UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS ANTHONY FERREIRA DE OLIVEIRA

UM SISTEMA PARA GERENCIAMENTO DE EMPRESAS JUNIORES

CAMPINA GRANDE - PB

2022

LUCAS ANTHONY FERREIRA DE OLIVEIRA

UM SISTEMA PARA GERENCIAMENTO DE EMPRESAS JUNIORES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : José Antão Beltrão Moura

CAMPINA GRANDE - PB

2022

LUCAS ANTHONY FERREIRA DE OLIVEIRA

UM SISTEMA PARA O GERENCIAMENTO DE EMPRESAS JUNIORES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

José Antão Beltrão Moura

Orientador – UASC/CEEI/UFCG

Carlos Wilson Dantas de Almeida

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de Setembro de 2022.

CAMPINA GRANDE - PB

RESUMO

Empresas juniores são associações civis sem fins lucrativos, que são formadas e geridas por estudantes de um ou mais cursos superiores, ocasionando assim, uma grande rotatividade de membros. Por esse motivo, diversas vezes o gerenciamento das pessoas, projetos e relatórios e a passagem de tais dados para as diretorias posteriores apresentam dificuldades, pois eles se encontram descentralizados em diversas planilhas e documentos. Desse modo, este trabalho tem como objetivos centralizar as informações através de um sistema web, trazendo mais facilidade e praticidade no gerenciamento da empresa, bem como apresentar as estratégias utilizadas e decisões que foram tomadas, tais como elicitação de requisitos, escolhas de tecnologias e plano de testes.

Um sistema para gerenciamento de empresas juniores

RESUMO

Empresas juniores são associações civis sem fins lucrativos, que são formadas e geridas por estudantes de um ou mais cursos superiores, ocasionando assim, uma grande rotatividade de membros. Por esse motivo, diversas vezes o gerenciamento das pessoas, projetos e relatórios e a passagem de tais dados para as diretorias posteriores apresentam dificuldades, pois eles se encontram descentralizados em diversas planilhas e documentos. Desse modo, este trabalho tem como objetivos centralizar as informações através de um sistema web, trazendo mais facilidade e praticidade no gerenciamento da empresa, bem como apresentar as estratégias utilizadas e decisões que foram tomadas, tais como elicitação de requisitos, escolhas de tecnologias e plano de testes.

Palavras chave

Desenvolvimento de software, empresa júnior, gerenciamento, tecnologia, gestão de projetos.

1 INTRODUÇÃO

Empresa júnior[1] é o nome dado a empresas que são gerenciadas e mantidas voluntariamente por estudantes de um ou mais cursos de graduação. A ideia principal desse tipo de associação é levar conhecimento do mercado para dentro da universidade, tais como gerenciamento de projetos, negociação com clientes, desenvolvimento de sistemas (No caso de empresas de TI) e também o envolvimento em projetos que geram impacto positivo na vida da sociedade.

Tendo em vista esse cenário e também minha experiência na CodeX Jr.[2], as empresas juniores muitas vezes têm dificuldades para gerenciar os dados sobre membros, projetos e relatórios, uma vez que a volatilidade de membros é muito alta, o que torna difícil manter os membros que vem posteriormente cientes de como se encontra a empresa. Com isso, torna-se importante a criação de um ambiente que centralize todos os dados para mantê-los de fácil acesso e simule os ambientes adotados nas empresas seniores.

2 SOLUÇÃO

O projeto tem como objetivo criar uma aplicação web para que os diretores e pessoas autorizadas gerenciem dados relevantes da empresa.

Desse modo, os gestores poderão cadastrar os projetos, membros e links que acharem importantes, bem como gerenciar o acesso ao sistema através da criação de novos usuários com suas respectivas senhas de acesso. Assim o sistema possibilitará que os integrantes da EJ dediquem mais tempo em experiências que possibilitam o crescimento profissional e menos em tarefas de gestão que deveriam ser simples.

