades, faz-se necessário uma opção de sair da conta, que simplesmente retorna para a tela principal do sistema, revogando seu acesso.

Nome: Sair da conta
Ator: Usuário autenticado
Fluxo normal:
1. Usuário seleciona a opção de sair da conta.
2. Sistema redireciona o Usuário para a autenticação, removendo seu acesso.

3. Arquitetura

Como visto anteriormente, se faz necessária a existência de um banco de dados para persistência dos dados dos usuários, o que, por motivos de segurança, implica no uso de uma arquitetura capaz de comunicar os clientes (usuários), com um servidor responsável pela comunicação com o banco, assim como mostra a Figura 2.

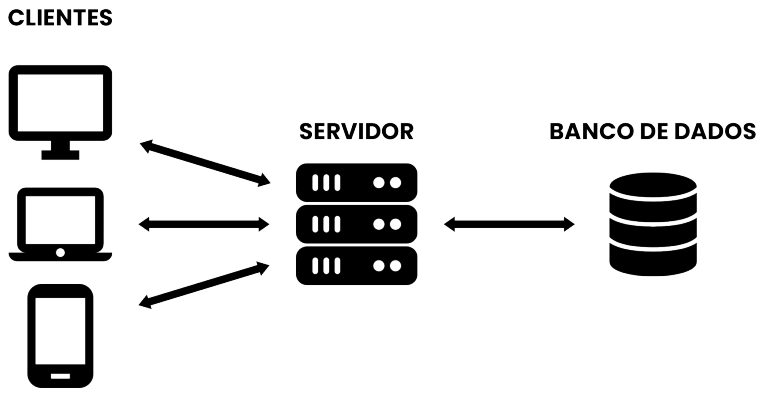

Figura 2. Esquema de comunicação da aplicação

A estratégia proposta se encaixa no padrão de arquitetura **Cliente-Servidor**. Essa arquitetura facilita a distribuição de tarefas e recursos em uma rede, onde o servidor centraliza e gerencia informações, enquanto os clientes requisitam e recebem dados ou serviços por meio de requisições HTTP. Essa abordagem permite a escalabilidade, flexibilidade e compartilhamento eficiente de recursos em diversos contextos, especialmente em aplicações Web, onde a portabilidade é maior.

Para o desenvolvimento desse sistema, optou-se por utilizar o framework **Next.js** devido à sua capacidade de facilitar o desenvolvimento de aplicações web modernas e eficientes. O Next.js é uma estrutura de desenvolvimento **React** que oferece funcionalidades como renderização do lado do servidor (SSR), pré-renderização estática e suporte a APIs, proporcionando um ambiente ideal para a arquitetura escolhida.

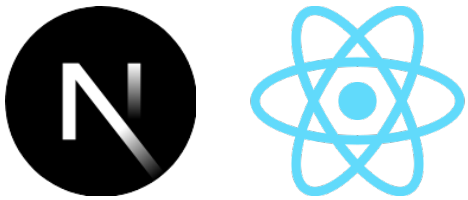

Figura 3. Next.js e React

3.1. Banco de dados

Uma estrutura de banco de dados bem projetada é fundamental para assegurar a eficácia das operações de consulta. Nesse sentido, optou-se por uma abordagem não-relacional para o banco de dados, que simplifica significativamente o armazenamento das entidades, permitindo a utilização da Notação de Objetos JavaScript (JSON). Essa escolha se mostra vantajosa pela flexibilidade oferecida, facilitando a manipulação e o acesso aos dados de forma mais intuitiva pelo lado do cliente.

O diagrama de classes que representa a forma que as entidades foram organizadas pode ser visualizado na Figura 4.

