

**CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO
ITAJAÍ - UNIDAVI**

IVAN VINICIUS BONETI

PROTÓTIPO DE APLICAÇÃO PARA GESTÃO ORÇAMENTÁRIA DE LAVOURAS

**RIO DO SUL
2021**

**CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO
ITAJAÍ - UNIDAVI**

IVAN VINICIUS BONETI

PROTÓTIPO DE APLICAÇÃO PARA GESTÃO ORCAMENTÁRIA DE LAVOURAS

Trabalho de Conclusão a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Cleber Nardelli

**RIO DO SUL
2021**

**CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO
ITAJAÍ - UNIDAVI**

IVAN VINICIUS BONETI

PROTÓTIPO DE APLICAÇÃO PARA GESTÃO ORÇAMENTÁRIA DE LAVOURAS

Trabalho de Conclusão a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí - UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

Professor Orientador: Cleber Nardelli

Banca Examinadora:

Prof. Me. Jeancarlo Visentainer

Prof. Me. Fernando Andrade Bastos

Rio do Sul, 29 de junho de 2021.

Há um tempo para partir, mesmo quando não há um lugar certo para ir (Tennessee Williams).

A Deus e aos meus pais por todo apoio.

AGRADECIMENTOS

Aos meus pais, familiares e amigos que demonstraram interesse e ao mesmo tempo deram apoio durante o período de formação acadêmica.

Ao professor e orientador deste trabalho, Cleber Nardelli, no qual compartilhou de seu tempo e conhecimento para ajudar a tornar este projeto realidade.

RESUMO

Tecnologias para facilitar a vida dos gestores agrícolas surgem diariamente, mas infelizmente, nem todas trazem uma experiência simples e agradável, tampouco, melhoram a comunicação dos agricultores com os fornecedores de produtos e serviço, como agropecuárias. As propriedades agrícolas, sejam elas de pequeno ou grande porte dependem da correta administração de diversas atividades. Lidar com estoque, compra e venda de produtos e a gestão financeira, acaba deixando os produtores sobrecarregados. Diante das dificuldades enfrentadas, as inevitáveis perdas de lucratividade bem como gastos desnecessários causam a diminuição da competitividade e rentabilidade nas atividades do campo. Diante deste cenário, qual seria a forma mais fácil e acessível para resolver os problemas de gestão orçamentária de lavouras nas propriedades agrícolas, usando a tecnologia? É possível aumentar a rentabilidade e a competitividade no mercado com o uso das mesmas? Como a elaboração de orçamentos de produtos nas agropecuárias poderia ser facilitada? Do ponto de vista da usabilidade, de qual forma seria possível desenvolver uma aplicação de fácil entendimento para agricultores e agropecuaristas que não possuem muita afinidade com a tecnologia? Pensando nesses pontos, pretende-se através do presente trabalho, o desenvolvimento de um protótipo de aplicativo com foco justamente nesses problemas enfrentados pelos agricultores, espera-se que seja de grande valia, pois, através dele será possível analisar e prever gastos e investimentos nas atividades desempenhadas no campo. Todas essas funcionalidades serão desenvolvidas sem esquecer das facilidades de uso do usuário, utilizando técnicas de design de interface e experiência de usuário consolidadas.

Palavras-Chave: usabilidade, gestão agrícola, tecnologia.

ABSTRACT

Technologies to make life easier for agricultural managers emerge daily, but unfortunately, not all of them offer a simple and pleasant experience or improve communication between farmers and suppliers of products and services. Agricultural properties, whether small or large, depending on the correct administration of activities. Dealing with inventory, buying and selling products, and financial management can be hard. Facing difficulties as the inevitable loss of profitability, as well as unnecessary expenses, cause a reduction in the competitiveness and profitability of field activities. Given this scenario, what would be the easy and accessible way to solve the problems of budget management of crops in agricultural properties, using technology? Is it possible to increase profitability using them? How could the preparation of product budgets in agriculture be facilitated? From the usability view, what way would it be possible to develop an application which is easy to understand for farmers who do not have much affinity with the technology? Thinking about these points, it is intended through this work, the development of an application prototype focusing precisely on problems faced by farmers, is expected to be of great value because through it will be possible to analyze and predict expenses and investments in activities performed in the field. All these features will be developed without forgetting the ease of use by the user, using consolidated user experience and interface design techniques.

Keywords: usability, agricultural management, technologies.

LISTA DE FIGURAS

| | |
|--|----|
| Figura 1 - Mudança de estado da árvore DOM..... | 22 |
| Figura 2 - Forma de trabalho do React | 23 |
| Figura 3 - Diagrama de interseções e nomenclaturas do <i>UX Design</i> | 26 |
| Figura 4 - Cadastro de despesas do Aegro | 31 |
| Figura 5 - Aegro com a <i>API</i> do Google Maps no celular | 32 |
| Figura 6 - Diagrama de caso de uso do agropecuarista e administrador do sistema | 37 |
| Figura 7 - Diagrama de caso de uso do agricultor em seu aplicativo móvel | 38 |
| Figura 8 - Diagrama de atividades sobre a criação de composições de produtos..... | 39 |
| Figura 9 - Diagrama de atividades do orçamento realizado pelo agricultor | 40 |
| Figura 10 - Diagrama de entidade e relacionamento DER..... | 41 |
| Figura 11 - Dialogo simples e natural | 42 |
| Figura 12 - Falar a linguagem do usuário | 43 |
| Figura 13 - Minimizar a carga de memória do usuário | 44 |
| Figura 14 - Consistência..... | 44 |
| Figura 15 - <i>Feedback</i> | 45 |
| Figura 16 - Saídas evidentes | 46 |
| Figura 17 - Atalhos | 46 |
| Figura 18 - Mensagens de erro | 47 |
| Figura 19 - Prevenção de erros..... | 48 |
| Figura 20 - Documentação de ajuda | 49 |
| Figura 21 - Uso de <i>decorators</i> na classe de usuário..... | 50 |
| Figura 22 - Repositório customizado de usuário | 51 |
| Figura 23 - JSX no componente React..... | 51 |
| Figura 24 - Componente de <i>button</i> com Styled Components | 52 |

| | |
|---|-----------|
| Figura 25 - Diagrama de distribuição da aplicação | 54 |
| Figura 26 - Formulário de cadastro da agropecuária no sistema <i>web</i>..... | 55 |
| Figura 27 - Tela inicial do sistema <i>web</i> do agropecuarista | 56 |
| Figura 28 - Formulário de cadastro de produto no sistema <i>web</i>..... | 57 |
| Figura 29 - Listagem de produtos do portfólio do estabelecimento no sistema <i>web</i> | 57 |
| Figura 30 - Listagem de composições do sistema <i>web</i> | 58 |
| Figura 31 - Informando a quantidade recomendada de produto na composição | 59 |
| Figura 32 - Detalhes da composição no sistema <i>web</i> | 59 |
| Figura 33 - Tela inicial da aplicação móvel do agricultor | 60 |
| Figura 34 - Listagem de áreas do agricultor em seu aplicativo..... | 61 |
| Figura 35 - Cadastro da temporada de plantio..... | 62 |
| Figura 36 - Detalhes da composição do fornecedor no aplicativo do agricultor..... | 63 |

SUMÁRIO

| | |
|---|----|
| 1. INTRODUÇÃO | 11 |
| 1.1 PROBLEMA DE PESQUISA | 11 |
| 1.2 OBJETIVOS | 12 |
| 1.2.1 Geral | 12 |
| 1.2.2 Específicos | 12 |
| 1.3 JUSTIFICATIVA | 12 |
| 2. REVISÃO DA LITERATURA | 14 |
| 2.1 BANCO DE DADOS | 14 |
| 2.1.1 Tipos de banco de dados | 15 |
| 2.1.2 PostgreSQL | 16 |
| 2.2 JAVASCRIPT | 17 |
| 2.2.1 Javascript, estrutura léxica | 18 |
| 2.3 NODE | 19 |
| 2.4 REACT | 19 |
| 2.5 REACT NATIVE | 20 |
| 2.5.1 Vantagens | 21 |
| 2.5.2 Reutilização do código fonte | 21 |
| 2.5.3 Riscos e desvantagens do <i>framework</i> | 21 |
| 2.5.4 Trabalhando com React Native | 22 |
| 2.5.5 Criando Componentes com React Native | 23 |
| 2.5.6 Estilizando componentes nativos no React Native | 23 |
| 2.5.7 <i>Host Platform APIs</i> | 24 |
| 2.6 <i>USER EXPERIENCE DESIGN</i> | 24 |
| 2.6.1 O que é <i>UX Design</i> ? | 25 |
| 2.6.2 Arquitetura de informação | 25 |

| | |
|---|----|
| 2.6.3 Principais disciplinas do <i>UX Design</i> | 26 |
| 2.6.4 Usabilidade em produtos digitais | 27 |
| 2.7 HEURISTICA DE USABILIDADE | 27 |
| 3. METODOLOGIA | 29 |
| 3.1 ESTADO DA ARTE | 30 |
| 3.1.1 Aegro | 30 |
| 4. DESENVOLVIMENTO | 33 |
| 4.1 VISÃO GERAL DO SISTEMA | 33 |
| 4.2 RESULTADOS OBTIDOS COM OS QUESTIONAMENTOS | 34 |
| 4.3 REGRAS DE NEGÓCIO | 34 |
| 4.4 REQUISITOS | 35 |
| 4.5 DIAGRAMAS | 37 |
| 4.6 AVALIAÇÃO HEURISTICA DE USABILIDADE | 41 |
| 4.6.1 Dialogo simples e natural | 42 |
| 4.6.2 Falar a linguagem do usuário | 42 |
| 4.6.3 Minimizar a carga de memória do usuário | 43 |
| 4.6.4 Consistência | 44 |
| 4.6.5 <i>Feedback</i> | 45 |
| 4.6.6 Saídas evidentes | 45 |
| 4.6.7 Atalhos | 46 |
| 4.6.8 Mensagens de erro | 47 |
| 4.6.9 Prevenção de erros | 47 |
| 4.6.10 Documentação e ajuda | 48 |
| 4.7 IMPLEMENTAÇÃO | 49 |
| 4.8 UTILIZAÇÃO E FUNCIONAMENTO | 53 |
| 4.8.1 Estrutura de distribuição da aplicação | 53 |
| 4.8.2 Funcionamento | 55 |

| | |
|---|-----------|
| 5 CONSIDERAÇÕES FINAIS | 65 |
| 5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS | 65 |

1. INTRODUÇÃO

Nos dias atuais as atividades ligadas ao setor do agronegócio crescem anualmente, Papp e Chiara (2016), observam que o setor do agronegócio foi o único que não parou de crescer com a crise de 2015. Naquele cenário, a produção de soja, principal cultivar da agricultura brasileira, alcançou aproximadamente 100 milhões de toneladas. Com esses resultados e o crescimento da produtividade estimados para os próximos anos, o Brasil se tornou um dos principais exportadores de grãos no mundo, caminhando para ultrapassar a produção dos Estados Unidos. Tudo isso se tornou possível pelo investimento que os agricultores têm feito em tecnologias de ponta e agricultura de precisão, driblando os gargalos da economia e tornando-os competitivos em um cenário internacional, relatam os autores.

As propriedades agrícolas, assim como qualquer outro empreendimento necessitam atender as demandas de qualidade, competência e produtividade exigidas pela sociedade. Para isto, os gestores agrícolas têm investido em novas tecnologias visando alcançar patamares mais elevados sem perder a qualidade de seus produtos. Desta forma, alcançam as metas e satisfazem seus clientes.

Tendo isso em mente, o presente trabalho visa apresentar um protótipo de aplicação que permita os agricultores com poucos cliques selecionarem dados de espécies de cultivares, indicarem o tamanho de suas áreas de plantio e realizarem orçamentos de insumos. Com base nesses dados e através de uma rede de agropecuárias parceiras cadastradas no sistema, será possível identificar onde estão os insumos necessários e, ao mesmo tempo saber quais agropecuárias possuem o valor mais atrativo para a compra dos mesmos. Dessa forma, o agricultor poderá ter acesso a um histórico de safras e orçamentos dos produtos necessários para as suas lavouras, tornando-se possível desta forma fazer análises rápidas para saber quais características contribuíram para a maior produtividade sem gastos desnecessários.

1.1 PROBLEMA DE PESQUISA

Qual seria a forma mais eficiente para resolver os problemas de gestão orçamentária de lavouras no agronegócio utilizando a tecnologia, e consequentemente contribuir para uma maior integração entre as agropecuárias e os agricultores?

1.2 OBJETIVOS

Na presente seção serão expostos em forma de tópicos os objetivos que o trabalho visa alcançar.

1.2.1 Geral

- Desenvolver protótipo de aplicativo que auxilie o produtor agrícola na elaboração de orçamento de suas lavouras.

1.2.2 Específicos

- Identificar oportunidades para o uso de tecnologias no campo;
- Realizar análise de requisitos do protótipo baseado nas oportunidades identificadas;
- Aplicar modelo de avaliação heurística de usabilidade;
- Implementar protótipo.

1.3 JUSTIFICATIVA

No Brasil, o agronegócio vem crescendo de forma significativa representando uma das principais atividades do país, contando com clima favorável, alta luminosidade e abundância de água, o desenvolvimento de tecnologias para dinamizar as atividades no campo também tem tido grande crescimento.

Segundo Amcham (2018), as atividades agrícolas são responsáveis por 21% do PIB brasileiro, contribuindo com metade das exportações e sendo responsável por 1/3 dos empregos no país.

Com o avanço da tecnologia, a área tem recebido grandes inovações, os produtores têm tido mais contato com dispositivos tecnológicos e diariamente surgem novas ferramentas para facilitar as atividades no campo. Atualmente, as propriedades agrícolas, sejam elas de pequeno ou grande porte, possuem muitos ativos a serem administrados, ou até mesmo, investimentos e despesas que geram certo trabalho e que muitas vezes não são gerenciadas de forma correta, acarretando gastos desnecessários que acabam causando a diminuição dos

lucros. Os lucros de uma propriedade agrícola por sua vez, estão diretamente ligados a rentabilidade das atividades sendo desempenhadas, quanto mais controle o produtor possuir sobre as atividades, maior será o retorno financeiro das mesmas.

Para Amaral (2017), com o aumento da demanda dos produtos agrícolas e, o avanço do crescimento populacional econômico expressivo, a agricultura tem exercido alta pressão sobre os recursos naturais do planeta, que servem de matéria prima para a atividade. Consequentemente, é preciso ter-se maior oferta de produtos e, ao mesmo tempo limitar os impactos ambientais causados pelo agronegócio. Nesse cenário, a tecnologia tem auxiliado o agricultor no desenvolvimento sustentável da atividade, impactando em uma melhor produtividade e na diminuição do consumo de recursos naturais.

Segundo Lamas (2017), o censo agropecuário do IBGE evidenciou que a tecnologia foi responsável pelo crescimento de quase 70% na produção de grãos em 2006, enquanto em 1996 esse valor era apenas de 50%. Esses dados mostram a importância do uso de novas tecnologias no campo, melhorando os processos e a produtividade da agricultura, relata o autor.

Por tudo isso, justifica-se este projeto, visto que o desenvolvimento de um protótipo de aplicação poderia resolver alguns desses problemas, facilitando e organizando os gastos realizados pelos agricultores em suas cultivares ao longo das safras. Espera-se um maior controle sob as atividades do campo, melhorando a tomada de decisões e a comunicação entre os agricultores e agropecuaristas, aumentando a eficiência no mercado.

2. REVISÃO DA LITERATURA

No presente capítulo é realizada uma abordagem das tecnologias usadas no desenvolvimento do trabalho. A linguagem de programação Javascript possui um enfoque muito grande, levando em consideração a importância do seu uso nos dias atuais. Qualquer aplicação *web* por menor que seja, não dispensa o uso de Javascript, causando o surgimento de novos *frameworks* e métodos diariamente, aumentando ainda mais sua abrangência.

2.1 BANCO DE DADOS

Segundo Rob e Coronel (2011) os benefícios que os bancos de dados relacionais trazem, são: A capacidade de armazenamento, alteração e acesso de dados de forma rápida; A facilidade de implantação e operação e o baixo custo de implementação.

Mas isso nem sempre foi assim, até meados da década de 70, os bancos de dados armazenavam seus registros de forma hierárquica, tornando as aplicações lentas e a navegação péssima. Os autores também afirmam que antes de um projeto de banco de dados ser implementado, era necessário ter uma noção completa do que os clientes queriam fazer, incrementar ou modificar a estrutura futuramente tornar-se ia um processo caro e demorado.

Conforme Rob e Coronel (2011), as informações são o resultado do processamento dos dados para revelar seu significado. O processamento pode ser simples, apenas organizando os dados, é possível identificar padrões, e, posteriormente fazer análises estatísticas. Sem processamento, as informações não possuem significados, tornando impossível uma tomada de decisões.