2.1 Funcionalidades

2.1.1 Elicitação de requisitos [3]

As funcionalidades citadas a seguir foram frutos de um processo de observação e vivência ao longo de dois anos participando da CodeX Jr. (Empresa júnior de Ciência da Computação da UFCG). A dor primária percebida foi que a gestão dos membros não era feita de forma gerenciável a longo prazo pois a planilha utilizada reunia muitas informações, tornando-se massante a sua atualização periódica e até mesmo encontrá-la nos arquivos da empresa. Tendo notado isto, foi realizada uma reunião ao estilo *brainstorming[4]* com os membros, na qual se originaram várias features que viriam a ser interessantes e resolveria gargalos existentes na empresa, sendo elas:

- *Projetos em andamento e seus respectivos times de desenvolvimento*;
- *Gerência de membros através do sistema;*
- *Gestão de links importantes para a empresa;*
- *Perfil dos integrantes da empresa;*
- *Mural de avisos;*
- *Lista de presença em reuniões;*
- *Área de "sobre nós", com informações gerais sobre a empresa;*

2.1.2 Funcionalidades implementadas

Inicialmente, foram selecionados os requisitos considerados mais importantes para a implementação da versão inicial da aplicação, através de conversas com membros atuais e antigos diretores que participaram da CodeX Jr., sendo eles o gerenciamentos dos membros (feature que originou a ideia do sistema), acompanhamento de projetos em andamento, salvamento de links importantes e/ou muito utilizados e por fim o controle de acesso de usuários por meio do presidente da empresa. Definidos os pontos a serem executados, temos as *user stories* que serão executadas na presente versão. *[Figura 1]*

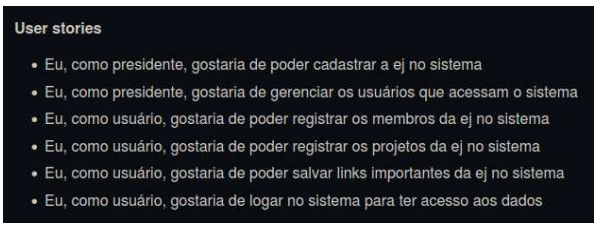


Figura 1: User stories utilizadas como base para as atividades

Tendo isso em mente, a aplicação conta com dois tipos de usuários, o Presidente e os diretores/gestores. O cadastro de ambos é feito de forma distinta, logo, o tipo de usuário é definido no momento do cadastro.

2.1.2.1 Funcionalidades de todos os usuários

Cadastrar membro: Qualquer usuário pode registrar um membro, preenchendo suas respectivas informações. *[Figura 3]*

Gerenciar membros: O usuário pode criar, visualizar, editar ou excluir membros. A criação e edição consistem no preenchimento de um formulário com alguns campos pré-definidos, sendo eles *nome, email, telefone, habilidades* e um campo de observações livre para anotações gerais sobre o membro. A visualização apresenta os dados cadastrados e a exclusão permite a remoção de um membro da empresa.

Gerenciar projetos: O usuário pode criar, visualizar, editar ou excluir projetos. A criação e edição consistem no preenchimento de um formulário com alguns campos pré-definidos, sendo eles *nome, time envolvido, descrição, contato do cliente, data de início, data de término* e um campo de observações livre para anotações gerais sobre andamento do projeto. A visualização apresenta os dados cadastrados e a exclusão permite a remoção de um projeto da empresa.

Gerenciar links importantes: O usuário pode criar, visualizar, editar ou excluir links importantes. A criação e edição consistem no preenchimento de um formulário apenas com campos pré-definidos, sendo eles *nome, url, departamentos interessados e tags*. A visualização apresenta os dados cadastrados, a exclusão permite a remoção de um link e existe um botão de copiar o link juntamente com seu título para compartilhamento em outros locais.