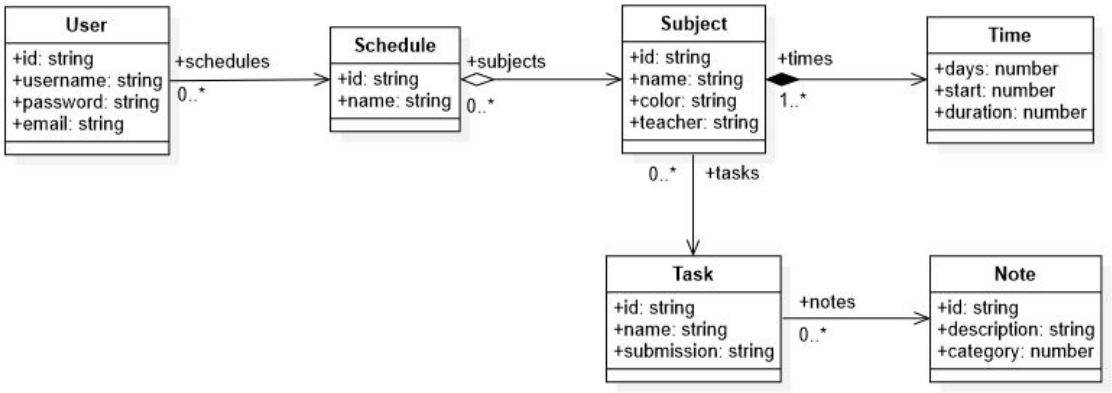

Figura 4. Diagrama de Classes das entidades

4. Testes

Testes no desenvolvimento de software são avaliações sistemáticas realizadas em sistemas, aplicativos ou componentes para verificar se estão em conformidade com os requisitos e se funcionam corretamente. Eles desempenham um papel crucial na garantia da qualidade do software, detectando defeitos e permitindo correções econômicas. Além disso, os testes fornecem documentação do comportamento esperado do software, facilitando sua manutenção e evolução ao longo do tempo, garantindo que o sistema atenda às expectativas dos usuários com eficácia e confiabilidade.

Foram formulados três casos de teste para validar e verificar algumas funcionalidades do sistema, entre elas o cadastro do usuário, uma etapa fundamental do fluxo de acesso à plataforma.

Funcionalidade: Cadastro do Usuário
Comportamento Esperado:

1. Ao entrar com o nome de usuário, e-mail, senha e confirmação de senha, o usuário deverá ser redirecionado para a tela de login.
2. Campos preenchidos erroneamente devem ser indicados para correção.

Verificações:

- Todos os campos devem ser obrigatórios.
- Tamanho mínimo da senha (8 caracteres).
- E-mail em formato válido.
- Validação da confirmação de senha.
- Verificar se o nome e/ou e-mail já estão em uso.
- Exibição de mensagem de erro para todos os campos.
- Redirecionamento para a tela de login.

Critérios de Aceitação:

- Opção para mostrar/ocultar a senha.
- Opção para redirecionar para a tela de login caso o usuário já possua conta.

Também se faz importante o teste da adição das disciplinas ao cronograma, tendo em vista que é o ponto de maior interesse ao usuário, portanto, garantir sua boa performance é extremamente necessário.

Funcionalidade: Adicionar disciplina ao cronograma
Comportamento Esperado:

1. Usuário digita o nome da disciplina, professor, cor e pelo menos uma ocorrência (dias da semana, horário de início e tempo de duração).
2. Usuário confirma a criação
3. Disciplina aparece no cronograma.

Verificações:

- Todos os campos devem ser obrigatórios, exceto o de professor.
- Deve haver no mínimo uma ocorrência adicionada.
- Disciplina deve aparecer no cronograma com todos os dados inseridos e na cor informada.

Critérios de Aceitação:

- Posições e tamanhos da disciplina no cronograma devem seguir corretamente os dias, o tempo de início e a duração de todas as suas ocorrências.
- O cronograma deve ser responsivo em todos os tamanhos de tela.
- O formulário deve pertencer à mesma página do cronograma, evitando redirecionamentos desnecessários.
- Opção para cancelar o preenchimento do formulário.

É indispensável a opção de edição dos dados preenchidos pelo usuário no momento do cronograma, fazendo com que sua experiência ao utilizar a plataforma seja mais fluida, o deixando menos apreensivo em cometer erros e ter de refazer todo o processo para adicionar a disciplina novamente.

Funcionalidade: Editar disciplina do cronograma
Comportamento Esperado:

1. Usuário seleciona a disciplina que deseja editar.
2. Usuário modifica os dados da disciplina.
3. Usuário confirma salva as edições.
4. Sistema atualiza o cronograma.

Verificações:

- Todos os campos devem ser verificados da mesma forma que na adição da disciplina.
- Assegurar que não está sendo criada uma nova entidade ao invés de editar uma pré-existente.
- O cronograma deve ser atualizado após a confirmação da edição.

Critérios de Aceitação:

- Mesmos critérios de aceitação da adição de uma disciplina.
- Opção para apagar a disciplina no formulário.

Por fim, é essencial assegurar que o salvamento dos dados do usuário ocorra sem qualquer tipo de problema, por se tratar de um recurso que deve oferecer disponibilidade e segurança das informações.

Funcionalidade: Salvamento das alterações realizadas
Comportamento Esperado:

1. Usuário seleciona a opção de salvar alterações.
2. Sistema salva os dados no banco de dados.
3. Sistema notifica o sucesso da operação.

