UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

BEATRIZ DE ALMEIDA FERNANDES DE LIMA

PILAS: UM SISTEMA WEB PARA CONTROLE DO CONSUMO DE GULOSEIMAS EM UMA EMPRESA DE TECNOLOGIA

PROJETO DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

GUARAPUAVA
2022

BEATRIZ DE ALMEIDA FERNANDES DE LIMA

PILAS: UM SISTEMA WEB PARA CONTROLE DO CONSUMO DE GULOSEIMAS EM UMA EMPRESA DE TECNOLOGIA

Projeto de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Sistemas para Internet – TSI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Andres Jessé Porfirio

Coorientadores: Henrique Fernando de Oliveira Rodrigues e Maurício Barfknecht

GUARAPUAVA
2022

RESUMO

ALMEIDA, Beatriz. Pilas: um sistema web para controle do consumo de guloseimas em uma empresa de tecnologia. 2022. 18 f. Projeto de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2022.

Cultura empresarial ou organizacional é o conjunto de valores, costumes e ideais inseridos no dia a dia de uma empresa com intuito de engajar seus colaboradores e proporcionar bem estar e qualidade de vida à equipe. Tendo isso em vista, uma empresa de tecnologia implantou um projeto denominado *Pilas*, onde seus colaboradores tem acesso a diversos itens alimentícios em troca de uma moeda fictícia. Seu controle se da atualmente através de planilhas que são manipuladas pelos próprios usuários, o que as torna suscetíveis a falhas, não havendo um sistema computacional capaz de suprir tais demandas, dadas suas características únicas e específicas. Posto isto, com a finalidade de auxiliar esse projeto, o presente trabalho busca desenvolver um sistema web com versão em PWA[1] que gerencia e automatiza os processos envolvidos, como cadastro de produtos para compra e as movimentações de aquisição dos mesmos, bem como gerenciamento de usuários com diferentes níveis de permissões.
Palavras-chave: Cultura organizacional. Empresas. Tecnologia. Programação para Internet.

[1] Progressive Web App: aplicação híbrida entre as páginas da web regulares e um aplicativo móvel.

ABSTRACT

ALMEIDA, Beatriz. Pilas: a web system to treats consumption control in a technology company. 2022. 18 f. Projeto de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2022.

Organizational culture is the set of values, customs and ideals that are a everyday life part of a company in order to engage its employees and provide well-being and quality of life to the team. Thinking of this, a technology company implanted a project called *Pilas*, where its employees have access to several kinds of food in exchange for a fictitious coin. As there is no computational system capable to fulfill this project demands, given their unique and specific characteristics, its control currently takes place through spreadsheets that are manipulated by users themselves, which makes them likely to failures. With the aim of helping this project, the present work seeks to develop a web system with a version in PWA, which will be able to manages and automates the procedures of *Pilas*, such as the registration of products and their purchase transactions, as well as management of users with different levels of permissions.

Keywords: Organizational culture. Company. Technology. Internet Programming.

LISTA DE FIGURAS

Figura 1 – Tela de resumo de vendas do sistema Nex 4
Figura 2 – Tela de registro de um produto no aplicativo Contestoque 5
Figura 3 – Tela de movimentação de venda no sistema Tiny 6
Figura 4 – Fluxograma do ciclo da feature 11
Figura 5 – Design System 13

LISTA DE QUADROS

Quadro 1 – Cronograma de Atividades. 15

LISTA DE TABELAS

Tabela 1 – Requisitos não funcionais do sistema 12
Tabela 2 – Requisitos funcionais do sistema 14

LISTA DE ABREVIATURAS E SIGLAS

PWA	Progressive Web App
ORM	Object-Relational Mapping
SQL	Structured Query Language
MoSCoW	Must, Should, Could e Wouldn't
PR	Pull Request
MVP	Minimum Viable Product
ERP	Enterprise Resource Planning

SUMÁRIO

1 – INTRODUÇÃO 1

1.1 Objetivos 2

1.1.1 Objetivo Geral 2

1.1.2 Objetivos Específicos 2

1.2 Pilas 2

2 – TRABALHOS CORRELATOS 4

2.1 Nex 4

2.2 Contestoque 5

2.3 Tiny ERP 5

3 – REFERENCIAL TEÓRICO 7

3.1 TypeScript 7

3.2 React.js e Next.js 7

3.3 PostgreSQL e Prisma ORM 8

3.4 PWA 8

3.5 Git e GitHub 9

4 – METODOLOGIA 10

5 – RESULTADOS PRELIMINARES 12

5.1 Levantamento de requisitos 12

5.2 Design System 12

5.3 Cronograma 12

6 – CONSIDERAÇÕES FINAIS 16

Referências 17

1 INTRODUÇÃO

Cultura empresarial ou organizacional é o conjunto de valores, costumes e ideais inseridos no dia a dia de uma empresa. Sua importância está no fato de ser uma excelente forma de engajar os colaboradores e torná-los mais unidos e motivados em prol do sucesso da organização, além de proporcionar bem estar e qualidade de vida à equipe. Alguns exemplos de atividades, atitudes ou rituais que podem compor uma cultura organizacional são: um ambiente descontraído com música e liberdade para conversar, *happy hours*[1], comemorações de aniversários e conquistas da equipe, premiações e refeições gratuitas (MOTTA; VASCONCELOS, 2002). O bem estar de um colaborador em uma empresa é fundamental, primeiro porque incentiva a produtividade, e depois, porque ajuda a amenizar o estresse que muitas vezes algumas atividades do trabalho podem trazer (MASLACH; SCHAUFELI; LEITER, 2001).