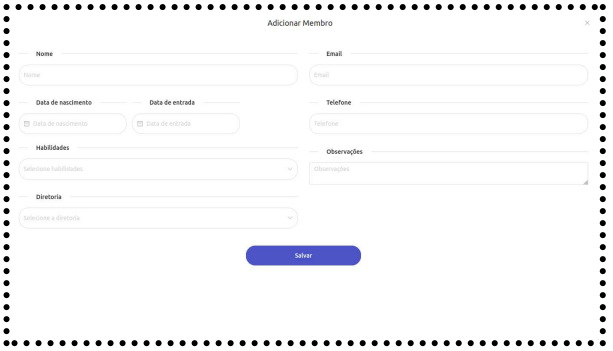


Figura 2: Modal para cadastro de membro

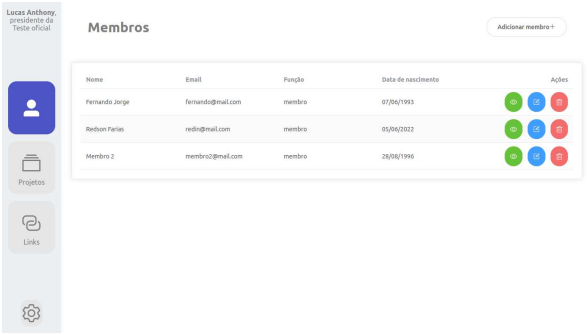


Figura 3: Listagem dos membros

2.1.2.2 Funcionalidades do presidente

Cadastrar usuários: O presidente da empresa pode cadastrar novos usuários ao sistema, que poderão utilizar as telas de gestão porém não podem adicionar novas pessoas.

Gerenciar usuários*:* O presidente pode excluir, editar e alterar senha dos usuários cadastrados. *[Figuras 4 e 5]*

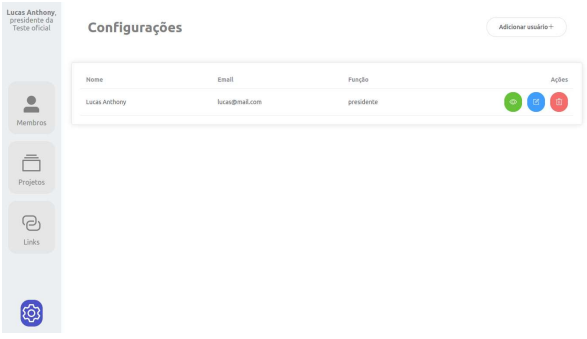


Figura 4: Configuração dos usuários

Figura 5: Edição de usuário

3 ARQUITETURA

A arquitetura escolhida para aplicação neste sistema foi o modelo cliente-servidor, na qual o cliente (frontend) realiza uma requisição ao servidor (backend), que por sua vez retorna os dados solicitados.

O sistema é estruturado em duas partes, sendo elas: O frontend, que é toda a parte visual que será mostrada ao usuário, e o backend, que fica responsável por definir e gerenciar as regras de negócios, bem como persistir e tratar os dados que serão salvos no banco de dados.

3.1 Backend

3.1.1 Tecnologias do Backend

O desenvolvimento do backend se deu utilizando o Node.js [5], uma *runtime javascript* que permite a execução da linguagem fora de um navegador web de forma bem eficiente e simples. A tecnologia foi escolhida pois facilita bastante o desenvolvimento, visto que utiliza uma linguagem de programação não tipada e também executa de forma bem mais rápida e leve que outras tecnologias, como por exemplo o *Spring Boot[6]*.

Para a persistência dos dados, foi escolhido o MongoDB[7], um banco de dados não relacional que armazena informações em documentos, tornando o gerenciamento dos dados mais prático pelo servidor, bem como permitindo alterações pontuais nos esquemas mais facilmente.

3.1.2 Estrutura do Backend

A estruturação dos diretórios do backend foi pensada tendo como base experiências prévias obtidas em projetos realizados, desse modo, foi feita a divisão por módulos e cada um deles possui seus arquivos associados, que são o *model, service, controller e routes.* Além disso, temos as pastas de *config, middlewares e test. [Figura 6]*

Os *controllers* são responsáveis por realizar a lógica de manipulação de dados e aplicar a regra de negócio da aplicação.

Os *models,* como o nome indica, modelam as estruturas que serão guardadas no banco de dados, tendo todos os campos necessários para o funcionamento da lógica de negócio.

Os *services* são responsáveis por toda a regra de negócio da aplicação, bem como o tratamento necessário dos dados, fazendo a conexão dos controllers com os *models*.

Os *middlewares* são responsáveis como interceptadores das requisições, tendo como função principal fazer algum tratamento específico no conteúdo da requisição antes de repassá-la para o *service* alvo, um bom exemplo de utilização é na autorização de usuários, fazendo a filtragem e validação do token JWT[6].