Na Visão de Rob e Coronel (2011, p.4), “Por exemplo, uma leitura de temperatura média de 105° não tem muito significado, a menos que saibamos seu contexto: está em graus *Fahrenheit* ou *Celsius*?”.

As informações servem de fundamento para tomadas de decisões mais precisas. Por exemplo, reunindo os dados de uma entrevista é possível saber quais são os pontos fortes e fracos de um serviço ou produto, dizem os autores.

Na atual “era da informação”, a produção de informações precisas, relevantes e rápidas é a chave para uma boa tomada de decisões. Por sua vez, uma boa tomada de decisões é a chave para a sobrevivência comercial no mercado global. Dizem que estamos entrando na “era do conhecimento”. Os dados são o fundamento das informações, que é a base do conhecimento[...]. (ROB; CORONEL, 2011, p.7).

Segundo os autores o conhecimento é capaz de implicar na familiarização, consciência e compreensão das informações conforme se aplicam em um ambiente, onde a característica fundamental do conhecimento é que o “novo” pode ser obtido a partir do “antigo”.

2.1.1 Tipos de banco de dados

Segundo Rob e Coronel (2011), existem dois tipos de banco de dados que podem ser gerenciados, que são conhecidos como monousuário (um único usuário) e multiusuário (vários usuários). Os bancos monousuários oferecem suporte a apenas um usuário por vez. Em outras palavras, enquanto o usuário A estiver utilizando o banco os usuários B e C devem esperar até que o usuário A termine suas operações.

Um banco de dados monousuário executado em um computador pessoal recebe a nomenclatura de “banco de dados de *desktop*”. Sendo assim, o banco de dados multiusuário que oferece suporte a vários usuários simultaneamente, é chamado de “banco de dados de grupo de trabalho”, quando associado a um departamento. Mas, quando associado a uma organização inteira que engloba vários departamentos, o banco de dados é chamado de “banco de dados empresarial”, citam os autores.

Rob e Coronel (2011) também sugerem que a localização do banco de dados pode definir qual tipo de banco ele é. Como exemplo, podemos citar o “banco de dados centralizado”, que define o banco de dados que está localizado em apenas um local físico. Já o banco de dados que está separado em vários locais físicos é chamado de “banco de dados distribuído”.

Atualmente, o modo mais popular de classificação baseia-se em como os bancos de dados serão utilizados e na sensibilidade ao tempo das informações nele coletadas. Por exemplo, transações como vendas, pagamentos e aquisições de suprimentos de produtos refletem operações diárias e fundamentais. (ROB; CORONEL, 2011, p.9).

Para os autores, esses tipos de transações precisam ser registrados de modo imediato e preciso. Um banco de dados projetado com intuito de oferecer suporte às operações diárias de uma empresa, é conhecido como “banco de dados operacional”, onde as vezes pode ser chamado também de “banco de dados transacional” ou “banco de dados de produção”.

Por outro lado, os conhecidos *data warehouses* (armazéns de dados em português) possuem foco no armazenamento de dados utilizados para gerar informações, que serão utilizados para a tomada de decisões, afirmam Rob e Coronel (2011).

Os bancos de dados também podem ser classificados de modo a refletir o grau de seus dados, ou seja, se eles estão sendo armazenados de forma estruturada ou não. Os dados não estruturados são aqueles armazenados em seu estado natural, sem nenhum tipo de tratamento após sua coleta. Sendo assim, possuem um formato que não permite seu processamento e a extração de conhecimento, sugerem os autores.

Dados estruturados segundo Rob e Coronel (2011), são os dados obtidos a partir do pré-processamento das informações, tornando possível a extração de conhecimento. Além de servirem para processamentos futuros, os dados estruturados facilitam seu armazenamento e utilização.

Os autores também citam a existência de dados semiestruturados. Esse tipo de dado já foi parcialmente processado. Como exemplo, podemos citar os dados apresentados em uma página *web*, é muito comum que eles sejam apresentados em um formato pré-organizado tornando possível a transmissão de informações.

2.1.2 PostgreSQL

Para Carvalho (2017), PostgreSQL é um sistema gerenciador de banco de dados muito poderoso, de código fonte aberto para a comunidade de desenvolvedores. Durante um longo tempo foi discriminado no mundo dos bancos de dados relacionais, mas recentemente ganhou um aumento de popularidade através da migração de usuários de outros bancos, em busca de um sistema com mais confiabilidade, e melhores recursos de consulta de dados, ou simplesmente pelo desejo de algo mais fácil de aprender e utilizar.

Com desenvolvimento ativo a mais de 15 anos, a arquitetura vem ganhando boa reputação através da integridade de dados, conformidade, padrões e confiabilidade. Essas características vem tornando a ferramenta algo fácil de usar, conservando a velocidade e segurança, diz Carvalho (2017).

O autor cita que a facilidade de uso está relacionada aos comandos SQL do PostgreSQL. As ferramentas de linhas de comando aceitam os mesmos argumentos, e os tipos de dados não possuem nenhuma espécie de truncamento silencioso ou comportamento estranho.

Sua segurança está diretamente relacionada ao encadeamento transacional durante uma operação, que incluem mudanças estruturais destrutivas. Sendo assim, o usuário pode realizar qualquer operação dentro de uma transação, até mesmo uma exclusão de dados ou uma alteração na estrutura de tabelas, onde as mesmas, podem ser revertidas facilmente, se o

usuário desejar, basta reverter a transação, menciona Carvalho (2017). Outro ponto citado pelo autor, foi a facilidade na realização de *backups* do sistema ao clonar um banco de dados.

Segundo Carvalho (2017), o segredo da velocidade do PostgreSQL está no sistema estratégico de indexação e otimização de consulta, para que seja possível trabalhar sem exigir muito processamento de dados.

PostgreSQL é um banco de dados relacional de nível corporativo, possuindo as funcionalidades mais sofisticadas, como por exemplo: controle de concorrência multiversionada ou, recuperação em um ponto no tempo. Com todas essas características, é possível criar qualquer aplicação, independente de tamanho ou complexidade, relata o autor.

2.2 JAVASCRIPT

Conforme Flanagan (2013), Javascript é uma linguagem de programação criada para *web*, atualmente a maioria dos sites usam essa ferramenta e todos os navegadores modernos são compatíveis. Javascript também pode ser utilizado em videogames, computadores, celulares e tablets que tenham a capacidade de interpretar as linhas de comando da linguagem. Com essa grande gama de aparelhos e ferramentas utilizando de seus recursos, o Javascript é a linguagem mais onipresente da história.

O desenvolvimento de aplicações *web* possui algumas ferramentas básicas onde as mesmas são indispensáveis, Javascript faz parte de uma tríade de três ferramentas que tornam as aplicações fluídas, responsivas e personalizadas, são elas: HTML, CSS e Javascript, diz Flanagan (2013).

É uma linguagem de alto nível, dinâmica, não tipada e interpretada. Sua estrutura pode ser usada com estilos de programação como a orientação a objetos, ou, estilos funcionais, sua sintaxe deriva-se da linguagem Java, das funções de primeira classe de *scheme* e da herança baseada em protótipos de *self*, cita o autor.

Segundo Flanagan (2013), o nome Javascript pode ser enganoso, pois apesar de sua sintaxe ser muito semelhante ao Java, as duas linguagens são completamente diferentes. O Javascript perdeu suas características de linguagem *script* a muito tempo, e atualmente a linguagem vem se tornando cada vez mais robusta, eficiente e de uso geral para todas as aplicações.

O autor relata que o Javascript foi criado na NetScape na fase inicial da internet e, basicamente “Javascript” é nome da marca registrada, que foi licenciada pela empresa Sun Microsystems, atualmente conhecida como Oracle. Ela foi usada para descrever a

implementação da linguagem pelo Netscape, que, nos dias atuais, é chamado de Mozilla Firefox. A linguagem NetScape foi enviada para padronização na ECMA (*European Computer Manufacture's Association*) e, devido a questões de marca registrada a versão padronizada manteve o nome de “ECMAScript”.

Flanagan (2013) ressalta que para toda linguagem de programação ser útil, ela deve possuir uma plataforma ou biblioteca padrão, ou até mesmo, uma *API* de funções para fazer operações básicas como saída e entrada de dados. O Javascript baseia-se na definição de uma *API* mínima para trabalhar com textos, *arrays*, datas e expressões regulares, mas não inclui ferramentas que fazem entrada ou saída. A entrada e saída de dados muitas vezes é feita através dos ambientes onde o Javascript está sendo utilizado, normalmente esse ambiente é um navegador de internet.

2.2.1 Javascript, estrutura léxica

Segundo Flanagan (2013), a estrutura léxica do Javascript é o conjunto de regras básicas que definem o modo de escrita dos programas. Essa estrutura de nível baixo especifica como as variáveis e seus nomes devem ser declarados, bem como, os caracteres delimitadores que fazem a separação de instruções.

O autor cita que os programas Javascript são escritos usando conjuntos de caracteres *unicode*, ou seja, ele utiliza os caracteres da tabela ASCII e Latin-1, que suportam praticamente qualquer tipo de idioma escrito nos dias atuais. Além da vasta quantidade de caracteres aceitos, a linguagem também faz a diferenciação de caracteres maiúsculos e minúsculos dando aos programadores mais opções de nomenclaturas para suas variáveis.

Javascript é uma linguagem que diferencia letras maiúsculas de minúsculas. Isso significa que palavras-chave, variáveis, nomes de função e outros identificadores da linguagem sempre devem ser digitados com a composição compatível de letras. A palavra-chave *while*, por exemplo, deve ser digitada como “*while*” e não como “*While*” ou “*WHILE*”. Da mesma forma, *online*, *Online*, *OnLine* e *ONLINE* são quatro nomes de variável distintos. (FLANAGAN, 2013, p.37).

Flanagan (2013, p.37) sugere “[...] entretanto, que HTML não diferencia letras maiúsculas e minúsculas (embora a XHTML diferencie). Por causa de sua forte associação com Javascript do lado do cliente, essa diferença pode ser confusa.”.

O autor também relata que muitos objetos e propriedades Javascript no lado do cliente possuem os mesmos nomes de *tags* e atributos HTML, onde normalmente são digitadas com letras maiúsculas e minúsculas, já no Javascript normalmente devem ser minúsculas.

2.3 NODE

Powers (2012) diz que a tecnologia que evolui em torno do Node ainda é jovem e com muitos recursos interessantes, ao mesmo tempo, ela consegue atingir um nível de maturidade alto, garantindo que todo o tempo investido na mesma trará bons resultados. A instalação do Node é muito fácil, principalmente no Windows, e para o autora existem duas coisas importantes que devem ser levadas em consideração quando o assunto é Node, são elas: O Node é baseado em Javascript; O Node não atua com o mesmo propósito do Javascript comum, seu objetivo é trabalhar no lado do servidor, onde algumas funcionalidades do *client side* foram deixadas de lado e outras funcionalidades foram implementadas para melhorar as operações realizadas no *server side*, cita a autora.

Powers (2012) relata que se explorado o código fonte do Node, é possível encontrar trechos de código do v8 da Google, mecanismo Javascript que está presente no núcleo do navegador Google Chrome. Portanto, a principal vantagem do Node é permitir aos desenvolvedores a criação de aplicações utilizando apenas da implementação do Javascript.

Sua arquitetura é projetada para suportar grandes quantidades de entradas e saídas de dados, mas o mais importante, usando Node os programadores não precisam se preocupar com o bloqueio do processamento de uma aplicação pois, todas as suas funcionalidades são assíncronas por padrão. Sendo assim, não há necessidade de trabalhar com *threads*, porque o Node tem sua implementação em um único *thread*, cita o autor.

Powers (2012) informa que Node é escrito em uma linguagem na qual muitos desenvolvedores *web* tradicionais estão familiarizados, o Javascript. Essa familiaridade com a linguagem torna o aprendizado muito fácil e rápido.

2.4 REACT

Para Banks e Porcello (2017), React é uma biblioteca Javascript muito popular usada para criar interfaces de aplicações *web*, construída pelo Facebook para aumentar a produtividade na criação de aplicações para internet.

Quando a biblioteca foi lançada em 2013, inicialmente foi alvo de ceticismo, acreditava-se que a forma de trabalho *framework* saia dos padrões conhecidos naquela época, onde muitas perguntas surgiram sobre seu funcionamento na prática. Com isso, a equipe do Facebook escreveu um artigo chamado “*Why React?*” (Por que React em português), onde eles encorajavam os desenvolvedores a trabalharem com *framework*, citam os autores.

Banks e Porcello (2017) relatam que no início o React era pequeno e utilizado apenas em algumas partes do projeto, pois, não fornecia todas as ferramentas necessárias que um *framework* Javascript tradicional oferece, ou seja, existia uma grande dependência dos desenvolvedores decidirem em qual ecossistema o React seria utilizado.

Além disso, os autores citam que o *framework* chegou em um momento caótico da história do Javascript. O ECMA não costumava liberar atualizações com frequência podendo demorar até 10 anos para lançar uma nova especificação, algo que era bom para os desenvolvedores, pois não precisavam aprender novas sintaxes constantemente. Em 2015, novos recursos começaram a ser lançados, mudando totalmente o cenário do Javascript, fazendo com que novas especificações fossem lançadas anualmente.

Com as mudanças que surgem em nível de linguagem, existe também muita motivação para a programação funcional no Javascript. A linguagem não é necessariamente funcional, mas as técnicas funcionais podem ser utilizadas no código, relatam Banks e Porcello. O React enfatiza esse tipo de programação, trazendo benefícios de desempenho e facilidade nos testes, citam os autores.

2.5 REACT NATIVE

Segundo Eisenman (2016), o React Native é um *framework* feito em Javascript para a criação de aplicativos nas maiores plataformas *mobile* atuais, como IOS e Android. O autor afirma que o React Native é baseado em React, um *framework* desenvolvido pelo Facebook para a construção de interfaces *web*.

Na visão de Eisenman (2016), construir aplicações utilizando o React Native tornam a experiência do usuário muito intuitiva, já que o *framework* imita as ações e comportamentos dos componentes nativos dos dispositivos. O desempenho das aplicações criadas em React Native também é semelhante ao desempenho de aplicações nativas, e tudo isso pode ser feito para várias plataformas utilizando o mesmo código.

Eisenman (2016) cita que aplicações construídas com React Native funcionam da mesma forma que com React, onde os códigos são escritos utilizando uma mistura de

Javascript e XML, conhecidos com JSX. Então por baixo dos panos, o React Native cria uma ponte com as *APIs* de renderização dos dispositivos, no caso do IOS, será em Objective-C, já para o Android, será em Java.

2.5.1 Vantagens

Para Eisenman (2016) o fato de o React Native reenderizar telas usando as *APIs* padrões da plataforma de *host* permite que ele se destaque da maioria dos métodos existentes de desenvolvimento de aplicativos de multiplataforma, como o Cordova ou o Ionic. Métodos existentes para escrever aplicativos móveis usando combinações de Javascript, HTML e CSS, gerando as conhecidas *web views*. Nesses casos há uma grande perda de performance dos dispositivos, já que as *web views* não são capazes de criar elementos nativos das plataformas. React Native também consegue ter acesso a recursos dos dispositivos, como câmeras e localizações, sem o uso de *plug-ins* e outras ferramentas.

O autor também relata que o ciclo de atualizações do React Native continua o mesmo do React. Quando o desenvolvedor faz alguma alteração no código fonte o React Native faz o recarregamento das suas *views*.

2.5.2 Reutilização do código fonte

Eisenman (2016) afirma que trabalhar com React Native pode ser algo complicado no começo, pois a ferramenta tem muitos recursos e dependências para a construção de uma aplicação *mobile*. Com o passar do tempo, a aquisição de experiência no desenvolvimento torna as atividades mais rápidas e práticas, onde a equipe de desenvolvimento consegue compartilhar conhecimento e recursos de forma mais iterativa e efetiva.

O React Native permite que os desenvolvedores façam o uso quase por completo do mesmo código para plataformas diferentes, como exemplo desta afirmação o autor cita o aplicativo de propagandas do Facebook, que tem 87% do seu código fonte sendo utilizado no Android e IOS sem nenhuma alteração, diz Eisenman (2016).

2.5.3 Riscos e desvantagens do *framework*

O autor cita que trabalhar com React Native assim como qualquer outro tipo de ferramenta, possui suas desvantagens. É necessário ter conhecimento do cenário onde o

framework será aplicado, para saber se ele trará realmente o ajuste necessário tanto no desenvolvimento da aplicação como na aplicação em si.

O maior risco provavelmente está diretamente relacionado com a maturidade do React Native, as duas principais plataformas de celulares, Android e IOS, lançaram suporte a esse *framework* apenas em 2015, ou seja, algo muito recente. A documentação certamente é um ponto que também precisa ser melhorada e evoluída, cita o Eisenman.