Diante disso, uma empresa de tecnologia, que preza a construção de uma forte cultura organizacional, deu início a um controle interno de consumo de produtos, batizado de *Pilas*. São exemplos desses produtos: balinhas, paçocas, biscoitos, chocolates, frutas, iogurtes, entre outros. Para consumi-los, cada colaborador recebe, por mês, um determinado saldo de uma moeda fictícia chamada *Pila*.

O abastecimento do estoque e a distribuição da moeda, é responsabilidade de um membro da equipe, eleito mês a mês de forma democrática como "prefeito". Atualmente, o controle de itens no estoque, de saldo dos consumidores e movimentações de compra, são controlados através de uma planilha compartilhada no Google Sheets[2]. O projeto *Pilas*, conta também com um agregado chamado de "Fome Zero", através do qual, a equipe tem livre acesso a alimentos básicos para lanches, como café, leite, pão, manteiga, doce de leite, bolacha, açúcar, dentre outras opções.

Dessa forma, o objetivo com o *Pilas* é disponibilizar todo mês **guloseimas**[3] **adicionais**, e que normalmente uma empresa não provê gratuitamente. Contudo, se esse tipo de produto fosse de livre consumo, haveria um grande risco de todo o estoque ser consumido compulsivamente em poucos dias e de forma desproporcional, mais por alguns do que por outros. Adicionalmente, existe a preocupação com a saúde dos colaboradores, pois a ingestão exagerada desses alimentos ultraprocessados cotidianamente, pode ser prejudicial (BIELEMANN et al., 2015).

O modelo atual em execução é falho, pois existem muitas brechas para inconsistências, visto que cada um anota seus gastos, e por engano ou mesmo má fé, pode acabar fazendo anotações incorretas, não havendo uma forma de fiscalizar isso. Por conseguinte, destaca-se a necessidade de implementação de uma solução computacional com maior poder de gerência

[1] Reunião ou comemoração informal, normalmente envolvendo colegas de trabalho ou estudo, realizada após expediente de trabalho com petiscos e bebidas.

[2] Serviço de planilhas online, website oficial: https://sheets.google.com/

[3] Gíria utilizada para se referir a gulas, doces.

em relação às atuais planilhas, promovendo controle e segurança nas transações de consumo de guloseimas da empresa.

Dito isso, o presente documento propõe o desenvolvimento de um sistema web com versão também em PWA, que fará toda essa gestão, hoje realizada através de planilhas. Por meio deste sistema, será possível o cadastro de usuários, a verificação de saldo e extrato de *Pilas* e a realização de compra dos itens cadastrados. Além de possuir uma área exclusiva para o prefeito, que fará a gestão dos produtos, saldos, usuários e permissões.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolvimento de um sistema para controle do consumo de guloseimas pelos colaboradores de uma empresa de tecnologia.

1.1.2 Objetivos Específicos

- Desenvolver um sistema web que possibilite o cadastro de usuários com diferentes níveis de permissão;
- Permitir que o prefeito gerencie produtos;
- Permitir que o prefeito controle os saldos dos usuários, podendo conceder bônus ou confiscar *Pilas*;
- Permitir ao usuário a consulta de saldo e extrato de consumo de *Pila*;
- Possibilitar aos usuários realizarem movimentações de compra de itens disponíveis;
- Gerar relatórios de consumo para aprimorar o controle de estoque;
- Possuir compatibilidade com diversos tipos de sistemas operacionais, incluindo os de dispositivos móveis;
- Publicar a aplicação como PWA;
- Promover facilidade e agilidade na localização e compra de itens através de interface minimalista e intuitiva;
- Permitir que o sistema esteja disponível exclusivamente de forma online, evitando que o usuário precise realizar instalações e configurações.

1.2 Pilas

No primeiro semestre de 2022, foi iniciado na disciplina de Programação para Dispositivos Móveis, na UTFPR Campus Guarapuava, o desenvolvimento de um aplicativo com a ideia de suprir as necessidades apontadas no presente trabalho. Para desenvolvimento inicial desse aplicativo, foram utilizadas tecnologias como React Native, Node.js, Expo e Firebase. Proporcionando a autenticação de usuários com diferentes permissões, cadastros de produtos e opção para compra dos mesmos utilizando o saldo disponível.

Após o término da disciplina, foi considerada a possibilidade de dar continuidade ao aplicativo, tendo em vista uma empresa de tecnologia com interesse em automatizar o seu processo, mas foram observadas algumas limitações. Este teria que ser implantado para utilização dos colaboradores da empresa, sendo disponibilizado nas lojas de aplicativos, tanto de sistemas Android, quanto iOS. Analisando-se os termos dessas lojas, observou-se que ambas as plataformas exigem o pagamento de licenças e o custo para disponibilizá-lo para iOS[4] seria inviável em um primeiro momento.

Além disso, a disponibilização do sistema como um aplicativo móvel nativo nas lojas de Android e iOS não permite o teste instantâneo de novas funcionalidades e/ou correção de bugs, visto que cada plataforma possui um fluxo de *deploy*[5] próprio, que muitas vezes requer um período de espera até a aprovação de novas versões. Diante disso, cada atualização no sistema pode levar desde algumas horas, até dias para que todos os usuários recebam a notificação da nova versão e atualizem em seus dispositivos. Em contrapartida, uma aplicação web não requer essa espera, dado que feito o *deploy*, todos os usuários ficam imediatamente aptos a utilizarem o sistema atualizado.

Tendo em vista essas limitações, descartou-se a continuidade do aplicativo, dando início à ideia de desenvolvimento de um sistema web que inclui todos os tipos de usuários no que diz respeito aos seus dispositivos e fornece atualizações de maneira transparente. Destaca-se que a construção do sistema utilizando a abordagem mobile-first (ROTH, 2019) visando publicação como um PWA é uma possibilidade a ser investigada, pois permitiria um uso amigável do sistema em dispositivos móveis.