As *routes* são os arquivos que abrigam os endpoints da API, que por sua vez, são como o cliente conseguirá se comunicar com o servidor.

Os *tests agrupam os arquivos de teste do sistema, bem como o arquivo json com os dados fantasia que serão utilizados durante o teste. [Figura 7]*

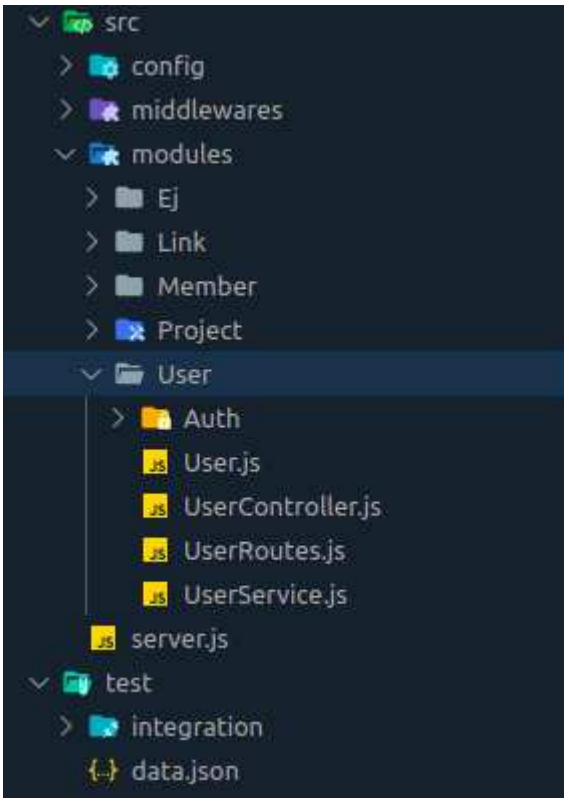


Figura 6: Estrutura do backend

Figura 7: Testes do módulo de usuário

3.2 Frontend

3.2.1 Tecnologias do Frontend

O frontend foi desenvolvido utilizando o Vue.js[8], um framework javascript para construção de interfaces web. A tecnologia proporciona um desenvolvimento ágil e sem complicações, se mostrando mais simples que outras alternativas, tal como o ReactJs. A escolha foi feita por boas experiências com a mesma em projetos anteriores. Em paralelo ao *Vue[4]* foi utilizado o *Pug [9][Figura 8]*, um pré processador de HTML, que permite a escrita de um código mais legível e simples de ser mantido e atualizado.

Para a comunicação com o servidor foi utilizada a biblioteca *Axios*, um cliente HTTP capaz de preparar e enviar as requisições e processar as respectivas respostas.

3.2.2 Estrutura do Frontend

O frontend foi dividido em *components, constants, pages, router, store* e *utils. [Figura 9]*

Os *components* agrupam os componentes que serão utilizados nas *pages*, eles tem a função de agrupar códigos que serão utilizados em várias partes do sistema.

As *pages* reúnem todas as telas presentes na aplicação, que por sua vez serão utilizadas nas *routes*.

As *constants* abrigam constantes que serão utilizadas no sistema.

As *stores* possuem as requisições que serão feitas para o servidor e também os arquivos relativos ao *Vuex*, que é o gerenciador de estados da aplicação.

Os *utils* possuem formatadores e funções auxiliares utilizadas em partes do código.

```
template(
  #footer
)
  span.dialog-footer
    el-button(
      v-if="!isVisualizar"
      @click="isEditar ? editar() : salvar()"
      type="primary"
      color="#4b53c6"
    ) Salvar
```