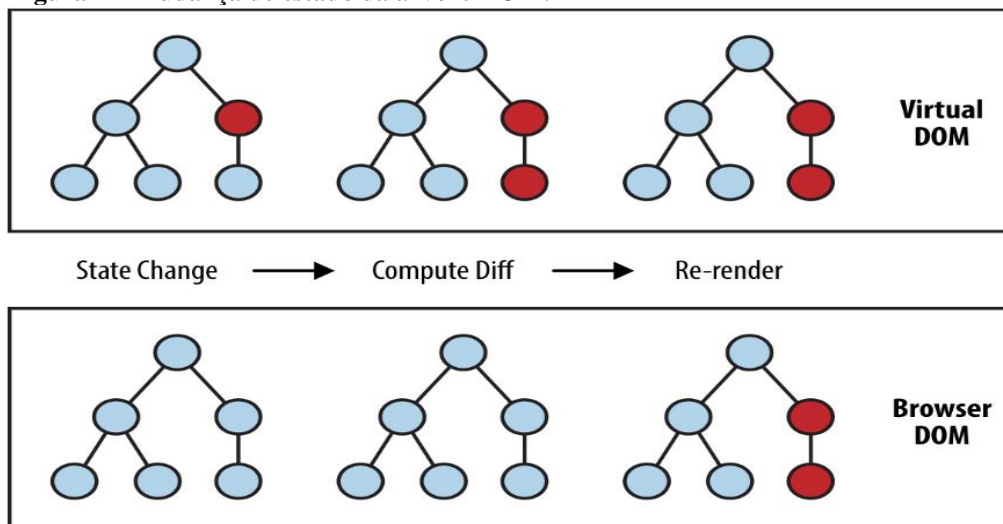
2.5.4 Trabalhando com React Native

Eisenman (2016) afirma que a ideia de desenvolver aplicações móveis usando a linguagem Javascript pode ser um pouco estranha. Como é possível usar React em um ambiente *mobile*? Vamos entender as técnicas usadas pelo *framework*, e como o mesmo trabalha com os recursos da árvore DOM (*Document Object Model*, em português, modelo de documentação por objetos).

No React a árvore DOM atua como uma ligação entre a descrição do que o desenvolvedor deve fazer, e o trabalho feito para que o aplicativo criado seja carregado de forma correta na página. Ao invés de reenderizar as alterações feitas diretamente na tela, o React calcula as alterações necessárias usando uma versão em memória da árvore DOM, dessa forma é possível fazer a renderização somente dos elementos que foram alterados, trazendo benefícios de desempenho, relata o autor.

A Figura 1 mostra como isso funciona na árvore DOM e no próprio navegador onde a aplicação será reenderizada.

Figura 1 – Mudança de estado da árvore DOM.

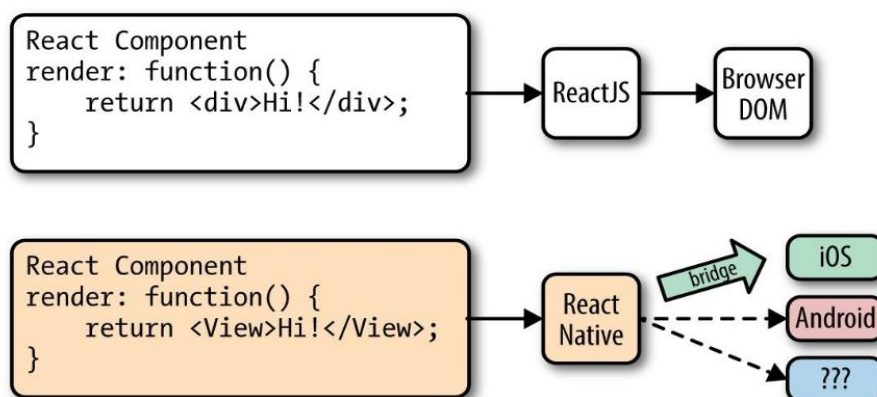


Fonte: Eisenman (2016, p.26)

Segundo Eisenman (2016), o React Native não funciona da mesma forma que o React, ao invés de fazer a renderização das aplicações na árvore DOM, o React Native invoca *APIs* do Objective-C para reenderizar no IOS, e *APIs* Java para reenderizar no Android.

A figura 2 a seguir, mostra as diferenças entre os dois *frameworks* no momento de uma renderização em tela.

Figura 2 - Forma de trabalho do React.



Fonte: Eisenman (2016, p.27)

Isso tudo é possível porque a *bridge* prove ao React Native uma interface de elementos nativos das plataformas IOS e Android. Os componentes de marcação usados no React Native descrevem como os elementos em tela devem ser, seus tamanhos, formas, cores e comportamentos.

2.5.5 Criando Componentes com React Native

Eisenman (2016) diz que todo o código escrito em React está nos componentes criados anteriormente. No React Native os componentes também são os mesmos, apenas com algumas diferenças de renderização e estilização. No React, o código escrito é rerenderizado para elementos HTML, com o React Native, todos os elementos são rerenderizado para componentes específicos da plataforma.

2.5.6 Estilizando componentes nativos no React Native

Eisenman (2016) relata que para estilizar componentes no React Native é possível utilizar as propriedades do CSS, como normalmente é usado em qualquer outra aplicação construída com HTML.

No React Native a forma como o código CSS é escrito não é afetada, plataformas não *web* possuem uma ampla variedade de abordagens quando o assunto é estilização de *layouts*, mas com React Native isso não acontece, felizmente é possível utilizar uma abordagem padronizada para a estilização das aplicações construídas com esse *framework*, diz Eisenman (2016).

Segundo o autor, nem todas as funcionalidades do CSS foram implementadas no React Native, isso acontece porque o *framework* trabalha apenas com *flexbox layout*, e se concentra na simplicidade ao invés de implementar todas as gamas de regras CSS.

2.5.7 Host Platform APIs

Segundo Eisenman (2016) a maior diferença entre React e React Native está na forma como pensamos sobre *APIs* para plataformas de *host*. Com React Native as *APIs* desempenham um papel muito forte na criação de aplicações trazendo uma experiência natural e excelente, diferentemente de aplicações *web*, onde existe uma gama muito grande de “padrões” que as vezes não trazem ao usuário uma experiência agradável e fluida. As *APIs* para dispositivos móveis incluem todo tipo de serviço para *hardware* e *software*, por padrão o React Native possui muitos recursos usados no IOS e Android suportando qualquer *API* assíncrona nativa.

2.6 USER EXPERIENCE DESIGN

Segundo Pereira (2018), a experiência do usuário engloba todos os aspectos de interação que as pessoas têm, seja com uma marca, serviço e principalmente com suas ferramentas digitais, sendo elas sites, aplicativos e softwares. Toda essa experiência inicia-se desde o primeiro contato com a marca ou produto, até o momento que isso é utilizado ou consumido de fato.

As grandes marcas já perceberam o quão é importante investir em experiência do usuário com o objetivo de melhorar a percepção sem esquecer do retorno financeiro. Uma pesquisa de satisfação aponta que quase 90% dos usuários alegam ter menos chance de retornar a um site depois de uma experiência negativa com o mesmo. Mesmo porque um novo site está sempre a um clique de distância. (PEREIRA, 2018, p.8).

Se preocupar com a experiência do usuário começou a fazer sentido quando construir bons produtos digitais tornou-se algo muito além de fazer um produto que seja somente agradável aos olhos. No início, o mercado era amador em relação a experiência de uso das aplicações, onde, muitas vezes, apenas uma pessoa ficava responsável por pensar em um produto do início ao fim. Não existia tanta complexidade e, as empresas ainda não estavam preocupadas em fornecerem um ambiente agradável para as pessoas, relata o autor.

Com o passar do tempo, o cenário mudou drasticamente devido a tamanha complexidade que existe no mercado atualmente, resultado do grande volume de informação que é gerado todos os dias, esclarece Pereira (2018).

A tecnologia evolui em uma velocidade extremamente assustadora, dessa forma, é mais difícil criar um produto que seja capaz de atingir todos os objetivos e, com isso, surgiu a necessidade de integrar vários profissionais de várias áreas juntos, para transformar o caos em algo plausível para as pessoas envolvidas.

2.6.1 O que é *UX Design*?

Pereira (2018) diz que, *UX* é o nível de satisfação que usuário tem ao usar um serviço ou produto, independentemente de ser ele físico ou digital. Essa satisfação manifesta-se em todos os objetos que usamos em nosso dia a dia. A interação que as pessoas têm com produtos digitais é ampla e só vem aumentando com o passar dos anos, nos dias atuais usamos computadores, celulares, tablets e videogames com grande frequência e é responsabilidade do profissional de *UX design* fazer interfaces que tragam uma boa usabilidade para os usuários.

É muito importante entender que para criar um projeto é necessário contar com o lado analítico e o lado criativo do *UX designer*. Referindo-se ao lado analítico, este está mais relacionado com as tarefas da parte estrutural e funcional do produto ou serviço, quanto ao lado criativo, está diretamente relacionado ao jeito elegante e mais interessante de resolver o problema em questão, argumenta Pereira (2018).

2.6.2 Arquitetura de informação

Para Pereira (2018), esse é um dos perfis mais importante para o *UX designer*. É nesse ponto que o produto estrutural será desenhado e ganhará sentido, as informações devem ser organizadas conforme o propósito do projeto, essa organização deve fazer sentido para o público que usará o produto. É importante lembrar que a arquitetura da informação vai muito

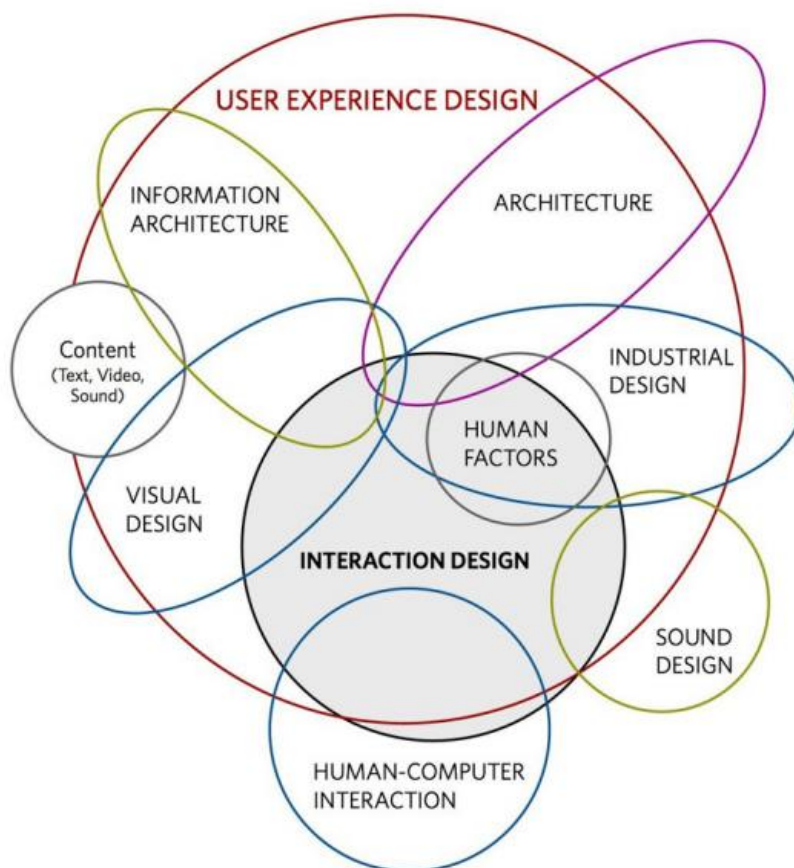
além de fazer *wireframes* e *sitemaps*, onde normalmente as pessoas não se aprofundam o suficiente e colocam em prática apenas uma parte dessa disciplina.

2.6.3 Principais disciplinas do *UX Design*

Dan Saffer, escritor do livro *Designing for Interaction* (2006) construiu um diagrama que exhibe de forma detalhada todas interseções e nomenclaturas que envolvem o *UX design*, relata Pereira (2018).

Na figura 3 a seguir, o maior círculo chamado “*User Experience Design*” engloba uma série de outras disciplinas, desde arquitetura da informação até *design* industrial, e passa por áreas que são menos comuns no dia a dia de quem trabalha com projetos digitais como o “*Human Factors*”. Também é possível perceber claramente que para a construção de uma boa experiência de usuário, existem outros vários fatores envolvidos, onde pode-se mencionar inclusive, os efeitos sonoros que acontecem no momento que uma nova mensagem é recebida, seja em uma rede social, ou na caixa de e-mail.

Figura 3 – Diagrama de interseções e nomenclaturas do *UX Design*.



Fonte: Pereira (2018, p.21)

Pereira (2018) faz o seguinte questionamento, por que as pessoas não param de usar serviços de *streaming* como Spotify ou Netflix, ou por que não abrem mão dos videogames já que gastam horas em frente à TV? A resposta é simples. Elas têm uma boa experiência de uso com esses serviços e produtos, fruto do bom trabalho do *UX designer*.

2.6.4 Usabilidade em produtos digitais

Para Pereira (2018), usabilidade é a facilidade e a simplicidade de uso que uma interface apresenta. Atualmente, o fator da usabilidade é visto como decisivo para o sucesso ou fracasso de um produto digital. Mas não é tão simples chegar em uma solução simples. O processo para criação de produtos que tragam uma boa usabilidade é muito trabalhoso. Normalmente, é necessário explorar muitas versões do *design* e discutir com outros membros do grupo.

“O processo de simplificar uma interface exige que você exponha o quanto antes sua solução para as pessoas que a usarão.”, diz Pereira (2018, p.139).

A informação deve ser exposta para os usuários em doses, empresas como a Apple e a Google são destaques em fazer isso, dessa forma idosos e crianças conseguem usar seus produtos sem dificuldade. Deve existir uma hierarquia de informações no produto, o *UX designer* deve sempre se questionar sobre o que ele deseja que o usuário veja por primeiro depois, quais benefícios devem ser expostos e em seguida, o que o produto possui de diferencial; Por fim, qual é o propósito do produto e o que acontece quando o usuário interage com determinada informação dele. Dessa forma, é possível deixar claro ao usuário o que o produto irá fazer e quais passos ele poderá tomar, relata o autor.

2.7 HEURISTICA DE USABILIDADE

Para Pereira (2018) a avaliação heurística de usabilidade é um método que torna possível a análise da eficiência em princípios de usabilidade em uma interface digital. É de suma importância dominar e conhecer esse método para servir como um *checklist* do que irá funcionar e o que irá falhar em um produto digital.

Segundo Nielsen (1994), a avaliação heurística é composta por 10 etapas descritas no quadro a seguir;

Quadro 1 – Heurísticas de usabilidade.

| Título | Descrição |
|---|---|
| Status do sistema | O sistema deve sempre manter os usuários informados sobre o que está acontecendo, dando o <i>feedback</i> apropriado no momento necessário; |
| Relacionar o sistema ao mundo real | O sistema deve sempre falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário. Seguir as convenções do mundo real fazendo as informações aparecerem em uma ordem natural e lógica; |
| Controle e liberdade do usuário | Pessoas normalmente cometem erros ao usar um sistema, então, elas precisam de um botão de “saída de emergência” para desfazer o erro cometido; |
| Consistência e padrões | Os usuários não devem ficar confusos com palavras, situações ou ações que significam a mesma coisa; |
| Prevenção de erro | Ainda melhor do que boas mensagens de erro é evitar a ocorrência de um problema. Eliminar condições propensas a erros confirmando que um usuário não irá comprometer o fluxo do sistema; |
| Reconhecimento em vez de recordação | É muito importante minimizar a carga de memória do usuário, tornando objetos, ações e opções visíveis sem a necessidade de o usuário ter que se lembrar das partes anteriores do diálogo; |
| Flexibilidade e eficiência de uso | Permitir que os usuários se adaptem com ações frequentes; |
| <i>Design</i> estético e minimalista | As interfaces do sistema não devem conter informações irrelevantes ou raramente necessárias, pois as mesmas concorrem com informações relevantes e dificultam a visualização; |
| Ajudar os usuários a reconhecerem, diagnosticarem e recuperarem erros | Mensagens de erros devem ser expressadas precisamente e sem códigos, indicando o problema e uma possível solução; |
| Documentação de ajuda | Se for possível, o programa deve ser usado sem documentação, mas pode ser necessário, então, qualquer informação valiosa deve ser encontrada de forma fácil e concisa. |

Fonte: Elaborado a partir de Nielsen (1994)

As heurísticas foram desenvolvidas originalmente para avaliação heurística em colaboração com Rolf Molich em 1990. Desde então, foram refinadas com base em uma análise fatorial de 249 problemas de usabilidade para derivar um conjunto de heurísticas com máximo poder explicativo, resultando nesse conjunto revisado de heurísticas, explica Nielsen (1994).