[4]Apple Developer Program: https://developer.apple.com/support/compare-memberships/

[5]Do inglês "implantar", é um termo utilizado em programação em seu sentido literal, para uma aplicação que é implantada.

2 TRABALHOS CORRELATOS

Dada a necessidade de um sistema para o controle supracitado, foram identificadas e analisadas algumas soluções existentes a fim de verificar se estas satisfazem os objetivos almejados.

2.1 Nex

O Nex[1], é um sistema para gestão de pequenos comércios, que permite o cadastro de produtos, fornecedores, histórico de movimentação, conferência de estoque, entre outros recursos relacionados. Embora possua funções semelhantes as buscadas, o método utilizado é pensado exclusivamente para comércios, não se encaixando com o *Pilas*, onde cada colaborador poderá agir autonomamente, comprando seus próprios itens dentro do sistema. O sistema Nex como um todo é complexo para o propósito desejado e contém vários processos desnecessários para o *Pilas*, além de não possuir uma gestão de usuários com saldo. A Figura 1 traz um exemplo de tela do sistema Nex.

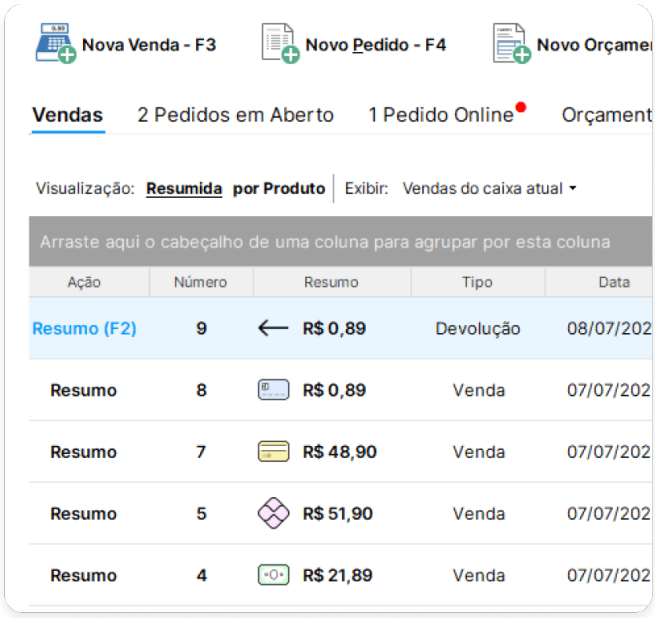

Figura 1 – Tela de resumo de vendas do sistema Nex
Fonte: Nex (2022)

[1] https://www.nextar.com.br

2.2 Contestoque

O Contestoque[2] foi desenvolvido de forma exclusiva para controlar estoque. Ele permite o cadastro de produtos com data de vencimento e código de barras, contando com um leitor que utiliza a câmera do celular e permitindo também o controle de quantidade dos itens, conforme mostra a Figura 2. Esse aplicativo atende principalmente aqueles que possuem estoque de alimentos, dos quais precisam ter completo controle, como comércios, restaurantes, ou cozinhas de instituições em geral. Pode também atender a necessidade de organizações que possuem outros tipos de estoque, como de ferramentas e equipamentos. Apesar de englobar algumas das funções buscadas para o *Pilas* na parte de estoque, o Contestoque não possui gestão de usuários ou de qualquer tipo de moeda, tampouco funções para compra de produtos, o que vem a descartar seu uso.

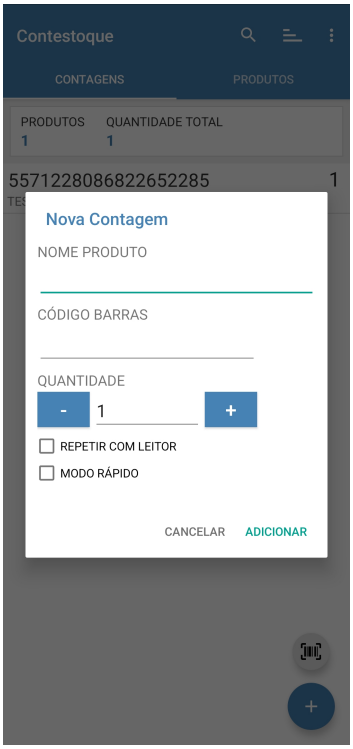

Figura 2 – Tela de registro de um produto no aplicativo Contestoque
Fonte: Contestoque (2022)

2.3 Tiny ERP

O Tiny[3] é um sistema web do tipo ERP (Enterprise Resource Planning) que une diversas funcionalidades úteis para as operações de uma empresa. Através dele é possível integrar alguns recursos como de vendas (Figura 3), cadastro e controle de produtos, controle

[2] http://www.contestoque.com.br/
[3] https://www.tiny.com.br/

de estoque e emissão de boletos e notas fiscais, além de possibilitar integração com diversos marketplaces para unificar as vedas da empresa. O sistema é pago e trabalha no formato modular, permitindo uma contratação personalizada com apenas as funções desejadas.