Figura 8: Exemplo do Pug

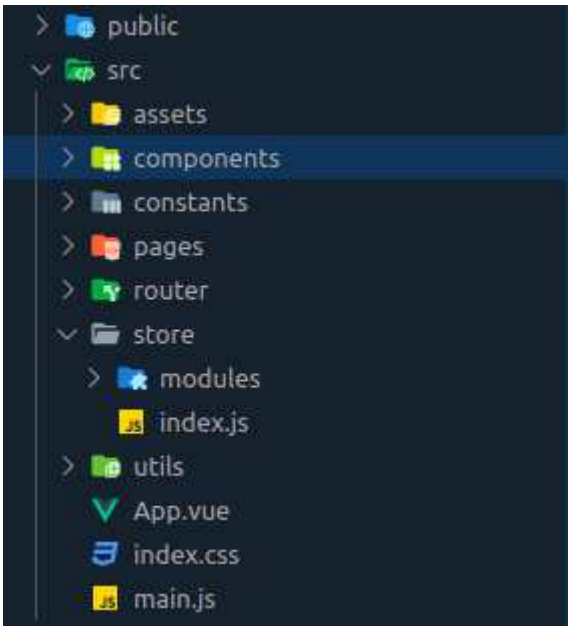


Figura 9: Estrutura do frontend

3.3 Autenticação

Para fazer a autenticação e autorização do sistema no servidor, foi utilizado o JWT (JSON web token)[10], que permite o agrupamento de informações de forma compacta e segura, identificando um usuário específico do sistema.

3.3.1 Fluxo de autenticação

O usuário entra com seu email e senha, que são enviados para o servidor fazer o tratamento, verificar se tais credenciais pertencem a alguém, se sim, é gerado um token JWT com algumas informações não sensíveis do usuário e uma data de expiração, que por sua vez é retornado para o cliente, o qual irá usá-lo em todas as requisições que necessitarem de acesso privilegiado.

4 AVALIAÇÃO

4.1 Deploy

Para disponibilizar a aplicação aos usuários, foi utilizada a plataforma *Heroku[11]*, que permite uma

hospedagem extremamente simples, não necessitando de configurações extras para funcionar e disponibilizar o sistema na nuvem. A escolha se deu por ser uma versão MVP[12] do sistema apenas para testes e validações, e a plataforma em questão é gratuita e possui a estrutura necessária para o uso do sistema sem problemas.

4.2 Validação

Para avaliar o sistema de gerenciamento de empresas juniores foi criado um formulário contendo perguntas seguindo a escala Likert[13], uma metodologia popular em pesquisas de satisfação. ela consiste em opções de resposta em uma escala de pontos com descrições, nesta avaliação foram usados 5 níveis básicos: *Concordo totalmente*, *concordo*, *nem concordo nem discordo*, *discordo* e *discordo totalmente*. Ele foi repassado entre os diretores/gestores atuais da CodeX Jr. para colher informações sobre o uso do sistema pelos mesmos, que se deu durante duas semanas nas quais a aplicação foi utilizada, ao total foram cinco utilizadores de diferentes departamentos da empresa, como presidência, projetos e negócios.

O formulário contou com 6 questionamentos, visando analisar os seguintes pontos o sistema desenvolvido:

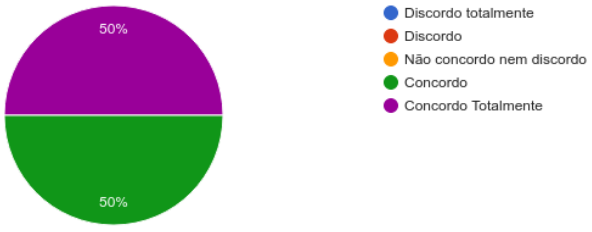
- Utilidade
- Usabilidade
- Comportamento
- Interface
- Desempenho
- Satisfação geral

4.3 Resultados

Os resultados obtidos representam o nível de satisfação dos empresários juniores que testaram o sistema em relação aos pontos citados anteriormente.

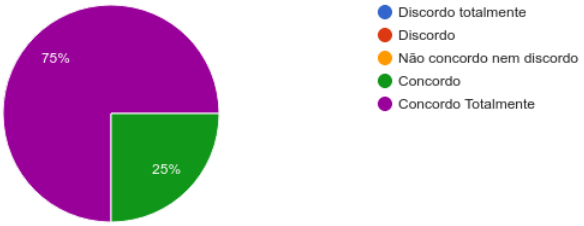
Usabilidade: A representação da facilidade e simplicidade de uso da aplicação pelos usuários alvo.