3. METODOLOGIA

Este trabalho de conclusão de curso caracteriza-se como pesquisa aplicada, descritiva, pois seu objetivo foi desenvolver um protótipo de sistema para gestão de áreas de plantio, utilizando o georreferenciamento. Buscou-se responder aos seguintes questionamentos de pesquisa: Qual seria a forma mais fácil e acessível para resolver os problemas de gestão de lavouras nas propriedades agrícolas usando tecnologias móveis? É possível aumentar a rentabilidade e a competitividade no mercado com o uso das mesmas? Do ponto de vista da usabilidade, de qual forma seria possível desenvolver uma aplicação de fácil entendimento para agricultores que não possuem muita afinidade com tecnologia?

Referindo-se aos procedimentos utilizados, para levantamento de informações, fez-se o uso da pesquisa qualitativa, onde foram analisados os fluxos dos principais serviços realizados envolvendo a gestão de lavouras, bem como foi realizado questionamento através de entrevista junto a um agricultor e agropecuarista, para obter-se dados e regras de negócio envolvendo os gastos e investimentos realizados durante o período de plantio. Através dessa atividade foi possível determinar de qual forma o desenvolvimento do aplicativo deveria seguir.

Ao analisar o cenário atual das ferramentas que operam no mercado e apresentam algumas semelhanças com os objetivos deste projeto, houve um estudo sob a ferramenta Aegro, afim de identificar pontos positivos, negativos, divergências e espaços para possíveis melhorias.

Para o desenvolvimento do protótipo da aplicação, foi feito o uso do sistema operacional Ubuntu 18.04 LTS, no qual, facilita a configuração e o uso de ferramentas como o gerenciador de pacotes Yarn. Como editor de código, fez-se o uso do Microsoft Visual Studio Code 1.38, com ele, foi possível configurar *templates* de padrões de código Javascript. Para a construção da *API* a linguagem Node 10.16.3 foi utilizada. Quanto ao banco de dados utilizou-se o PostgreSQL 9.6 juntamente com o PgAdmin 3 para fazer o gerenciamento de dados. Na construção do *front-end* da aplicação, foi utilizada a biblioteca React 16.13.1, referindo-se ao aplicativo mobile fez-se o uso do React Native 0.63 com direcionamento de desenvolvimento para a plataforma Android 9.1.

3.1 ESTADO DA ARTE

Atualmente no mercado, existem várias aplicações com características que se assemelham as propostas desenvolvidas neste projeto. Para análise, a seguir, apontaremos alguns pontos positivos e negativos da ferramenta Aegro.

3.1.1 Aegro

O Aegro está presente em grande parte do território nacional, contando com mais 2 milhões de hectares mapeados e mais de 5 mil usuários mensalmente. Os valores da ferramenta variam, e são baseados na quantidade de hectares que o agricultor possui cadastrada. Dependendo da necessidade do agricultor, é possível contratar funções adicionais como a previsão do tempo por exemplo, todas essas características são consideradas ao gerar o valor da licença, (AEGRO, 2021).

Ao fazer a aquisição do sistema, o agricultor usará suas funcionalidades pelo computador, tablet ou celular, trazendo maior praticidade e mobilidade nas atividades diárias. O sistema também conta com uma versão de teste que dura 7 dias. Caso o agricultor faça a assinatura do software, poderá fazer a solicitação de treinamento gratuito oferecido pela empresa.

Uma das principais propostas da ferramenta é o gerenciamento inteligente de propriedades agrícolas, auxiliando o agricultor em suas atividades operacionais e financeiras.

A cada compra feita, o sistema contabiliza uma despesa no financeiro, e adiciona os produtos comprados ao estoque permitindo que o agricultor utilize os mesmos em suas atividades. Ao chegar no fim do período de cultivo, o agricultor pode informar ao sistema dados de sua colheita, dessa forma é feito o cruzamento de dados entre as despesas e os lucros informando ao agricultor a rentabilidade de sua lavoura.

A ferramenta é relativamente fácil de usar. Mas na tela de cadastros de despesas o agricultor pode acabar se perdendo pela grande quantidade de informações que o sistema mostra ao mesmo tempo.

Na figura 4, podemos ver a tela de cadastro de despesas, com o campo de seleção de produtos em foco. Há uma grande quantidade de informações, muitas delas de categorias distintas, e que devem ser informadas no formulário. Já no campo de seleção de produtos, podemos ver a disposição de alguns ícones para serem usados como filtros de busca, onde os

mesmos não possuem nenhuma nomenclatura detalhada para informar ao usuário qual ação ele está executando.

Figura 4 – Cadastro de despesas do Aegro.

Nova despesa ☐ Repetir

Informações gerais

Despesa ▼ Categoria * Insumos Agrícolas ×

Data de lançamento * 27/09/2019

Agrupadores

Fornecedor Cravil ×

Nota Fiscal 874361532323320015565212121

Descrição compra de insumos 17 / 300

Moeda Real - (R\$) ▼

Produtos e serviços

Item

Valor ☐ Meu catálogo

| | |
|------------------------------|------|
| 16-16-16 Trigo - YARAMILA | 0,00 |
| 2B633 PW Milho | 0,00 |

CAN R\$ 0,00 CHAR

Fonte: Acervo do autor (2021)

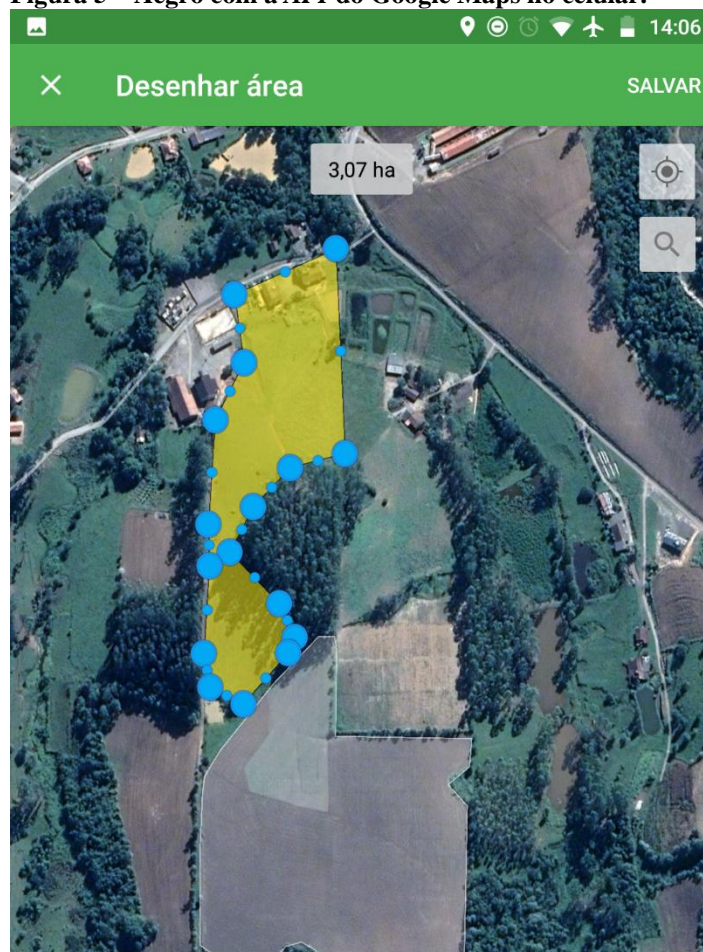
A ferramenta também permite que o agricultor cadastre seu patrimônio e despesas geradas com o combustível ou manutenções. Nessa área também é possível ter acesso as horas que cada equipamento trabalhou e vida útil da máquina.

Quanto ao cadastro de áreas, o Aegro possui integração com a *API* do Google Maps. Essa integração permite que o agricultor visualize suas áreas de plantio com informações detalhadas.

Ao utilizar o aplicativo móvel foi possível notar maior facilidade de uso, uma interface com menos informações e mais focada nos objetivos.

A figura 5 mostra a integração da ferramenta com o Google Maps nos dispositivos móveis.

Figura 5 – Aegro com a *API* do Google Maps no celular.



Fonte: Acervo do autor (2021)

O Aegro tem se mostrado uma ótima ferramenta para auxiliar as atividades do campo, e a gestão financeira interna da propriedade. Mas até o momento, o sistema não conta com funcionalidades que facilitam a orçamentação de lavouras por parte do agricultor, e ainda existe uma grande distância entre os agricultores e os agropecuaristas, distância essa, causadora de gastos desnecessários de insumos, tempo e principalmente dinheiro.

4. DESENVOLVIMENTO

A seguir, serão abordadas as funcionalidades que o sistema deve possuir e a forma como ele deverá funcionar. Para facilitar o entendimento, foram criadas seções com enfoque nas regras de negócio, requisitos funcionais, não funcionais, diagramas de caso de uso, diagramas de atividades e uma apresentação detalhada do desenvolvimento e uso do sistema. Como o sistema conta com algumas características de usabilidade, foi criada também uma análise das heurísticas de usabilidade presentes na aplicação.

4.1 VISÃO GERAL DO SISTEMA

O objetivo principal do sistema é melhorar a comunicação entre os agricultores e as agropecuárias, oferecendo ao agricultor uma visão mais detalhada sobre o mercado de produtos agrícolas.

O sistema é voltado para o agronegócio, seja em uma propriedade de pequeno, médio ou grande porte, podendo trazer melhorias na tomada de decisões da parte do agricultor na compra de seus produtos. Aos agropecuaristas, a aplicação poderá aumentar a competitividade no mercado, dessa forma as agropecuárias podem melhorar a qualidade de seus produtos e serviços.

O agricultor operará o sistema através de um aplicativo móvel específico, o agropecuarista utilizará uma página *web* para administração da oferta de produtos da agropecuária, já o responsável pelo sistema fará a administração da plataforma diretamente pelo banco de dados, ao menos nessa versão inicial do protótipo.

Neste momento, em sua primeira versão o sistema não poderá ser incorporado a um sistema gerenciador de estoques, site ou qualquer outra ferramenta que a agropecuária ou o agricultor possuam. O mesmo não fará nenhum tipo de notificação, como por exemplo indicar as agropecuárias que houve o cadastro de um novo orçamento em seu estabelecimento.

Para o desenvolvimento do sistema foram utilizadas tecnologias *web* atuais. A principal linguagem utilizada no projeto foi o Javascript, que atuou tanto na *API* com Node, quanto no *front-end* e *mobile*, com React e React Native. A facilidade na configuração e praticidade no desenvolvimento foram os principais motivos para essa escolha.

4.2 RESULTADOS OBTIDOS COM OS QUESTIONAMENTOS

Conforme apresentando na metodologia deste projeto, foi realizado entrevista com o agricultor e agropecuarista, para que fosse possível identificar algumas das dificuldades já mencionadas anteriormente.

A primeira entrevista foi realizada com um agricultor de Taió, no dia 18 de março de 2021. Ele relatou que a compra dos seus insumos não é uma tarefa tão simples quanto parece. Segundo o agricultor, essa atividade demanda um bom tempo de conversa, para saber quais os produtos são os mais adequados para sua cultura. Após a identificação dos produtos mais adequados, ainda é necessário fazer uma pesquisa de preço em outras agropecuárias, pois muitas vezes é possível encontrar os mesmos produtos com preços mais acessíveis, ou com recomendações de uso em menor quantidade, o que gera menos despesas e gastos desnecessários no manejo da cultura.

A segunda entrevista foi realizada no dia 21 de abril de 2021, com um representante de vendas externas e técnico agrônomo. Ele relatou que existe um grande retrabalho durante a orçamentação de uma lavoura, isso acontece porque muitas vezes o agricultor quer realizar uma comparação de preços entre os fornecedores, ou, saber características mais detalhadas dos produtos. Ao ter essas dúvidas, o agricultor contata o fornecedor, onde pode haver um grande período de conversa e negociação, sendo que em alguns casos, o agricultor acaba não comprando os produtos, causando um gasto de tempo, o que poderia ser evitado com a presença de um portfólio digital disponível para o agricultor. Outro ponto mencionando pelo vendedor, foi a vantagem competitiva que as agropecuárias poderiam ter ao ofertar produtos de boa qualidade com um preço atrativo, podendo ser um diferencial dos demais estabelecimentos.

4.3 REGRAS DE NEGÓCIO

As regras de negócio do protótipo são responsáveis por indicar as principais normas que regem o negócio, evidenciando como ele será gerenciado, eventuais decisões e parâmetros que serão utilizados.

Podemos ver no quadro a seguir, as principais regras de negócio que estão presentes nesse projeto.

Quadro 2 – Regras de negócio.

| Número | Descrição |
|--------|--|
| RN01 | O agricultor poderá indicar vários tamanhos de áreas de plantio. Dentro de uma propriedade rural, as áreas de plantio correspondem ao espaço de terra onde serão realizados as atividades relacionadas ao cultivo de uma ou mais culturas. |
| RN02 | O agricultor poderá selecionar culturas diferentes em uma mesma área de plantio. A cultura denomina a espécie de planta que será cultivada em uma determinada época do ano. |
| RN03 | O agricultor poderá indicar vários períodos de plantio de suas culturas, em suas áreas disponíveis. Como as plantações são sazonais, os períodos de plantio poderão indicados em um mesmo ano, ou em anos diferentes. Por exemplo, na região sul do Brasil, é comum a cultura da soja ser plantada em outubro, sendo que sua colheita acontece apenas em março do ano seguinte. |
| RN04 | O agropecuarista poderá ter vários produtos em seu portfólio, inclusive com tamanhos, preços e marcas diferentes. Essa característica pode trazer um diferencial competitivo, sendo que quanto maior seu portfólio for, mais produtos poderão ser utilizados nas composições, onde em alguns casos, pode baixar o valor dos orçamentos realizados pelo agricultor. |
| RN05 | As composições são a união dos produtos necessários para realizar o cultivo de determinada cultura. São elas criadas pelo agropecuarista, através de seu portfólio de produtos disponível na agropecuária. Através das mesmas, o agricultor poderá ver qual das composições se encaixa melhor em seu cenário, podendo observar o valor e o nível de produtividade de cada uma. |
| RN06 | As áreas de plantio serão mensuradas através da medida “hectare”. Na região sul do Brasil, um hectare diz respeito a dez mil metros quadrados de terra. |
| RN07 | O orçamento será calculado considerando o tamanho da área de plantio informada, os preços dos produtos e a quantidade de produtos necessária em um hectare. Para o cálculo primeiramente é utilizada a regra de três para identificar a quantidade necessária do produto no tamanho da área de plantio informada pelo agricultor, logo após, a quantidade obtida será multiplicada pelo preço dos produtos. Por fim, os preços serão somados para que seja possível ter o valor final do orçamento. |

Fonte: Acervo do autor (2021)

4.4 REQUISITOS

Os requisitos são funcionalidades que podem ou devem estar disponíveis no protótipo para que seja possível cumprir o proposto anteriormente.

O quadro a seguir, representa os requisitos não funcionais do protótipo. Esses requisitos apontam as características de qualidade esperadas mais relevantes para o sistema. Essas características podem envolver segurança, usabilidade, performance, restrições de plataforma, etc.

Quadro 3 - Requisitos não funcionais.

| Número | Descrição |
|--------|---|
| RNF01 | O sistema deve gerar um token que será válido por 1 dia. O token será utilizado em todas as comunicações que o usuário fará com o servidor, servindo para identificar o mesmo e garantir a segurança de seu acesso. |
| RNF02 | A senha dos usuários deve ser criptografada antes de ser cadastrada no banco, dessa forma o usuário terá mais segurança ao utilizar o sistema. |
| RNF03 | O sistema deve possuir técnicas de usabilidade consolidadas para facilitar seu uso. Essas técnicas de usabilidade serão baseadas nas Heurísticas de usabilidade de Nielsen (1994). |
| RNF04 | O sistema poderá utilizar os ícones do <i>Material Design</i> . Esse pacote de ícones é utilizado por aplicações consolidadas, e já são familiares para a maioria dos usuários que navegam na |

| | |
|-------|---|
| | internet. |
| RNF05 | O sistema deverá ser construído com uma paleta de cores de no máximo 8 cores principais, podendo ter até 3 variações de tons mais claros ou escuros das mesmas. Essa limitação de cores evita que haja uma poluição visual seguindo uma das heurísticas de usabilidade. |
| RNF06 | O sistema deverá mostrar uma mensagem amigável para o usuário em casos de problemas de conexão, servidor ou erros internos da aplicação. |
| RNF07 | Os elementos visuais do sistema podem possuir bordas arredondadas, essa característica deixa a aplicação mais “amigável” para a utilização. |

Fonte: Acervo do autor (2021)

No próximo quadro, serão pontuados os requisitos funcionais do protótipo, que estão ligados com as atividades que serão realizadas no sistema. Eles representam as funções propriamente ditas, e que estarão disponíveis ao usuário.