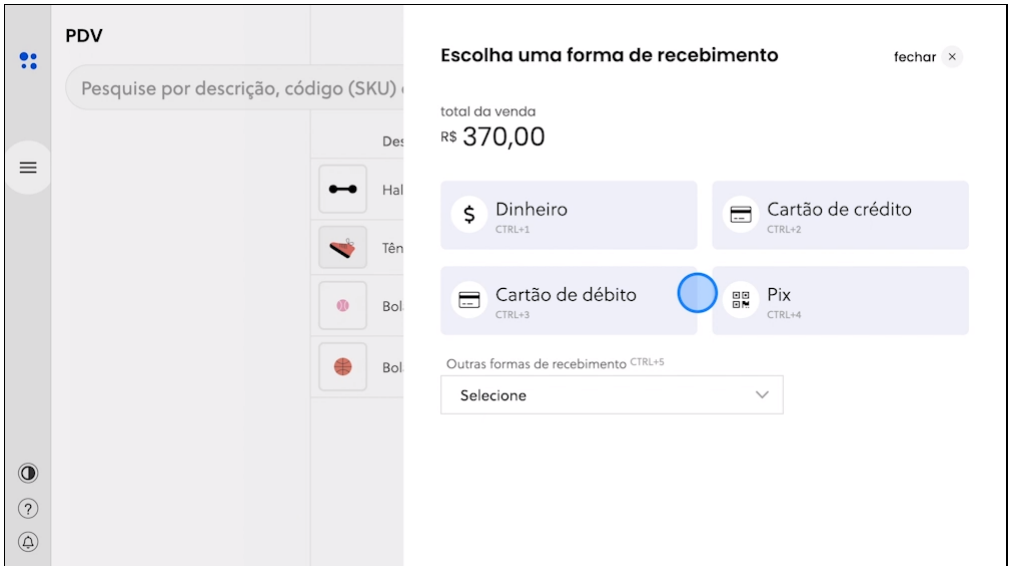

Figura 3 – Tela de movimentação de venda no sistema Tiny
Fonte: Tiny (2022)

Apesar de bastante abrangente, o Tiny, assim como ambos os sistemas citados anteriormente, não trabalha com gestão de usuários de diferentes níveis de permissões, onde cada um possua um saldo para realizar suas próprias movimentações de compras de produtos e possa acompanhar seu extrato, consequentemente torna-se também inviável para o objetivo buscado.

3 REFERENCIAL TEÓRICO

Neste capítulo estão apontadas e descritas as tecnologias que serão utilizadas para desenvolver um sistema que atenda os objetivos propostos no presente projeto.

3.1 TypeScript

A **linguagem de programação** que será utilizada é o TypeScript[1], um superconjunto de JavaScript que agrega novos recursos a esta, como a **tipagem estática** que consiste em definir regras para a forma em que cada tipo de valor pode ser utilizado. Sendo assim, enquanto para o JavaScript variáveis *string* ou numéricas podem ser tratadas da mesma forma, para o TypeScript o tipo dessas variáveis precisa ser devidamente declarado para que o código seja compilado corretamente (MICROSOFT, 2022).

A principal vantagem em utilizar uma linguagem com tipagem estática é a redução de erros da aplicação em tempo de execução, o que auxilia e facilita o trabalho do programador (MEIJER; DRAYTON, 2004).

Apesar de aumentar um pouco a complexidade de desenvolvimento do sistema, optar pelo uso do TypeScript em detrimento de JavaScript proporciona uma estrutura mais escalável, o que é interessante para este projeto, já que poderão ser percebidas novas necessidades conforme o crescimento do *Pilas* e o uso da aplicação.

3.2 React.js e Next.js

React[2] é uma **biblioteca JavaScript** de código aberto que foi desenvolvida por engenheiros do Facebook em 2013, a fim de resolver alguns desafios relacionados a interfaces de usuário que mudam constantemente (GACKENHEIMER, 2015). Sendo assim, seu objetivo é a criação de interfaces interativas de usuário (META, 2022).

Dentre as principais vantagens que o uso dessa ferramenta oferece, estão a experiência eficiente de usuário e a facilidade de escrita e reutilização de componentes (FEDOSEJEV, 2015). Mas apesar de ter o intuito de auxiliar no desenvolvimento, React é apenas uma biblioteca, ou seja, ele oferece recursos úteis para construção de interfaces, mas fica a critério do desenvolvedor a forma de utilizá-los, exigindo certo esforço para desenvolver uma aplicação do zero.

Diante disso, com o intuito de facilitar ainda mais o desenvolvimento do *Pilas*, será utilizado o Next.js[3], um **framework de React** que provê diversos recursos prontos, como de roteamento, busca de dados, integração com outros serviços, infraestrutura, performance e escalabilidade (VERCEL, 2022). Uma das principais vantagens da utilização do Next.js neste

[1] https://www.typescriptlang.org/
[2] https://pt-br.reactjs.org/
[3] https://nextjs.org/

projeto é a sua capacidade de execução de código no lado servidor, eliminando a necessidade de um sistema auxiliar (como uma API) para fornecimento de recursos à interface React.js, que é nativamente executada no lado cliente.

3.3 PostgreSQL e Prisma ORM

Para **base de dados** optou-se pelo PostgreSQL[4], uma ferramenta de código aberto do tipo **objeto relacional** que possui inúmeros recursos avançados para armazenar e escalar os dados de forma segura (POSTGRESQL, 2022), sendo ideal para aplicações de pequeno, médio ou grande porte e estando entre os bancos mais utilizados quando há busca por maior estabilidade (MILANI, 2008).

Um banco de dados do tipo objeto relacional une características de bancos relacionais e orientados a objetos. Isso significa que proporciona maior escalabilidade juntamente com suporte para tipos de dados complexos (DEVARAKONDA, 2001).

Além dos pontos supracitados, o fato de a empresa onde o sistema será implementado já utilizar PostgreSQL para suas bases de dados, foi um ponto de grande influência na escolha do mesmo para execução do projeto.