1 - Foi fácil entender como a aplicação funciona



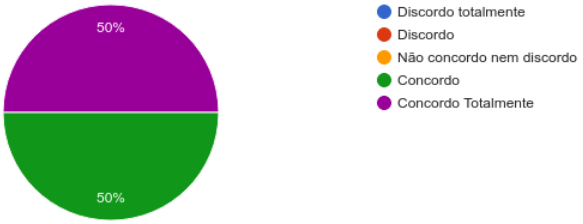
Utilidade: Representação da efetividade da aplicação aplicada ao contexto proposto.

2 - As funcionalidades da aplicação foram úteis no contexto proposto



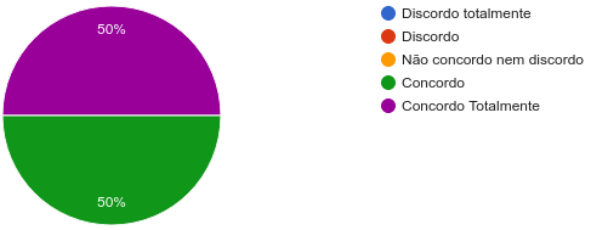
Interface: Representação de elementos visuais utilizados na aplicação e da forma que foram utilizados.

3 - A aplicação possui uma interface agradável



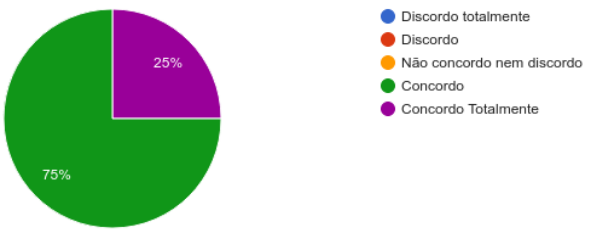
Comportamento: Representação de como a aplicação se comporta aos estímulos dos usuários.

4 - A aplicação se comporta da maneira esperada



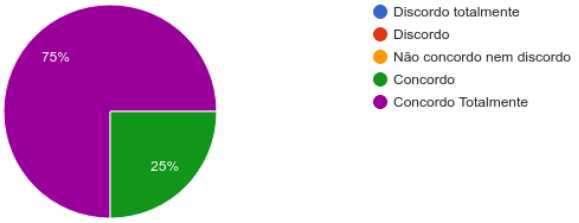
Desempenho: Representação da eficiência do sistema em gerenciar e processar informações.

5 - A aplicação possui um bom desempenho



Satisfação geral: representa o nível de satisfação que a aplicação trouxe para os usuários que a utilizaram.

6 - No geral, gostei da aplicação



4.3.1 Considerações sobre os resultados

A pergunta 3 *A aplicação possui uma interface agradável* obteve um índice mediano de aprovação total (50%), um dos principais motivos se deve a pouca experiência que tenho com design de interfaces, optando assim por um design bem simples e limpo, sem maiores detalhamentos.

A pergunta 5 *A aplicação possui um bom desempenho,* obteve um alto nível (75%) de aprovação parcial, muito se deve aos servidores utilizados para a hospedagem do sistema, que por serem gratuitos oferece recursos limitados, tais como baixa memória e tempo de atividade sem interrupção reduzido.

A pergunta 6 *No geral, gostei da aplicação,* expõe que o apanhado geral do sistema foi muito positivo (75%), não atingindo sua totalidade pelo fato de se tratar de um MVP e apresentar pequenos problemas, como os citados acima, que por sua vez são naturais nesta fase de desenvolvimento.

Em suma, fica claro que a versão atual do sistema se encontra disponível e funcionando no cenário que foi proposto, mesmo com os problemas pontuais citados acima. Alguns pontos que podem ajudar na melhoria do sistema em próximas versões são a introdução de um designer de interfaces para aprimorar o visual e usabilidade do sistema, time de desenvolvimento focado na manutenção e correção do sistema, resolvendo bugs descritos no quadro kanban e também implementando as funcionalidades que foram citadas anteriormente que não entraram nesta versão. Também seria interessante profissionais de teste de software para encontrar futuros bugs e repassá-los para os desenvolvedores, por fim, um especialista de devops para cuidar da parte de infraestrutura da aplicação, como hospedagem e banco de dados em nuvem. Com esses profissionais, os problemas citados acima seriam solucionados, visto que cada uma teria foco apenas em uma parte específica do sistema, dividindo para conquistar.