Verificações:

- Emitir alerta impedindo o usuário de fechar instantaneamente a aplicação havendo alterações não salvas.
- Assegurar que a versão do cliente seja a mesma salva no sistema após a operação.

Critérios de Aceitação:

- Não acumular ”lixo”no banco de dados, em caso de remoção de entidades.
- Em caso de falta de conexão, abortar salvamento dos dados e alertar o usuário.

Obviamente, diversos outros casos de testes devem fazer parte da etapa de desenvolvimento do software. A ampla cobertura de testes é essencial, pois quanto maior a cobertura, menor a chance de ocorrerem imprevistos que podem levar a prejuízos tanto para os usuários quanto para os desenvolvedores.

5. Prototipação

A etapa de prototipação tem como objetivo criar uma representação inicial que demonstra o progresso alcançado até um determinado estágio. Este processo não apenas fornece uma visão tangível do produto em desenvolvimento, mas também serve como uma ferramenta valiosa para avaliar e validar conceitos, identificar requisitos ausentes e/ou não satisfativos e iterar sobre o design com o apoio das partes interessadas.

Os protótipos exibidos nas Figuras 5 e 6 mostram como as interfaces do formulário da adição de uma disciplina e o cronograma, respectivamente, serão desenvolvidas.

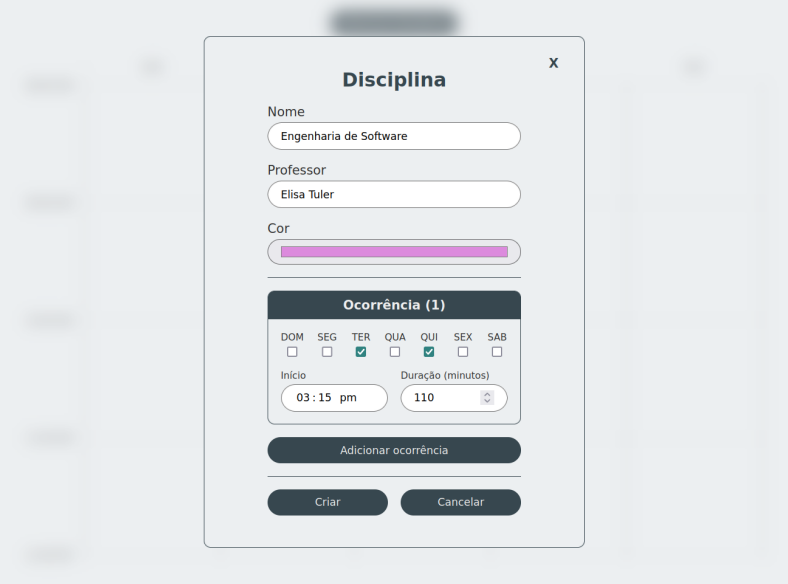

Figura 5. Protótipo funcional do formulário

Figura 6. Protótipo funcional do cronograma

6. Considerações Finais

Este projeto de desenvolvimento de software para gerenciamento de horários oferece uma solução abrangente e eficaz para usuários que buscam organizar suas agendas de forma intuitiva e prática. Ao oferecer uma maneira simplificada e eficiente de gerenciar horários, o software tem o potencial de impactar positivamente a rotina do seu usuário, facilitando a organização da rotina diária de estudantes e profissionais. Além disso, ao aumentar a produtividade e reduzir conflitos de horários, o software pode contribuir para o bem-estar e o equilíbrio entre vida pessoal e profissional dos usuários, promovendo uma maior qualidade de vida e eficiência em suas atividades cotidianas.

Com uma arquitetura Cliente-Servidor e a utilização do framework Next.js, aliados a um banco de dados não-relacional, o software promete proporcionar uma experiência fluida e responsiva. A estratégia de prototipação adotada garante não apenas um desenvolvimento progressivo, mas também a contribuição das partes interessadas, o que pode evitar conflitos de requisitos mal-entendidos.

Em todas as áreas do desenvolvimento de software, garantir qualidade nos processos de idealização, definição de requisitos, testes e prototipações é fundamental para o sucesso do projeto. Essas etapas não apenas estabelecem bases sólidas para a construção do produto, mas também influenciam diretamente sua viabilidade, desempenho e aceitação pelo usuário final, ressaltando a importância de documentar um software.

Referências

Sampaio, G. (2021). Introdução a arquitetura cliente-servidor. *Medium*.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Education, 9th edition.

Valente, M. T. (2020). *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Editora: Independente.