O Prisma[5] trata-se de uma **ferramenta ORM** (Object Relational Mapper) também de código aberto. Seu objetivo é facilitar para o desenvolvedor, a comunicação entre os objetos TypeScript e os dados de banco que os representam, através de métodos que dispensando a necessidade de codificação de elaboradas consultas em SQL (PRISMA, 2022).

3.4 PWA

Um PWA trata-se de uma **aplicação híbrida** que proporciona uma experiência semelhante ao uso de um aplicativo nativo, mas que na realidade é executada no navegador do dispositivo móvel. Algumas das vantagem de sua utilização são a possibilidade de ocultar a barra de endereço e demais botões do navegador e o fato de não ocupar espaço no armazenamento do dispositivo, já que não exige instalação.

Apesar de ser uma forma atrativa de disponibilizar aplicações, o PWA pode fornecer limitações quando se trata do acesso à recursos nativos do dispositivo (como restrições de espaço de armazenamento e ausência de notificações push no iOS). Embora existam desvantagens em restringir o acesso completo de alguns recursos dos dispositivos móveis, o fato de ser compatível com qualquer sistema operacional, acabou encorajando a escolha dessa tecnologia para desenvolvimento do sistema que está sendo proposto. Assim proporcionando um meio uniforme para a disponibilização da aplicação a todos os colaboradores da empresa, independente do tipo de dispositivo e sistema operacional utilizado (englobando desde celulares e tablets até computadores de mesa) (ADETUNJI et al., 2020).

[4] https://www.postgresql.org/
[5] https://www.prisma.io/

3.5 Git e GitHub

O Git é um sistema de controle de versões utilizado para versionamento de códigos de programação, possibilitando manter um histórico de todas as alterações realizadas, bem como fácil recuperação das mesmas. O GitHub por sua vez, trata-se de uma plataforma online onde são hospedados os repositórios do Git. Essas duas ferramentas formam juntas um grande aliado em projetos desenvolvidos em time, pois permitem que vários desenvolvedores trabalhem simultaneamente em um mesmo sistema sem gerar conflitos (BLISCHAK; DAVENPORT; WILSON, 2016).

Uma das grandes motivações para utilização do Git e GitHub no desenvolvimento deste projeto, é o fato de que com elas é possível trabalhar com diferentes *branchs* (ramificações) de um mesmo projeto, onde cada uma pode ser direcionada para o desenvolvimento de uma funcionalidade específica. Além disso o desenvolvedor pode criar *PRs* (Pull Requests) no GitHub para que suas branchs sejam revisadas e aprovadas pelo restante da equipe, ou por um responsável (neste caso, o professor orientador), antes da mesma ser mesclada a versão oficial do código.

4 METODOLOGIA

O primeiro passo para a execução deste projeto é o levantamento dos seus requisitos, sendo esses funcionais e não funcionais, onde os requisitos funcionais serão classificados através do método MoSCoW (KUHN, 2009), conforme suas prioridades. O nome MoSCoW é um acrônimo das palavras em inglês **M**ust, **S**hould, **C**ould e **W**ouldn't, onde as letras "o", foram adicionadas apenas para tornar o acrônimo mais pronunciável. Este método traz 4 (quatro) tipos de classificações, sendo elas, requisitos que o projeto:

- Precisa ter (Must have): aqueles requisitos que são essenciais e obrigatórios para o funcionamento do projeto;
- Deveria ter (Should have): requisitos importantes, mas não essenciais;
- Poderia ter (Could have): são os requisitos desejáveis, mas não vitais, eles podem ou não ser atendidos, conforme disponibilidade de tempo e recursos;
- Não teria (Wouldn't have): requisitos que estão fora do atual escopo.

Feito o levantamento e a classificação dos requisitos, serão definidas as melhores tecnologias a serem utilizadas para desenvolvimento do projeto, analisando as que mais se adéquam ao objetivo proposto. Isso inclui tecnologias para *frontend*, *backend*, banco de dados e *frameworks* em geral que auxiliam e agilizam os processos. Em seguida, será construído um Design System (MACDONALD, 2019) com o auxílio da plataforma Figma[1], onde serão definidos padrões de layout e paletas de cores, por exemplo.

Uma vez que todas as definições inicias tenham sido feitas, os requisitos levantados serão passados para *features*[2], estabelecendo junto do orientador um prazo estimado para o desenvolvimento de cada uma delas, o que possibilita um planejamento mais preciso de tempo para o projeto como um todo. Menciona-se que tais prazos podem sofrer alterações na medida em que as funcionalidades forem desenvolvidas e suas reais complexidades se tornem mais evidentes.

As *features* serão gerenciadas pelo modelo **Feature Branch Workflow**, através da ferramenta Git. Este modelo consiste em um desenvolvimento guiado por *features* que seguem sempre um mesmo ciclo, o qual funcionará conforme ilustrado no fluxograma da Figura 4.

O ciclo tem início na escolha da *feature* a ser desenvolvida, a qual corresponde a um requisito que foi levantado. É neste momento que os prazos definidos poderão ser revisados pela aluna e pelo orientador caso haja necessidade. Em seguida é criada uma nova *branch* local para que se inicie a escrita dos códigos de programação, realizando novos *commits* a cada etapa até que a mesma seja concluída. Essa divisão em vários *commits*, auxilia para que haja um versionamento do código, facilitando que alterações sejam revertidas se necessário. Uma vez que a tarefa tenha sido desenvolvida, é feito o *push* dessas alterações para a *branch*

[1] https://www.figma.com/

[2] Termo utilizado para se referir a funcionalidades ou recursos a serem desenvolvidos para um sistema computacional, por exemplo.