5 EXPERIÊNCIA

5.1 Processo de desenvolvimento

No desenvolvimento da aplicação, adotou-se o sistema *Kanban[14],* que consiste na utilização de um quadro, denominado *quadro kanban*, que possui basicamente três colunas, sendo elas *A fazer, em desenvolvimento* e *finalizado.* O quadro em questão foi criado em um projeto do *github*, pois desse modo seria possível gerenciar as issues diretamente relacionadas ao repositório do projeto, sendo possível por exemplo a automação dos cards de acordo com pull requests feitos e mudanças nos status das issues. *[Figura 10]*

As *user stories* definidas foram quebradas em atividades, que por sua vez foram divididas em issues de backend e frontend que foram alocadas na primeira coluna do quadro em questão, de onde foram sendo movimentadas de acordo com as respectivas prioridades.

Após a definição do seria feito, foram produzidos wireframes utilizando a plataforma *Miro,* que ajudariam a guiar a estrutura do frontend, as telas e componentes foram imaginados para que ficassem da forma mais intuitiva e simples possível, e também representando o mais fidedignamente possível a versão final das mesmas. *[Figura 11]*

O desenvolvimento foi iniciado através do backend, onde a configuração do ambiente foi feita, a estrutura das entidades foram definidas, os *services* configurados para manipular e tratar as informações, bem como aplicar as regras de negócio, os controllers criados para manipular as requisições e os arquivos de rotas para mapear os endpoints do sistema.

Finalizado o desenvolvimento do backend, foram realizados testes automatizados para garantir o correto funcionamento da aplicação, utilizando as tecnologias *Mocha, Chai* e também um banco de dados em memória para persistência não definitiva de dados. *[Figura 12]*

Após isso, foi iniciado o desenvolvimento do frontend, a estrutura foi definida e implementada, foi feita a configuração das bibliotecas necessárias e por fim, foram criadas as telas e componentes, bem como a estrutura de conexão com o backend.

A última etapa foi o deploy de ambas as partes para que o sistema fosse disponibilizado para os usuários testarem, foi configurado um banco de dados em nuvem no *Atlas* para utilização no *heroku,* plataforma onde foram hospedadas as duas partes.

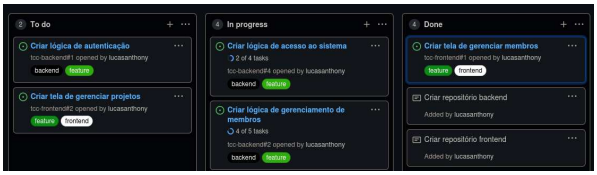


Figura 10: Exemplo do quadro kanban utilizado

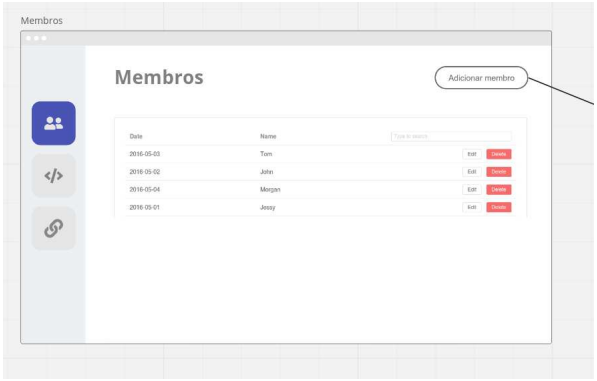


Figura 11: Wireframe da tela de membros

```
describe('@EJ', async function () {
    it('POST   | EJ | Cadastro de EJ', function (done) {
        chai.request(server)
            .post('/ej')
            .send(ejDefault)
            .end(async function (err, response) {
                response.should.have.status(data.HTTP_CODE.CREATED);
                const users = await User.find({});
                const ejs = await Ej.find({});
                users.length.should.be.equals(1);
                ejs.length.should.be.equals(1);
                done();
            })
    });
```