Quadro 4 - Requisitos funcionais.

| Número | Descrição |
|--------|---|
| RF01 | Para identificar o agricultor no sistema, o mesmo deverá fazer seu cadastro no aplicativo móvel, informando seu e-mail, nome e senha de acesso. |
| RF02 | Para identificar o agropecuarista no sistema, o mesmo deverá fazer seu cadastro na página <i>web</i> , informando seu e-mail, nome, senha e endereço da agropecuária. Automaticamente o agropecuarista será cadastrado no sistema como “provedor” de serviços. |
| RF03 | A senha do agricultor e do agropecuarista devem conter pelo menos 6 dígitos, seguindo o padrão dos sistemas já consolidados. |
| RF03 | O agricultor poderá manter o cadastro de períodos de plantio em seu aplicativo. O período de plantio diz respeito ao início e término de cultivo de uma cultura na safra, que ocorre no decorrer de um ano. |
| RF04 | O agricultor poderá manter o cadastro de novas áreas de plantio, onde será possível informar o tamanho da área em “hectares”. Também será possível nomear as áreas para facilitar a identificação futura. As áreas deverão ser cadastradas através do aplicativo móvel selecionando a localização das mesmas no mapa. |
| RF05 | O agricultor poderá manter o cadastro de novos orçamentos em seu aplicativo. Nos orçamentos ele poderá ver qual é a agropecuária fornecedora, qual a marca e preço dos produtos e ter informações detalhadas sobre a quantidade de produto necessária para cada hectare de plantio. |
| RF06 | O sistema contará com uma base de dados de produtos, culturas, marcas e unidades de medida padronizada. Essas informações serão cadastradas a nível gerencial pelo administrador do sistema, diretamente pelo banco de dados, e não poderão ser alteradas por outros usuários. |
| RF07 | O sistema deverá gerar o orçamento de forma automática para os agricultores. Ao selecionar a cultura, informar o tamanho da área de plantio e o nível de produtividade esperado, o sistema trará todas as agropecuárias que possuem composições disponíveis. |
| RF08 | O sistema deverá possuir login, da forma que identifique a categoria de usuários, sendo agricultor ou agropecuarista, isso deve ser feito de forma automática para facilitar o uso e aumentar a segurança da aplicação. |
| RF09 | O agropecuarista será capaz de manter o cadastro de produtos em seu portfólio, tendo como base os produtos que foram cadastrados de forma gerencial pelo administrador. |
| RF10 | O agropecuarista poderá manter o cadastro de composições de produtos em seu estabelecimento. As composições de produtos serão criadas com base nas informações dos produtos cadastrados no portfólio anteriormente. |

Fonte: Acervo do autor (2021)

Com base nos requisitos funcionais apontados no quadro 4, a seguir serão apresentados os diagramas de funcionamento do sistema.

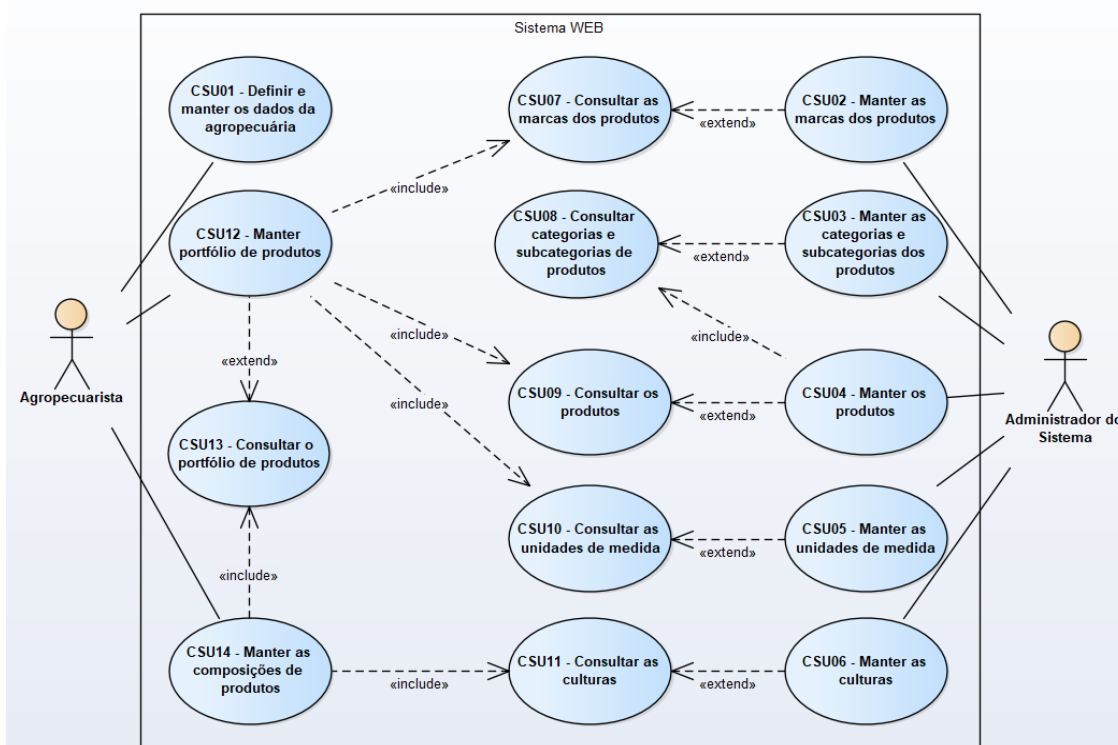
4.5 DIAGRAMAS

Os diagramas a seguir representam as ações que serão realizadas pelos atores envolvidos. Foram criados diagramas de caso de uso e diagramas de atividade, afim de melhorar o entendimento e a visualização dos principais recursos, como navegação entre telas ou cadastros de produtos e orçamentos.

O administrador do sistema é responsável por fazer o cadastro gerencial dos produtos, marcas, unidades de medida, categorias, subcategorias e culturas. Para os usuários, esses dados cadastrados pelo administrador apareceram de forma padrão e não poderão ser alterados. Em sua plataforma *web*, o agropecuarista fará o uso dessas informações cadastradas pelo administrador do sistema para a criação do portfólio de sua agropecuária. Além disso, esse ator também é responsável pela criação de composição de produtos, que será utilizada mais tarde pelos agricultores na criação de seus orçamentos.

Através da figura 6, podemos observar o diagrama de casos de uso envolvendo todas as ações disponíveis para o agropecuarista e administrador do sistema.

Figura 6 – Diagrama de caso de uso do agropecuarista e administrador do sistema.

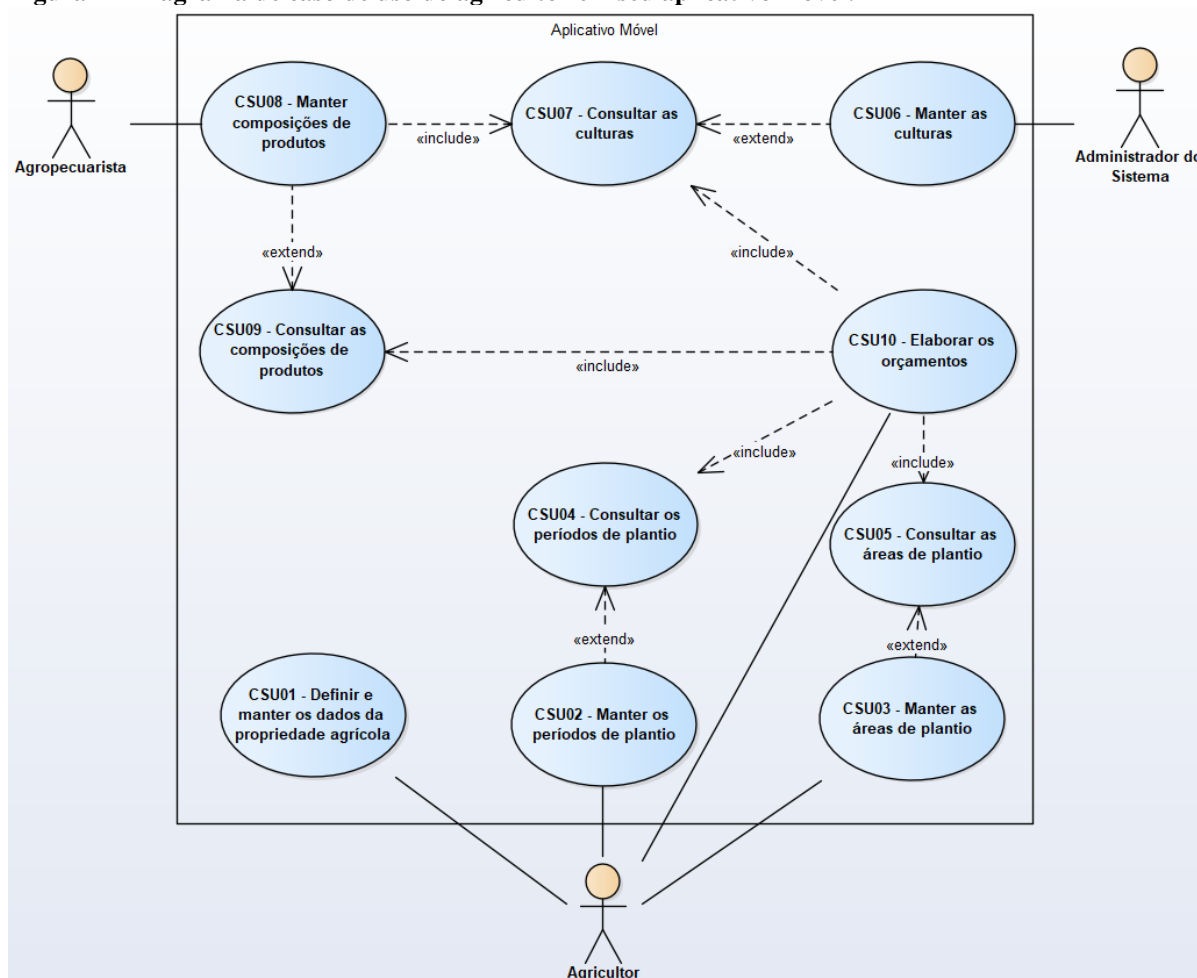


Fonte: Acervo do autor (2021)

Quanto aos agricultores, a tarefa exclusiva desse ator é o cadastro de orçamentos e o gerenciamento dos mesmos. No sistema, assim como os agropecuaristas, os agricultores têm suas atividades bem definidas. Sendo assim, somente o agricultor poderá realizar o cadastro de áreas através do georreferenciamento, e a criação de orçamentos de composições de produtos para as mesmas, podendo identificar qual estabelecimento agropecuário tem os melhores produtos para seu cultivo, observando os preços mais atrativos para a conclusão de seus orçamentos.

Com base na figura 7 a seguir, podemos observar o diagrama de caso de uso do agricultor, onde são evidenciadas todas as atividades que esse ator pode realizar em seu aplicativo móvel.

Figura 7 – Diagrama de caso de uso do agricultor em seu aplicativo móvel.



Fonte: Acervo do autor (2021)

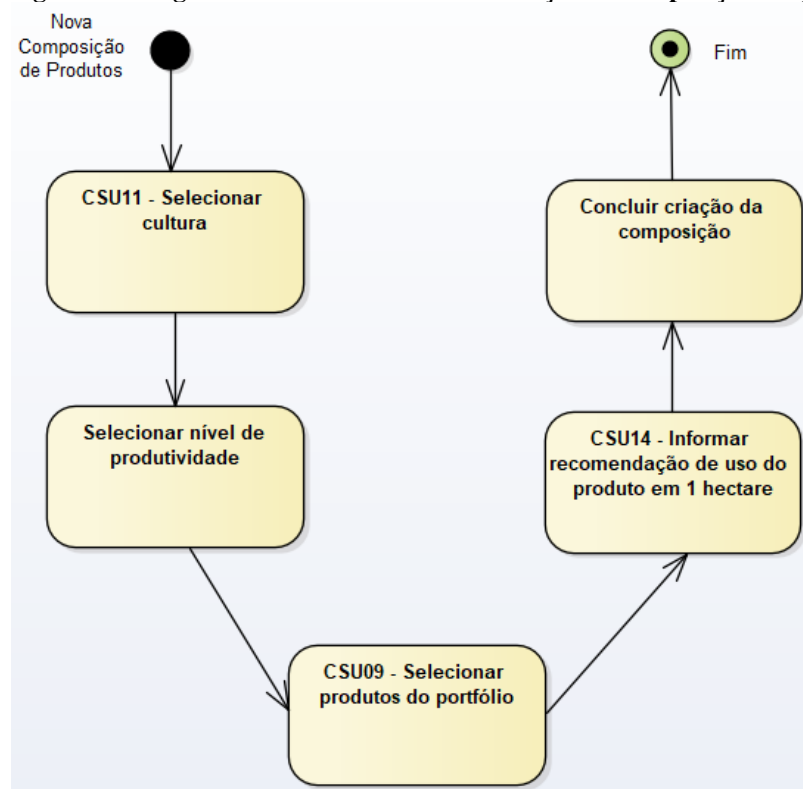
Uma das tarefas exclusivas dos agropecuaristas é o cadastro de produtos no portfólio do estabelecimento. Para facilitar essa tarefa, o sistema trará uma base de dados de produtos padronizada, onde estarão contidas várias informações sobre os produtos. Dessa forma, ao

cadastrar um novo produto em seu portfólio, o responsável pela atividade informará apenas o volume do produto, seu respectivo valor e unidade de medida.

Outra tarefa exclusiva desse ator é o cadastro de composições de produtos, essa atividade é necessária para que o agricultor não perca tempo escolhendo os produtos corretos para o cultivo de suas lavouras. Nessa etapa, o agropecuarista deve informar ao sistema os produtos indicados para determinada cultura, bem como a quantidade recomendada para o correto manejo no campo.

Com a figura 8, podemos observar o diagrama envolvendo as atividades do agropecuarista ao criar uma nova composição no sistema. Para facilitar o entendimento, nas atividades são referenciados os cenários de uso relacionados.

Figura 8 – Diagrama de atividades sobre a criação de composições de produtos.



Fonte: Acervo do autor (2021)

Os orçamentos poderão ser realizados pelo agricultor através da aplicação móvel, onde será possível selecionar no mapa a localização da área de plantio, informando seu respectivo nome e tamanho. É importante lembrar que as áreas sempre serão mensuradas através da unidade de medida hectare.

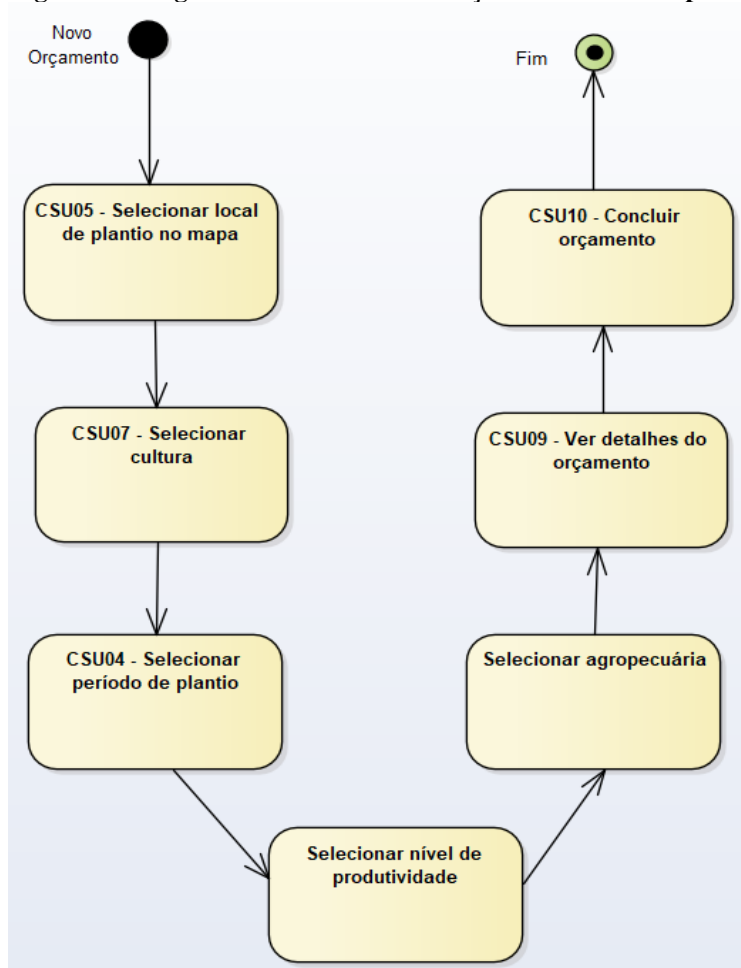
O próximo passo é fazer a seleção das culturas, essas culturas por sua vez já estarão cadastradas no sistema de forma gerencial, dessa forma o agricultor apenas selecionará a

cultura desejada. Após essa etapa, o usuário informará o período de plantio, indicando o início e o término da safra. Dessa forma é possível criar históricos de safras permitindo que agricultor faça análises rápidas sobre seus gastos e investimentos com o passar do tempo. O penúltimo passo é escolher o nível de produtividade que o agricultor espera para seu cultivo, normalmente o nível de produtividade de uma lavoura está diretamente relacionado a qualidade dos produtos nela utilizados, quanto maior a qualidade dos produtos, maior será seus respectivos valores, onde os mesmos, podem trazer uma maior produtividade no campo.