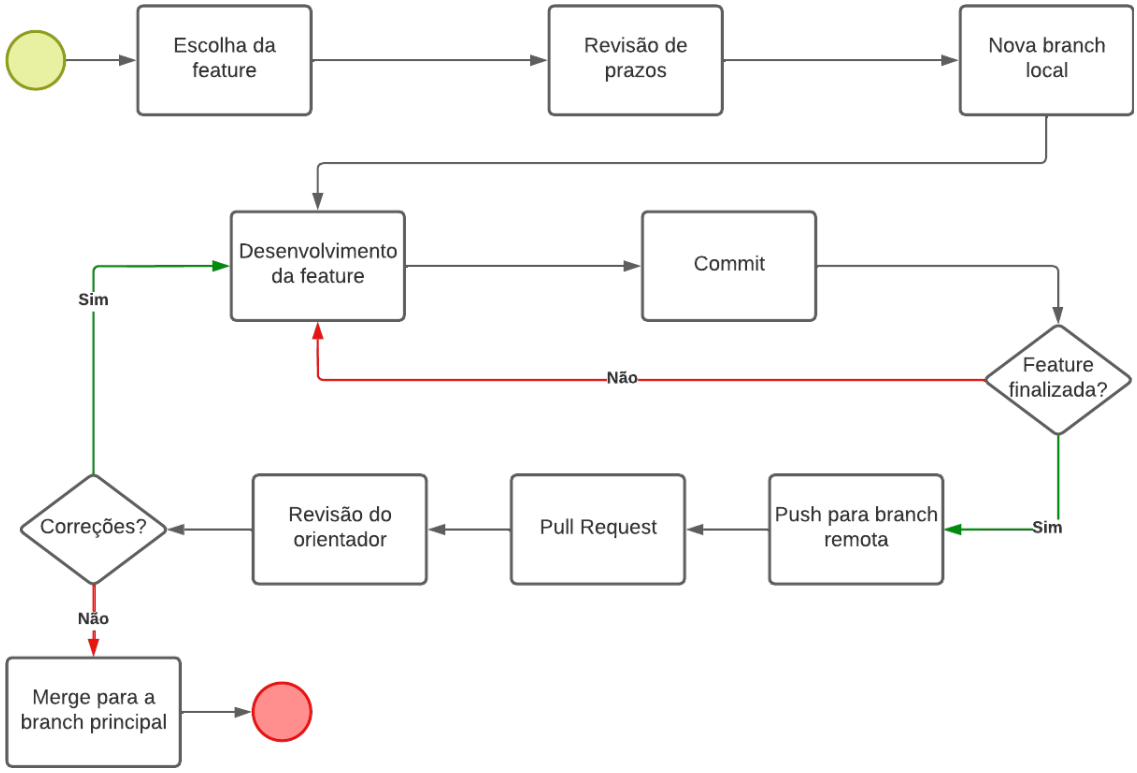

Figura 4 – Fluxograma do ciclo da feature
Fonte: A autora

remota e através do GitHub é criada uma *Pull Request* (PR). Neste ponto, é solicitado que o professor orientador faça a revisão dos códigos e os apontamentos necessários e caso hajam correções ou alterações a serem realizadas, estas devem ser cumpridas e enviadas para a *branch* remota novamente para aguardarem nova revisão. Caso contrário, é feito o *merge* para a *branch* principal finalizando assim, o ciclo da *feature*.

A partir do momento em que as principais *features* tenham sido desenvolvidas e que obtenha-se um Produto Mínimo Viável – em inglês *Minimum Viable Product* ou MVP (LENARDUZZI; TAIBI, 2016) – , este será implantado na empresa em caráter de testes, para que os colaboradores possam dar suas opiniões e queixas a respeito da usabilidade e possíveis conflitos que possam surgir, os quais serão resolvidos a partir da criação e desenvolvimento de novas *features*.

5 RESULTADOS PRELIMINARES

Neste capítulo serão descritos os resultados obtidos desde a proposta até o momento de entrega deste projeto.

5.1 Levantamento de requisitos

Esta sessão apresenta o levantamento preliminar dos requisitos do sistema. A Tabela 1 traz a listagem dos requisitos não funcionais, enquanto a Tabela 2 exibe os requisitos funcionais que foram elaborado conforme a metodologia explanada no Capítulo 4.

Requisito	Descrição
RNF01	Possuir compatibilidade com diversos tipos de sistemas operacionais, incluindo os de dispositivos móveis
RNF02	Publicar a aplicação como PWA
RNF03	Promover facilidade e agilidade na localização e compra de itens através de interface minimalista e intuitiva
RNF04	Permitir que o sistema esteja disponível exclusivamente de forma online, evitando que o usuário precise realizar instalações e configurações

Tabela 1 – Requisitos não funcionais do sistema.
Fonte: A autora

5.2 Design System

Com o intuito de direcionar e padronizar o design do sistema, foi criado um conjunto de componentes básicos de estilo através da ferramenta Figma, conforme ilustra a Figura 5. Esse tipo de recurso também se faz útil para o desenvolvimento *front-end* em momentos em que o sistema precisa ser escalado.

5.3 Cronograma

O planejamento do trabalho de TCC que será desenvolvido ao longo do período letivo, está descrito no cronograma do Quadro 1. Neste cronograma constam todas as atividades com seus respectivos prazos para o cumprimento.