Figura 12: Exemplo de teste realizado

5.2 Principais desafios

O principal desafio foi o entendimento do Vue.js 3, pelo fato de nunca ter sido utilizado por mim em nenhum projeto anterior, desse modo, demandando um tempo de aprendizado para conseguir fazer sua aplicação no sistema da melhor forma possível.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

O projeto desenvolvido atendeu a ideia inicial de permitir a centralização de informações para os diretores da empresa júnior, com funcionalidades simples por estar na sua primeira versão, porém de grande valia para os usuários como foi apresentado na seção 4.3. Proporcionou o aprimoramento e uso de várias tecnologias e ferramentas apresentadas durante a graduação, como foi mostrado na seção 5. Por fim, gerou uma base para continuidade do trabalho, disponibilizando os códigos e estruturas criadas para que outros desenvolvedores possam utilizar para implementar outras funcionalidades, principalmente as citadas na próxima subseção.

6.2 Trabalhos futuros

O Sistema de Gerenciamento de Empresas Juniores possui um escopo simples e um conjunto de funcionalidades básicas, porém fornece uma base feita de forma sólida e escalável tanto do lado do cliente quanto do servidor, desse modo, novas funcionalidades podem ser integradas de forma simples e rápida, sendo algumas delas:

- *Gerenciamento de atas de reuniões e gerenciamento de receita,* que podem ser implementados criando-se os respectivos módulos no backend e frontend da aplicação, cada um encapsulando suas funções e telas, dessa forma seguindo o padrão implementado no projeto e facilitando a compreensão.

- *Implementação do usuário membro, com funções limitadas,* que seria implementado através da adição de um novo tipo de usuário chamado *membro* na base de dados, a partir disso, seria necessário a adaptação da lógica de autorização tanto no cliente quanto no servidor, não demandando muito esforço pela forma que a mesma foi feita.

- *Módulo de analytics para uma visão geral dos dados da empresa.* Para implementá-la seria parecido com qualquer outro módulo existente, diferenciando apenas no frontend, que teria que utilizar alguma biblioteca de gráficos, como *chart.js,* que é bem simples e completa para o cenário do projeto.

- *Versão mobile do sistema,* seria a implementação que demandaria o maior esforço para ser feita, pois precisaria ser utilizada uma tecnologia diferente da versão web, como por exemplo o *react native*, no entanto, o backend está totalmente preparado para fornecer os dados para a aplicação mobile, assim como faz na versão web.

Os repositórios onde se encontram os códigos do frontend e do backend encontram-se nos seguintes links:

- https://github.com/lucasanthony/tcc-backend
- https://github.com/lucasanthony/tcc-frontend

8 REFERÊNCIAS

Esta seção faz menção aos conteúdos estudados, e tecnologias citados durante o processo de construção deste trabalho, são eles:

[1] Empresa Júnior. Disponível em: https://proex.ufes.br/o-que-e-uma-empresa-junior#:~:text=Segundo%20a%20Lei%20n%C2%B0,acad%C3%AAmico%20e%20profissional%20dos%20associados%2C.

[2] CodeX Jr. Disponível em: https://codexjr.com.br/.

[3] Elicitação de requisitos. Disponível em: https://sites.google.com/site/tecnicaselicitacao/.

[4] Brainstorming. Disponível em: https://pt.wikipedia.org/wiki/Brainstorming.

[5] Node. Disponível em: https://nodejs.org/en/.

[6] Spring Boot. Disponível em: https://nodejs.org/en/.

[7] MongoDB. Disponível em: https://www.mongodb.com/pt-br/cloud/atlas/efficiency.

[8] VueJs. Disponível em: https://vuejs.org/.

[9] Pug. Disponível em: https://pugjs.org/api/getting-started.html.

[10] JWT. Disponível em: https://jwt.io/.

[11] Heroku. Disponível em: https://www.heroku.com/platform.

[12] MVP. Disponível em: https://pt.wikipedia.org/wiki/Produto_vi%C3%A1vel_m%C3%ADnimo.

[13]Escala Likert. Disponível em: https://pt.wikipedia.org/wiki/Escala_Likert.

[14] Kanban. Disponível em: https://pt.wikipedia.org/wiki/Kanban.