Por fim, o orçamento será gerado em todas as agropecuárias que possuem produtos disponíveis para a cultura e produtividade selecionada, isso ocorrerá de forma automática. Após o orçamento ser gerado, o agricultor poderá ver os detalhes de cada orçamento selecionando a agropecuária desejada.

A seguir na figura 9, podemos ver o diagrama de atividades do orçamento do agricultor, mostrando o passo a passo que será realizado por este ator em seu aplicativo móvel.

Figura 9 – Diagrama de atividades do orçamento realizado pelo agricultor.



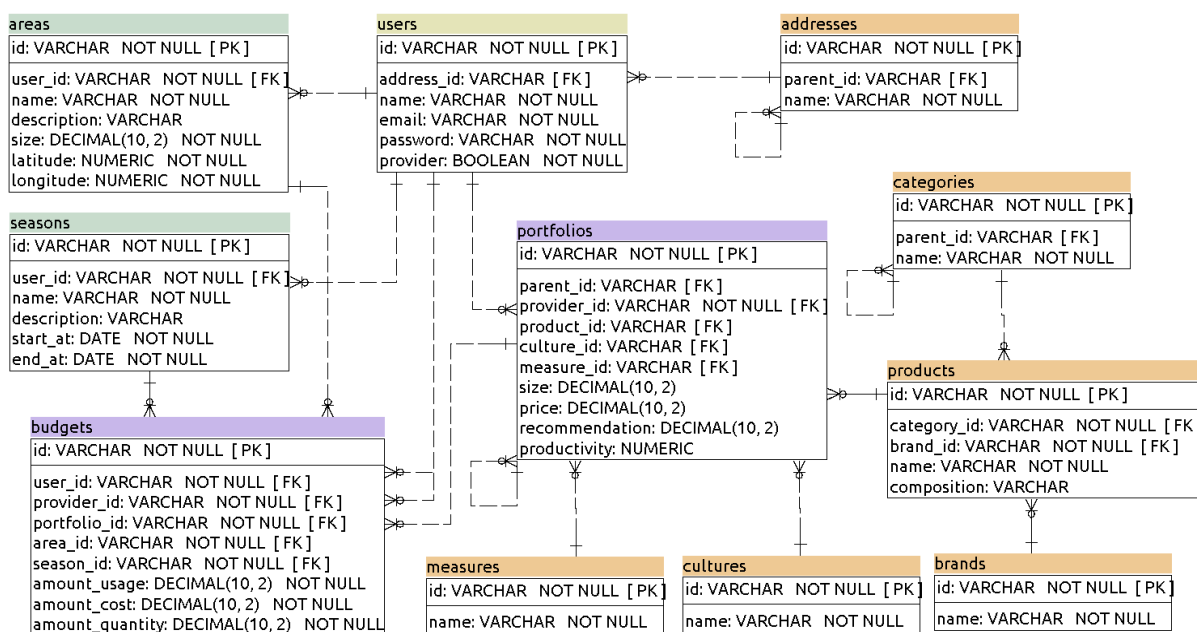
Fonte: Acervo do autor (2021)

Após a criação de um orçamento na aplicação, o agricultor poderá fazer a visualização do mesmo em seu aplicativo.

Para possibilitar uma melhor visualização da estrutura dos dados no banco de dados, foi elaborado o diagrama entidade e relacionamento, também conhecido pela sigla DER. Este diagrama visa evidenciar as ligações entre as entidades, bem como, as propriedades de cada uma delas, seguindo a nível de *software*, as regras de negócio e requisitos levantados anteriormente.

Na figura 10 a seguir, podemos ver o diagrama de entidade e relacionamento fazendo a representação das tabelas no banco de dados.

Figura 10 – Diagrama de entidade e relacionamento DER.



Fonte: Acervo do autor (2021)

Na construção das estruturas de dados, fora optado pelas nomenclaturas tanto de tabelas como de campos em inglês.

4.6 AVALIAÇÃO HEURÍSTICA DE USABILIDADE

Avaliação heurística é uma técnica de inspeção de usabilidade desenvolvida por Nielsen em 1994, onde os princípios de usabilidade denominados heurísticas avaliam se os elementos da interface do usuário estão de acordo com certos padrões. O objetivo da avaliação é identificar problemas no design da interface possibilitando a correção.

Nielsen determina como heurísticas básicas as dez regras apresentadas a seguir. Para cada característica é descrito como o esperasse que ela seja aplicada no protótipo.

4.6.1 Dialogo simples e natural

As interfaces devem ser simples, uma vez que a quantidade de recursos adicionais sobrecarrega a página, podendo não serem necessárias naquele momento, deve corresponder a tarefa do usuário de forma mais natural possível, para facilitar a relação entre conceitos do usuário e do computador, onde o ideal é mostrar a informação que o usuário precisa.

No protótipo utilizou-se o conceito de etapas de navegação, onde só é possível encontrar informações necessárias para a etapa em questão, evitando confundir o usuário com a poluição visual ou com informações de outras etapas que não serão úteis no momento.

Na figura 11, é apresentada a etapa de listagem de composições no sistema do agropecuarista, onde não há sobrecarga de informações adicionais, sendo possível apenas excluir ou ver detalhes de uma composição.

Figura 11 – Dialogo simples e natural.

| | Cultura | Nível de Produtividade |
|-------------------------------------|---------|------------------------|
| <input type="checkbox"/> | Soja | Produtividade Baixa |
| <input type="checkbox"/> | Arroz | Produtividade Baixa |
| <input checked="" type="checkbox"/> | Soja | Produtividade Média |

Excluir Ver Detalhes

Fonte: Acervo do autor (2021)

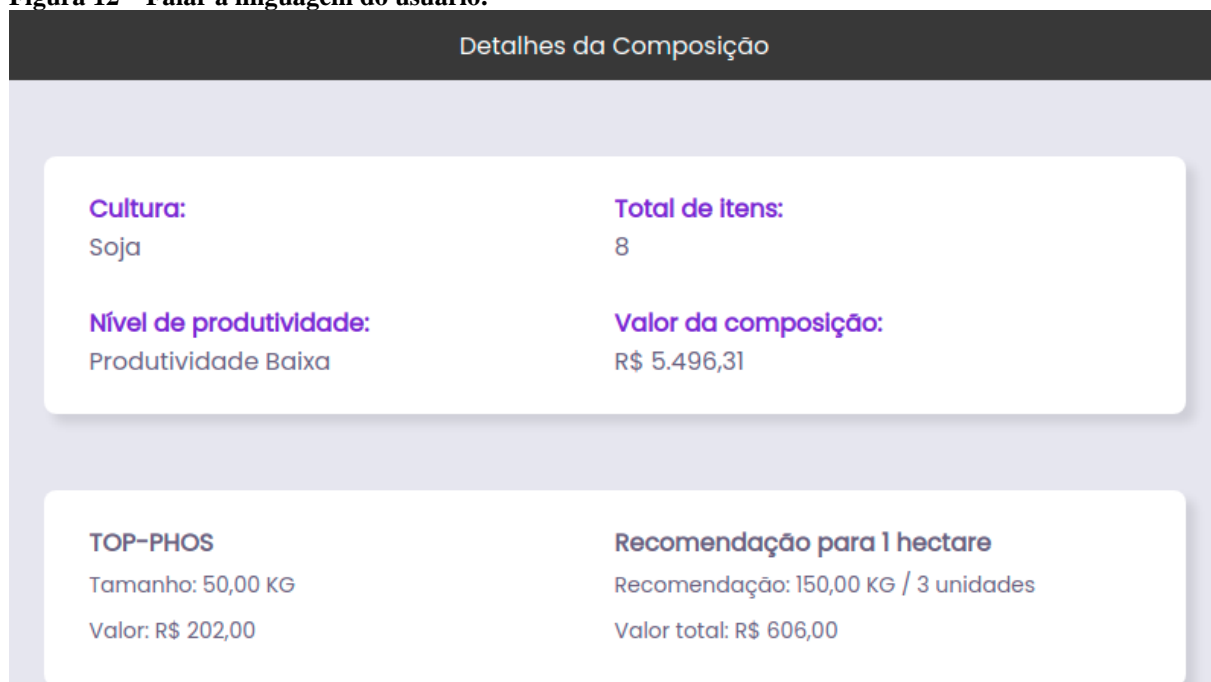
4.6.2 Falar a linguagem do usuário

A terminologia da interface do usuário deve ser voltada para o usuário, e não para o sistema, portanto a linguagem utilizada deve ser o mais natural possível, evitando códigos do sistema. É necessário um cuidado na utilização de elementos não verbais, como ícones.

Sendo assim, a linguagem utilizada foi baseada nos termos que os usuários estão acostumados a utilizar no dia a dia, sem empregar termos técnicos avançados, ajudando a identificar as ações de forma rápida.

Como é apresentado na figura 12 a seguir, a linguagem usada no sistema é muito simples e objetiva.

Figura 12 – Falar a linguagem do usuário.



Fonte: Acervo do autor (2021)

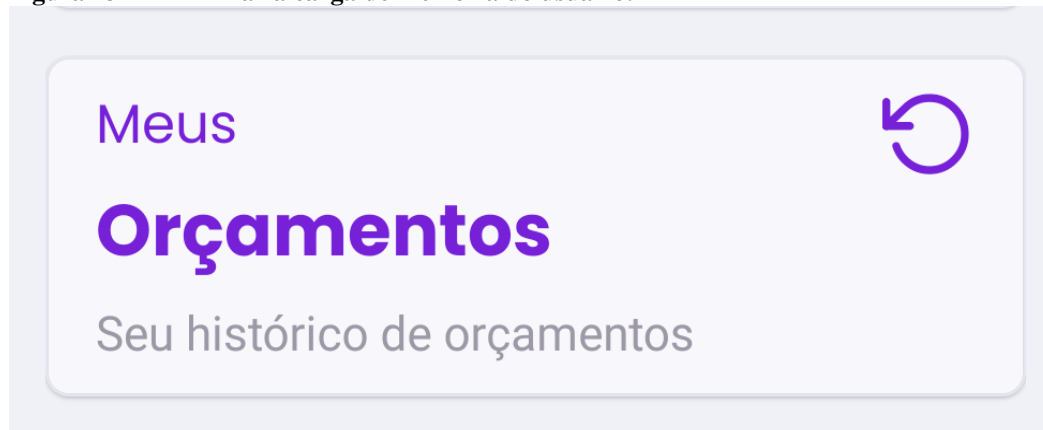
4.6.3 Minimizar a carga de memória do usuário

Para diminuir a necessidade de memorização do usuário, o sistema deve exibir elementos de diálogo, disponíveis para seleção, assim como elementos de interface que possibilitem escolher ações.

Pensando nisso, o sistema foi desenvolvido dando foco nas atividades realizadas pelos usuários. Por exemplo, ao selecionar uma determinada cultura, os orçamentos e outras informações serão gerados somente para a cultura em questão, sem haver informações concorrentes. Dessa forma, os usuários não precisam ficar pensando em qual informação será a opção correta para a cultura desejada. Outra característica que diminui a carga de memória do usuário é a presença da funcionalidade de históricos de orçamento, onde através da mesma, é possível conferir informações de orçamentos gerados anteriormente.

Com a figura 13, podemos visualizar a ação de históricos de orçamentos presente na tela inicial do aplicativo móvel do agricultor.

Figura 13 – Minimizar a carga de memória do usuário.



Fonte: Acervo do autor (2021)

4.6.4 Consistência

O sistema deve garantir integridade de suas funções, que deverão ter o mesmo efeito quando executadas, além de apresentar informações de acordo com um padrão estabelecido, facilitando assim o reconhecimento.

Diante do proposto, o sistema possui o mínimo possível de ações, sendo que quando executadas geram o mesmo resultado em diferentes partes do sistema. Mensagens de erros e avisos são padronizadas facilitando o reconhecimento dos usuários.

Podemos conferir através da figura 14, a mínima presença de ações disponíveis na tela de cadastro de produtos no portfólio, mantendo o usuário focado em suas atividades.

Figura 14 – Consistência.

| Q Pesquisar... | | Página 1 de 1 << < > >> | | | |
|-------------------------------------|------------------|-------------------------|---------------|--------------|------------|
| <input type="checkbox"/> | Produto | Marca | Categoria | Subcategoria | Composição |
| <input type="checkbox"/> | Sulfamo | Timac | Fertilizantes | Químico | AIN18 |
| <input type="checkbox"/> | Ureia Peletizada | Yara | Fertilizantes | Químico | N12K8E2 |
| <input type="checkbox"/> | Ureia | Yara | Fertilizantes | Químico | N12 |
| <input checked="" type="checkbox"/> | Round Up Ultra | Bayer | Agrotóxicos | Herbicida | Não contém |
| Cadastrar | | | | | |

Fonte: Acervo do autor (2021)

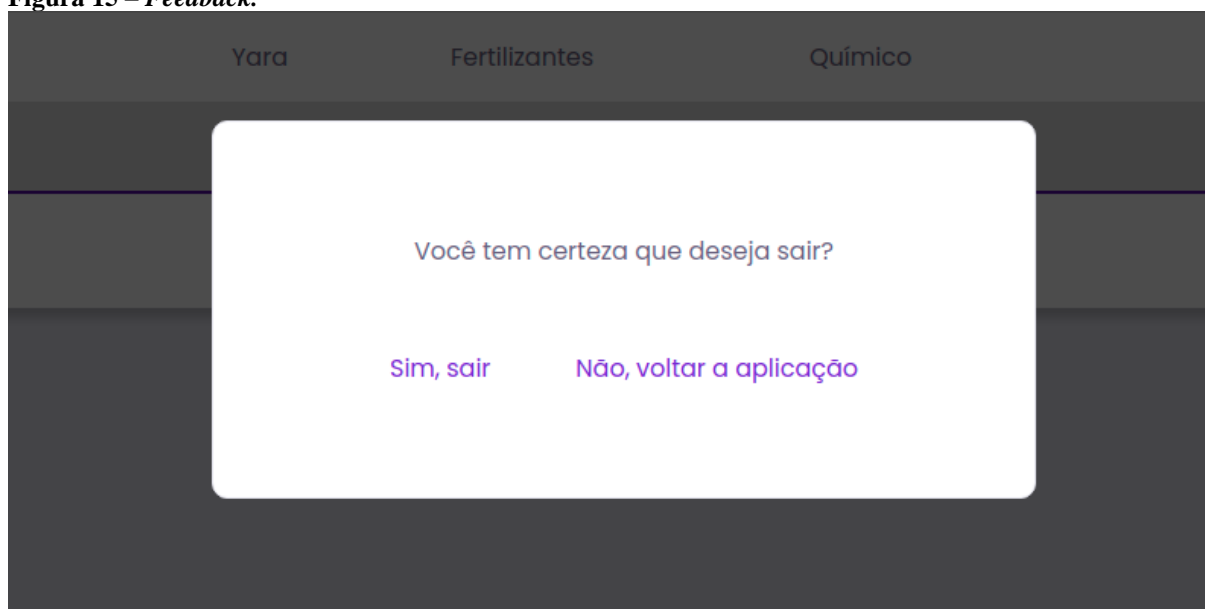
4.6.5 Feedback

O sistema deve manter um diálogo contínuo com o usuário, sobre o que o mesmo está fazendo, principalmente antes de ações definitivas como subscrição de arquivos ou exclusão de dados.

Dessa forma, o sistema mostra caixa de diálogos pedindo a confirmação dos usuários antes de uma ação definitiva. Também são mostradas caixas de diálogo indicando que ao finalizar uma operação obteve-se sucesso ou não.

Na figura 15, podemos ver a caixa de diálogo exibindo o questionamento para a ação definitiva “sair do sistema”.

Figura 15 – Feedback.



Fonte: Acervo do autor (2021)

4.6.6 Saídas evidentes

O sistema deve sempre disponibilizar uma forma de saída de uma determinada ação para o usuário, com isso a segurança do usuário em explorar o sistema aumentará, facilitando o aprendizado de forma exploratória.

Pensando nisso, o sistema *web* foi construindo usando o conceito de *single page application* (em português, aplicação de página única). Isso facilita a utilização do usuário pois não é necessário fazer o recarregamento das páginas no navegador, evitando a espera. Além disso, foi disponibilizado um recurso para voltar um passo na navegação da tela.

Com figura 16 a seguir, é possível observar a opção “voltar” muito evidente no cabeçalho do sistema, facilitando a saída do usuário quando o mesmo entrar em uma tela indesejada.

Figura 16 – Saídas evidentes.