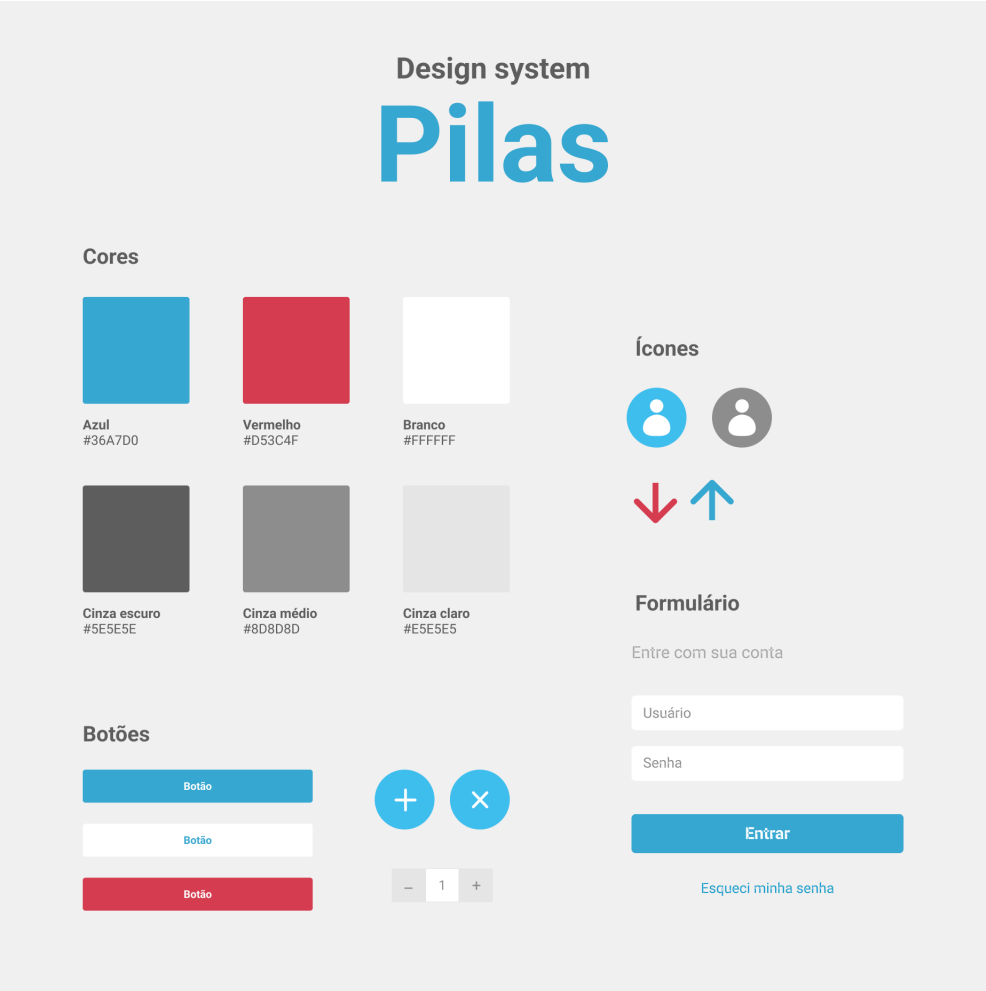

Figura 5 – Design System
Fonte: A autora

Requisito	Descrição	Prioridade
RF01	Possuir um usuário nativo administrador	Must
RF02	Permitir que um usuário faça logon	Must
RF03	Permitir que um usuário logado faça logoff	Must
RF04	Permitir que o administrador faça cadastro de usuários	Must
RF05	Permitir que o administrador edite usuários cadastrados	Must
RF06	Permitir que que o administrador liste os usuários cadastrados	Must
RF07	Disponibilizar função para atribuir o papel de prefeito à um usuário cadastrado	Must
RF08	Disponibilizar função para remover o papel de prefeito de um usuário cadastrado	Must
RF09	Disponibilizar função para desativar um usuário ativo	Must
RF10	Disponibilizar função para ativar um usuário desativado	Must
RF11	Permitir que o prefeito faça cadastro de produtos	Must
RF12	Permitir que o prefeito edite produtos cadastrados	Must
RF13	Permitir que o prefeito remova produtos cadastrados	Must
RF14	Permitir que um usuário liste os produtos disponíveis	Must
RF15	Permitir que um usuário compre produtos	Must
RF16	Permitir que um usuário verifique seu saldo	Must
RF17	Permitir que um usuário edite detalhes de sua própria conta	Should
RF18	Permitir que um usuário pesquise por produtos específicos	Should
RF19	Permitir que o prefeito faça cadastro de categorias de produtos	Should
RF20	Permitir que o prefeito edite categorias de produtos cadastradas	Should
RF21	Permitir que o prefeito remova categorias de produtos, desde que sem produto vinculado	Should
RF22	Permitir que um usuário visualize seu extrato de movimentações	Should
RF23	Permitir que o prefeito gere relatórios	Could
RF24	Disponibilizar função para que o prefeito atribua Pilas à um usuário	Could
RF25	Disponibilizar função para que o prefeito confisque Pilas de um usuário	Could
RF26	Permitir que o administrador faça cadastro de grupos de usuários	Could
RF27	Permitir que o administrador edite grupos de usuários cadastrados	Could
RF28	Permitir que o administrador remova grupos de usuários, desde que sem usuários vinculados	Could
RF29	Possuir um "Portal da transparência"onde os usuários podem acompanhar o uso da verba por parte do prefeito	Would Not
RF30	Possuir uma funcionalidade onde os usuários podem avaliar o desempenho de cada prefeito ao fim do mandato	Would Not
RF31	Disponibilizar função para leitura de código de barras de produtos	Would Not

Tabela 2 – Requisitos funcionais do sistema.

Fonte: A autora

Quadro 1 – Cronograma de Atividades.