Fonte: Acervo do autor (2021)

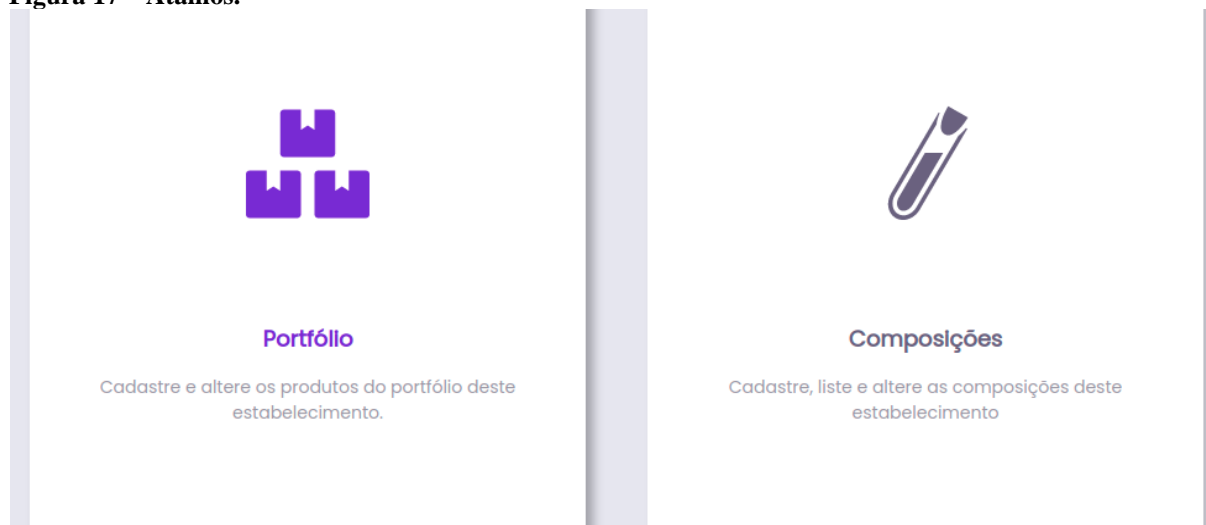
4.6.7 Atalhos

Atalhos devem ser fornecidos para as ações frequentemente usadas, como abreviações, teclas de função ou comandos chaves e caixas de diálogos com teclas para acessar funções importantes.

Diante do proposto, o desenvolvimento da interface foi pensada considerando as ações mais acessadas pelos usuários, deixando as mesmas mais atraentes e com disposição em destaque nas telas.

Através da figura 17, percebe-se o enfoque no ícone e título quando o mouse passa sobre o mesmo, atraindo e informando ao usuário que ele está prestes a executar uma ação.

Figura 17 – Atalhos.



Fonte: Acervo do autor (2021)

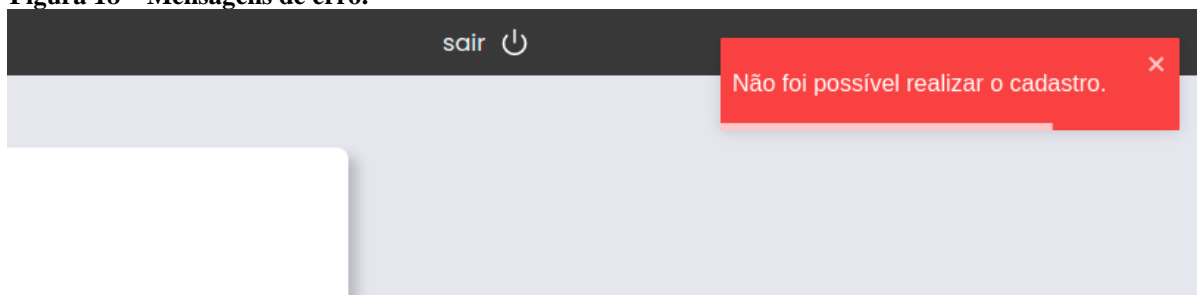
4.6.8 Mensagens de erro

Mensagens de erro devem ser claras e precisas, de forma que o usuário entenda facilmente o ocorrido, auxiliando-o a resolver o problema se possível.

Dessa forma, as mensagens de erro possuem coloração única, apresentando de forma clara e objetiva o que ocorreu na aplicação, sem o uso de termos técnicos avançados e com textos padronizados. Lembrando que mesmas, desaparecerão de forma automática após 3 segundos.

A seguir na figura 18, é possível identificar a mensagem de erro do sistema ao tentar realizar um cadastro.

Figura 18 – Mensagens de erro.



Fonte: Acervo do autor (2021)

4.6.9 Prevenção de erros

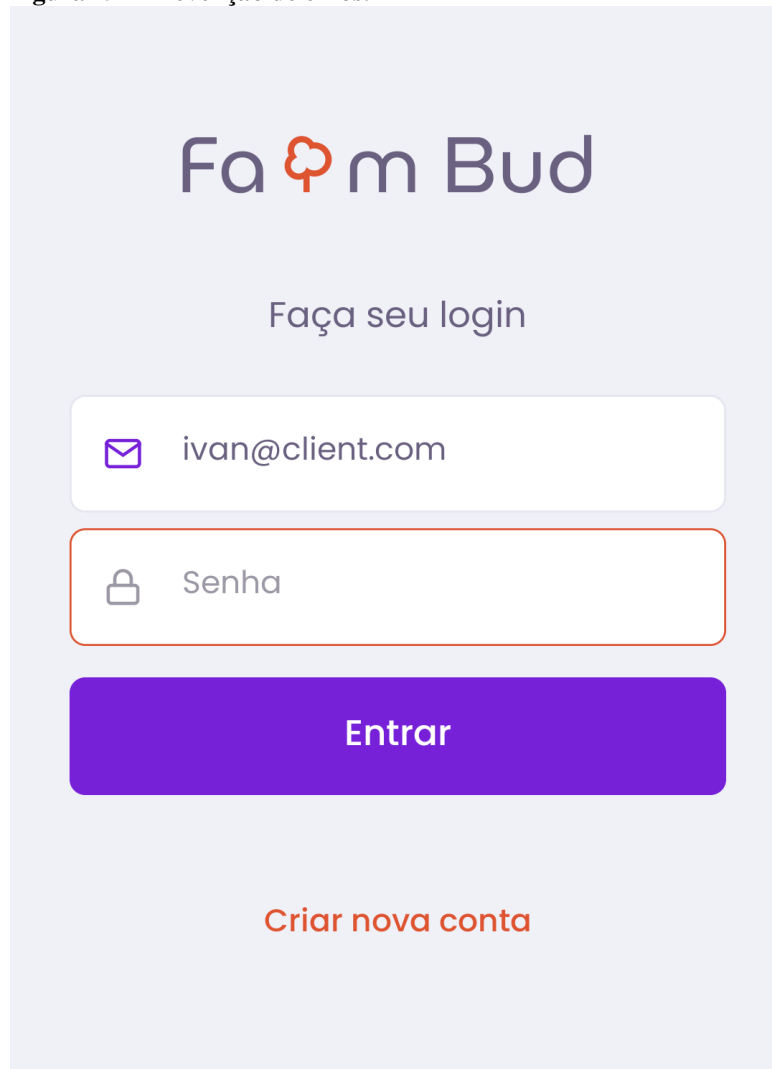
Situações propícias a erros devem ser previstas, como solicitações de dados que não poderiam ser selecionados pelo usuário, ou até mesmo, a falta de uma informação obrigatória para a execução de alguma tarefa seguinte. É importante informar o usuário de forma harmônica, oferecendo sugestões visuais adequadas para que mesmo possa fazer o uso correto da interface.

Tendo isso em mente, o sistema foi estruturado para mostrar aos usuários apenas informações que estão disponíveis no momento em questão, dificultando a execução de erros.

Na figura 19, ao tentar acessar o aplicativo sem realizar o preenchimento do campo “senha”, o mesmo retorna um feedback visual ao usuário, ganhando borda de cor laranjada, evidenciando que há informações faltantes para execução da operação de *login*. Se observarmos o ícone presente dentro do campo, também podemos ver que o mesmo está com coloração cinza, ou seja, nenhuma informação foi adicionada ao mesmo. Já no campo acima,

de e-mail do usuário, o ícone do campo aparece em cor roxa, indicando que informações foram adicionadas ao campo.

Figura 19 – Prevenção de erros.



The image shows a login interface for a service named 'Fa m Bud'. At the top, the logo 'Fa m Bud' is displayed, with a stylized orange flower icon between 'Fa' and 'm'. Below the logo, the text 'Faça seu login' is centered. There are two input fields: the first is for an email address, containing 'ivan@client.com', with a purple envelope icon on the left; the second is for a password, containing the placeholder 'Senha', with a red outline and a lock icon on the left. Below these fields is a large purple button labeled 'Entrar'. At the bottom, there is a link labeled 'Criar nova conta' in orange text.

Fonte: Acervo do autor (2021)

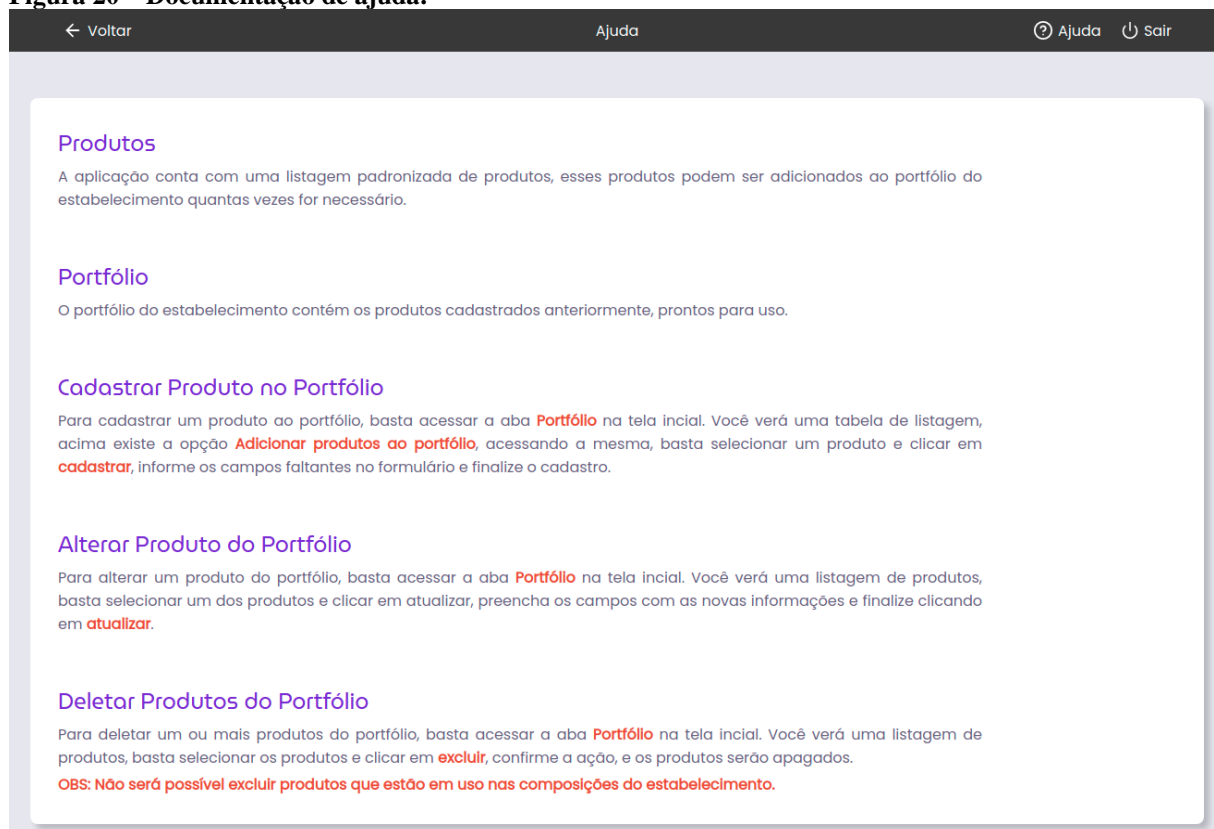
4.6.10 Documentação e ajuda

É preferível que um sistema seja simples o suficiente para possibilitar sua utilização sem a necessidade de algum tipo de ajuda ou documentação, mas este objetivo normalmente não pode ser cumprido, portanto deve-se ter um texto de qualidade e informações bem estruturadas.

Pensando nisso, o sistema contará com uma aba de ajuda contendo as principais dúvidas. As ajudas serão descritas de forma extremamente clara e objetiva, facilitando o entendimento e reduzindo o tempo de leitura dos usuários.

Através da figura 20, podemos visualizar a aba de ajuda na aplicação *web*, contendo uma listagem com todos os recursos da aplicação, explicando cada um deles de forma detalhada e objetiva.

Figura 20 – Documentação de ajuda.



Fonte: Acervo do autor (2021)

4.7 IMPLEMENTAÇÃO

Como anteriormente relatado, para o desenvolvimento do sistema, foram selecionadas tecnologias e ferramentas voltadas para *web*. O Javascript foi escolhido como linguagem de programação, foi usado na criação das telas do aplicativo móvel e nas páginas *web*, atuando também no controle de dados do lado do servidor, executando a *API*.

No desenvolvimento da *API* utilizou-se Node, um *framework* Javascript que faz o tratamento das informações no *back-end*. Para aumentar a produtividade no desenvolvimento fez-se o uso do Express, um *framework* com estrutura flexível e minimalista, que ajuda no tratamento das requisições HTTP enviadas do cliente.

Sobre o Javascript foi usado um *superset* de tipagens conhecido com Typescript. Essa ferramenta foi desenvolvida pela Microsoft visando melhorar a produtividade durante o desenvolvimento de aplicações com a linguagem Javascript. O Typescript permite não só o

uso de tipos sob as variáveis do código fonte, mas também, o uso de *decorators*. Os *decorators* são muito utilizados na linguagem Java e fornecem uma maneira de adicionar uma sintaxe de metaprogramação para declaração de classes, atributos e métodos.

Na figura 21 a seguir, podemos ver o uso dos *decorators* do Typeorm na classe de usuário da aplicação.

Figura 21 – Uso de *decorators* na classe de usuário.

```
@Entity('users')
export default class User {
  @PrimaryGeneratedColumn('uuid')
  id: string;

  @Column()
  name: string;

  @Column()
  email: string;

  @Exclude()
  @Column()
  password: string;
```

Fonte: Acervo do autor (2021)

Ao manipular dados no banco, também foram utilizadas as ferramentas do Typeorm, um ORM (*Object relational mapping*, em português, mapeamento objeto relacional), muito poderoso e flexível, responsável por reduzir a impedância da programação orientada à objetos. Com ele é possível gerenciar as entidades da aplicação, criar repositórios customizados e fazer o uso de *migrations* entre outras funcionalidades, tudo isso seguindo padrões de projeto como DataMapper ou ActiveRecord.

Na figura 22, podemos ver a utilização de um repositório customizado do Typeorm também é possível observar como são criados os comandos manipulação de dados no banco. Se observarmos, o retorno do método “findByEmail” (encontrar através do e-mail, em

português), o mesmo está gerando um comando de seleção de dados no banco, sem a necessidade da escrita de código SQL, sendo criado apenas através de objetos do Javascript.

Figura 22 – Repositório customizado de usuário.

```
import { getRepository, Repository } from 'typeorm';

export default class UsersRepository implements IUsersRepository {
  private ormRepository: Repository<User>;

  constructor() {
    this.ormRepository = getRepository(User);
  }

  public async findByEmail(email: string): Promise<User | undefined> {
    return this.ormRepository.findOne({ where: { email } });
  }
}
```

Fonte: Acervo do autor (2021)

Quanto as páginas *web* da aplicação, todas foram criadas utilizando a abordagem de componentização presente no React. Os componentes podem ser entendidos como a representação de funcionalidades isoladas dentro de uma aplicação. Esses componentes envolvem lógica através do Javascript, estrutura através do HTML e estilização através do CSS. A junção dessas três tecnologias formam a nomenclatura conhecida como JSX.

Na figura 23, podemos ver um breve exemplo de como o código é estruturado seguindo o que foi relatado anteriormente.

Figura 23 – JSX no componente React.

```
export default function ComponentExample() {
  const handleClick = useCallback(() => {
    alert('Você clicou no botão');
  }, [])

  return (
    <div>
      <button type="button" onClick={handleClick}>Botão</button>
    </div>
  );
}
```

Fonte: Acervo do autor (2021)

Uma característica presente em todos os projetos React, é o uso de bibliotecas de terceiros. Essas bibliotecas facilitam o desenvolvimento, trazendo muitas funcionalidades prontas para a utilização. Nesse projeto fez-se o uso de uma biblioteca chamada Styled Components (componentes estilizados em português). O Styled Components é responsável por separar a estilização da página do HTML, criando componentes estilizáveis evitando o uso do conhecido CSS *inline*. Além disso, o Styled Components também conta com as funcionalidades mais recentes do SCSS, são elas: Pré processamento, declaração de variáveis dentro do CSS, herança de estilos por hierarquia e operadores.