Atividades	**Dez/22**	**Jan/23**	**Fev/23**	**Mar/23**	**Abr/23**	**Mai/23**	**Jun/23**	**Jul/23**
1. Redação do projeto de TCC								
2. Defesa do projeto de TCC								
3. Revisão dos apontamentos da banca								
4. Revisão bibliográfica								
5. Desenvolvimento do sistema proposto								
6. Escrita da Monografia de TCC								
7. Elaboração da apresentação final								
8. Defesa final do TCC								

6 CONSIDERAÇÕES FINAIS

Esse trabalho propõe o desenvolvimento de um sistema web para controle do consumo de guloseimas pelos colaboradores de uma empresa de tecnologia em um projeto interno denominado *Pilas*, tendo a finalidade de substituir o método atualmente utilizado, onde esse controle se dá por meio de planilhas estáticas suscetíveis a falhas.

O *Pilas* é um projeto descontraído mas com propósito sério de fortalecimento da cultura organizacional interna da empresa em questão, sendo uma iniciativa que visa gerar engajamento e proporcionar bem estar à toda equipe. Dadas suas características e demandas únicas, não foi encontrado um sistema existente que suprisse por completo as necessidades do *Pilas*, encorajando dessa forma a idealização do presente trabalho, que pretende cumpri-las através de um sistema objetivo e intuitivo, com uma interface agradável ao usuário.

Dentre alguns dos pontos de dificuldades que foram encontrados na fase de definição de tecnologias a serem utilizadas, estava a tarefa de disponibilizar um sistema adepto a dispositivos móveis, visando a praticidade de uso no dia a dia, mas que atendesse a usuários de diferentes sistemas operacionais, o que conduziu à decisão pelas ferramentas apresentadas no Capítulo 3, onde o PWA é apontado como uma solução plausível.

Referências

ADETUNJI, O. et al. Dawning of progressive web applications (pwa): Edging out the pitfalls of traditional mobile development. **American Academic Scientific Research Journal for Engineering, Technology, and Sciences**, v. 68, n. 1, p. 85–99, 2020. Citado na página 8.

BIELEMANN, R. M. et al. Consumo de alimentos ultraprocessados e impacto na dieta de adultos jovens. **Revista de Saúde Pública**, SciELO Public Health, v. 49, p. 28, 2015. Citado na página 1.

BLISCHAK, J. D.; DAVENPORT, E. R.; WILSON, G. A quick introduction to version control with git and github. **PLoS computational biology**, Public Library of Science, v. 12, n. 1, p. e1004668, 2016. Citado na página 9.

CONTESTOQUE. [S.l.], 2022. Disponível em: <http://www.contestoque.com.br/>. Acesso em: 12 de outubro de 2022. Citado na página 5.

DEVARAKONDA, R. S. Object-relational database systems—the road ahead. **XRDS: Crossroads, The ACM Magazine for Students**, ACM New York, NY, USA, v. 7, n. 3, p. 15–18, 2001. Citado na página 8.

FEDOSEJEV, A. **React. js essentials**. [S.l.]: Packt Publishing Ltd, 2015. Citado na página 7.

GACKENHEIMER, C. **Introduction to React**. [S.l.]: Springer, 2015. v. 52. Citado na página 7.

KUHN, J. Decrypting the moscow analysis. **The workable, practical guide to Do IT Yourself**, v. 5, 2009. Citado na página 10.

LENARDUZZI, V.; TAIBI, D. Mvp explained: A systematic mapping study on the definitions of minimal viable product. In: IEEE. **2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.], 2016. p. 112–119. Citado na página 11.

MACDONALD, D. **Practical ui patterns for design systems: Fast-track interaction design for a seamless user experience**. [S.l.]: Apress, 2019. Citado na página 10.

MASLACH, C.; SCHAUFELI, W. B.; LEITER, M. P. Job burnout. **Annual review of psychology**, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 52, n. 1, p. 397–422, 2001. Citado na página 1.

MEIJER, E.; DRAYTON, P. Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages. In: CITESEER. [S.l.], 2004. Citado na página 7.

META, P. **Documentação**. [S.l.], 2022. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 05 de dezembro de 2022. Citado na página 7.

MICROSOFT. **TypeScript for the New Programmer**. [S.l.], 2022. Disponível em: <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html>. Acesso em: 01 de dezembro de 2022. Citado na página 7.

MILANI, A. **PostgreSQL-Guia do Programador**. [S.l.]: Novatec Editora, 2008. Citado na página 8.

MOTTA, F. C. P.; VASCONCELOS, I. F. G. A cultura organizacional. **MOTTA, Fernando C. Prestes. Teoria geral da administração**, v. 3, 2002. Citado na página 1.

NEX. [S.l.], 2022. Disponível em: <http://www.nextar.com.br/>. Acesso em: 05 de outubro de 2022. Citado na página 4.

POSTGRESQL. **About**. [S.l.], 2022. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 05 de dezembro de 2022. Citado na página 8.

PRISMA. **Documentation**. [S.l.], 2022. Disponível em: <https://www.prisma.io/docs/concepts/overview>. Acesso em: 05 de dezembro de 2022. Citado na página 8.

ROTH, R. What is mobile first cartographic design. In: INTERNATIONAL CARTOGRAPHIC ASSOCIATION BERN, SWITZERLAND. **ICA Joint Workshop on User Experience Design for Mobile Cartography**. [S.l.], 2019. Citado na página 3.

TINY. **Tiny**. [S.l.], 2022. Disponível em: <https://www.tiny.com.br/>. Acesso em: 06 de dezembro de 2022. Citado na página 6.

VERCEL. **About Next.js**. [S.l.], 2022. Disponível em: <https://nextjs.org/learn/foundations/about-nextjs>. Acesso em: 06 de dezembro de 2022. Citado na página 7.