A seguir na figura 24, podemos visualizar a criação de um componente de botão através das declarações do Styled Components.

Figura 24 – Componente de *button* com Styled Components.

```
import styled from 'styled-components';

export const Container = styled.button`
  width: 100%;
  height: 56px;
  margin-top: 32px;
`;
```

Fonte: Acervo do autor (2021)

Além do Styled Components, uma biblioteca chamada Axios foi utilizada para fazer as requisições ao *back-end*. O Axios é um cliente HTTP baseado em *promises*. Por baixo dos panos, a ferramenta faz requisições XMLHttpRequests, comumente conhecidas como requisições Ajax. Com ele também é possível declarar uma URL base para as chamadas ao servidor, bem como salvar *tokens* de autenticação nos *headers* da requisição HTTP, mantendo a comunicação segura entre o *front-end* e *back-end*.

Ao lidar com formulários, fez-se o uso de uma biblioteca chamada Uniform. Ela é responsável por garantir a performance e a escalabilidade, tratando os campos do formulário como componentes não controlados. Mas afinal, o que são componentes não controlados? No React por padrão, toda vez que um componente sofre alteração, o mesmo sofre renderização para mostrar em tela a alteração realizada. Dessa forma, sempre que digitamos uma letra em um campo do formulário é feito um recarregamento do campo, consumindo memória sem necessidade e causando lentidão em alguns cenários. O Uniform consegue contornar essa situação fazendo o uso dos *hooks* disponíveis nas versões mais recentes do React. Os *hooks*

são um conjunto de funcionalidades adicionadas ao *framework* com o intuito de facilitar o gerenciamento do ciclo de vida da aplicação.

4.8 UTILIZAÇÃO E FUNCIONAMENTO

Neste capítulo será apresentada a estrutura de distribuição da aplicação em forma de diagrama, exemplificando de forma visual quais foram as tecnologias usadas, e como as mesmas estão dispostas. Também são abordados detalhes quanto ao funcionamento e uso das rotinas do protótipo.

4.8.1 Estrutura de distribuição da aplicação

A construção do protótipo de aplicação envolveu várias tecnologias modernas, entre elas, o React Native 0.63. Utilizado para a construção de aplicativos móveis, a ferramenta consegue criar de forma nativa aplicações multiplataforma, fazendo uso apenas código Javascript. Tecnologias consolidadas também dispuseram seu espaço no cenário, a utilização do PostgreSQL foi indispensável ao salvar os dados de forma concisa no servidor de banco de dados. Quanto a manutenção do banco de dados, o PgAdmin 3 foi instalado, possibilitando que as configurações sejam feitas de forma mais visual, facilitando o trabalho do desenvolvedor.

Em relação ao servidor de aplicação, a tecnologia que possui maior enfoque é o Node, um *framework* com um conjunto de mecanismos que possibilita a utilização do Javascript no *back-end*. Neste cenário, o Express atuou fazendo o tratamento das requisições HTTP feitas a *API*, passando os dados tratados ao Typeorm, que por sua vez é responsável por fazer a manipulação de dados no servidor de banco de dados. Essa manipulação acontece de forma mais abstrata para o desenvolvedor, sendo um ORM, o Typeorm permite ao desenvolvedor que a manipulação de dados aconteça através de objetos e métodos presentes no Javascript, onde a criação das *queries* SQL acontecem por baixo dos panos.

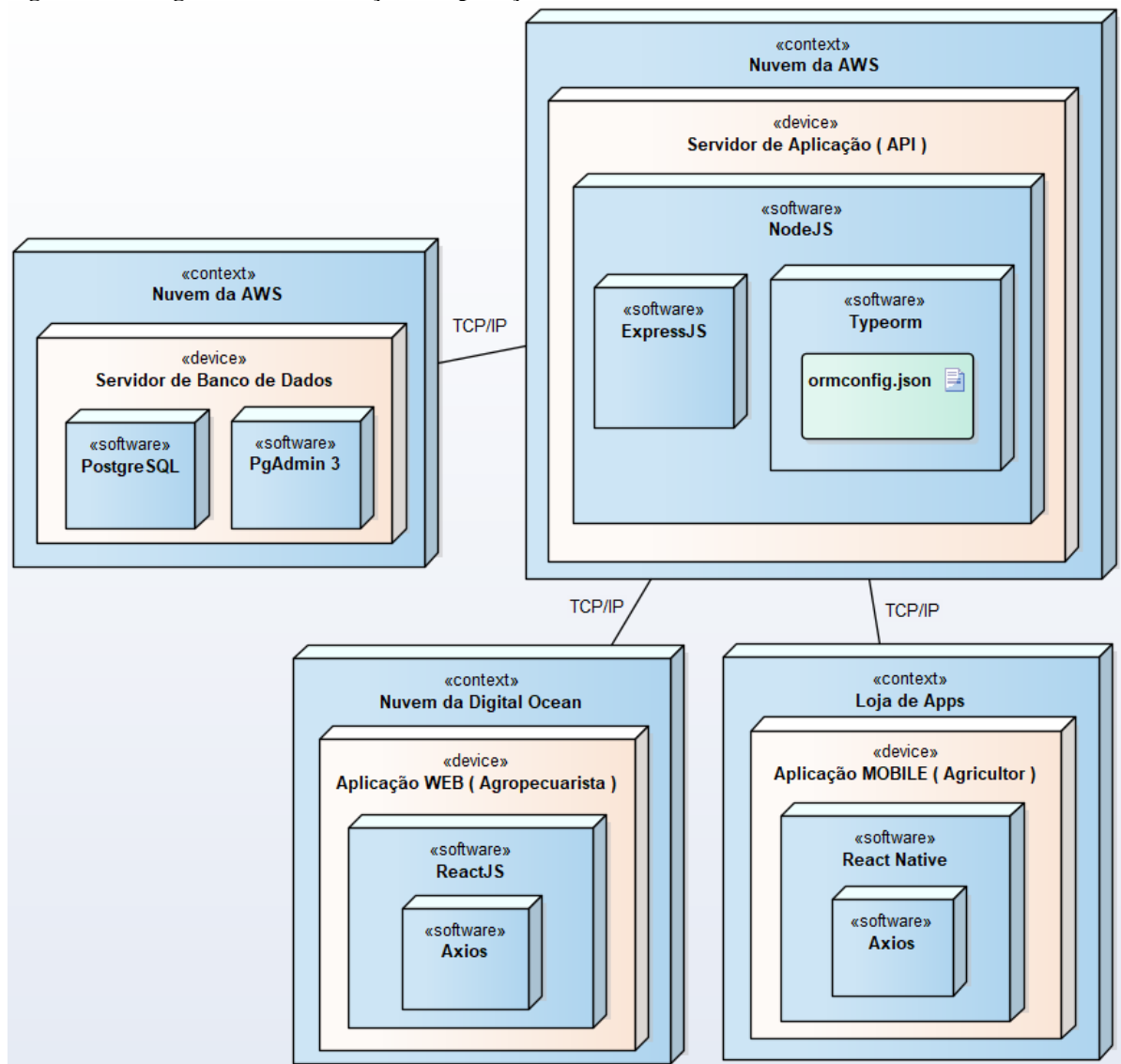
Quanto a página *web* do protótipo, fez-se uso do *framework* React. Escrito em Javascript e voltado para a construção de *interfaces*, a tecnologia usa o paradigma de componentização de elementos. Esse paradigma pode ser entendido como a quebra ou separação dos elementos em tela, isolando suas responsabilidades, trazendo mais desempenho e organização ao projeto.

Para que a comunicação da *API* com as páginas *web* e com o aplicativo móvel, o Axios atuou, fazendo o envio e o recebimento de requisições HTTP padronizadas.

Pensando na disponibilidade do protótipo nas camadas *back-end* e *front-end web*, foram feitas integrações com as nuvens da Amazon e Digital Ocean. Para o aplicativo móvel não existe a possibilidade de entrega através das ferramentas de nuvem anteriormente citadas, dessa forma, a aplicação móvel foi posta nas lojas de aplicativos já consolidadas como a AppStore da Apple e PlayStore do Google.

Na figura 25, podemos identificar através do diagrama de distribuição, as informações citadas anteriormente na sessão.

Figura 25 – Diagrama de distribuição da aplicação.



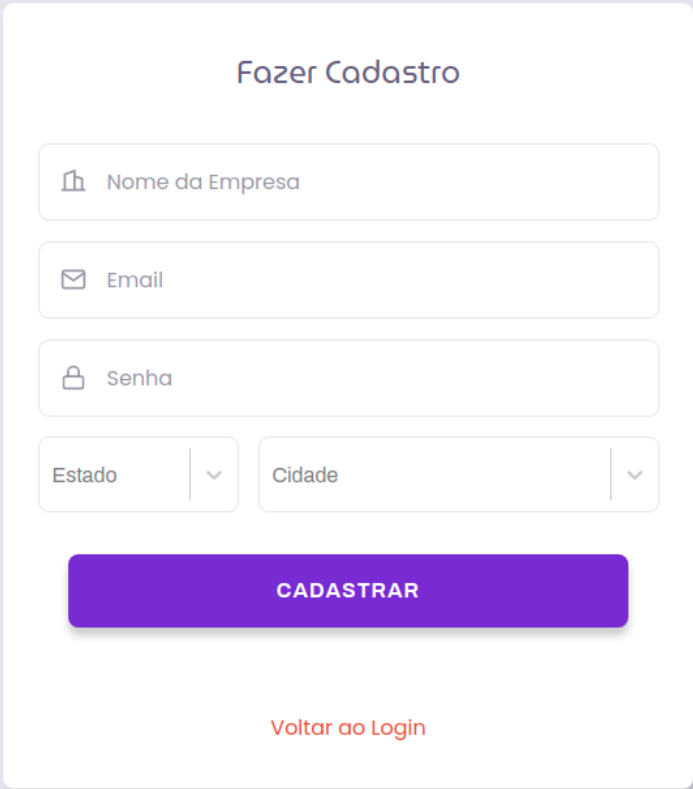
Fonte: Acervo do autor (2021)

4.8.2 Funcionamento

Como descrito anteriormente no quadro de requisitos funcionais do protótipo de aplicação, o agropecuarista deve fazer seu cadastro através da *web*, dessa forma o mesmo será identificado como provedor de serviços e produtos na aplicação. Para fazer o cadastrado da agropecuária no sistema, o usuário deve informar o respectivo nome, endereço de e-mail e senha. Também é necessário informar o estado e cidade onde o estabelecimento se encontra, como uma agropecuária pode fazer parte de uma rede de estabelecimentos, podem existir mais unidades de uma mesma agropecuária em cidades distintas.

Através da figura 26 a seguir, podemos visualizar o formulário de cadastro de agropecuária, contento os campos onde serão preenchidas as informações anteriormente relatadas.

Figura 26 – Formulário de cadastro da agropecuária no sistema *web*.



O formulário de cadastro, intitulado "Fazer Cadastro", apresenta os seguintes elementos:

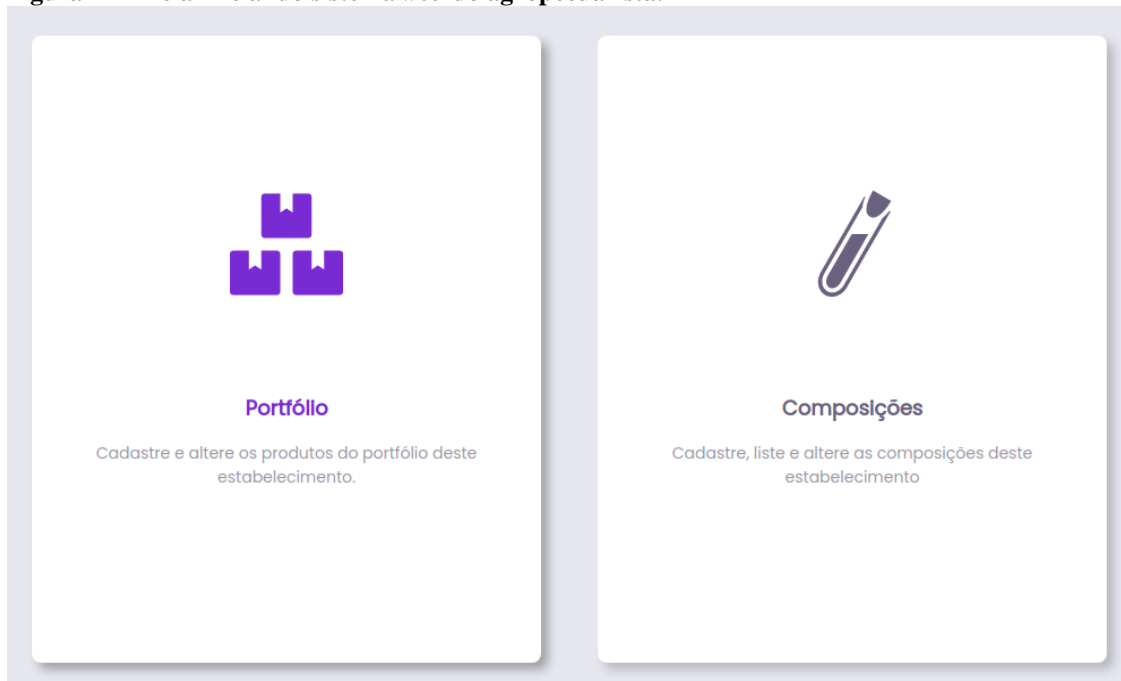
- Um campo de texto para "Nome da Empresa" com um ícone de prédio.
- Um campo de texto para "Email" com um ícone de envelope.
- Um campo de texto para "Senha" com um ícone de cadeado.
- Dois campos de seleção para "Estado" e "Cidade", cada um com uma seta para baixo.
- Um botão de ação "CADASTRAR" em cor verde.
- Um link "Voltar ao Login" em cor laranja.

Fonte: Acervo do autor (2021)

Após fazer o cadastro e realizar o *login* no sistema, o agropecuarista será redirecionado a tela inicial. Nessa etapa o mesmo terá acesso ao cadastramento de produtos em seu portfólio, e a criação das composições de produtos.

Com a figura 27, podemos visualizar a tela inicial do sistema *web*, evidenciando as duas principais ações que o agropecuarista poderá realizar.

Figura 27 – Tela inicial do sistema *web* do agropecuarista.



Fonte: Acervo do autor (2021)

Acessando a ação de “portfólio” na tela inicial, o agropecuarista será redirecionado para a listagem de produtos do estabelecimento. Para realizar o cadastro de um produto no portfólio do estabelecimento, o usuário deve acessar a funcionalidade “adicionar” acima da listagem, ao realizar essa ação, o mesmo será direcionado para outra listagem de produtos essa por sua vez, é uma listagem padronizada e é cadastrada de forma gerencial pelo administrador do sistema através do banco de dados, pois nesta versão do projeto, o mesmo não contará com nenhum tipo de interface visual que lhe auxilie no cadastramento de novas informações na aplicação. Selecionando um produto na listagem, o agropecuarista será direcionado para um formulário contendo informações detalhadas do mesmo, como nome, categoria e composição química, para os produtos que à possuírem, dessa forma o usuário ficará responsável apenas por informar a o tamanho do produto, com sua respectiva unidade de medida que pode ser mensurada através de quilos ou litros e o valor do mesmo.

Na figura 28, é possível fazer a visualização do formulário de cadastramento de produtos no portfólio do estabelecimento, os campos que aparecem com informações na cor marrom, estão bloqueados e não podem ser alterados pelo usuário, estes são os campos que

contém informações padronizadas sobre o produto. Quanto aos campos que apresentam valores na cor cinza, podem ter seus valores informados pelo agropecuarista.

Figura 28 – Formulário de cadastro de produto no sistema web.

Nome: Round Up Ultra

Marca: Bayer

Categoria: Agrotóxicos

Subcategoria: Herbicida

Composição: Não contém

Volume: 5

Unidade de medida: L

Valor: R\$ 189,21

CADASTRAR

Fonte: Acervo do autor (2021)

Ao realizar o cadastro do produto, o agropecuarista poderá visualizar a listagem dos produtos cadastrados em seu estabelecimento, acessando a ação “portfólio”. Nessa listagem ele poderá atualizar e excluir as informações.

Com base na figura 29, visualizamos a listagem de produtos e suas ações disponíveis.

Figura 29 – Listagem de produtos do portfólio do estabelecimento no sistema web.

Adicionar produto ao portfólio **Adicionar**

Q Pesquisar... Página 1 de 1 << < > >>

| | Marca | Produto | Tamanho | Valor | Categoria | Composição |
|--------------------------|-------|----------|----------|------------|-----------------|---------------|
| <input type="checkbox"/> | Timac | PhysAlg | 50,00 KG | R\$ 189,00 | Fert. Químico | N10P16K20 |
| <input type="checkbox"/> | Timac | TOP-PHOS | 50,00 KG | R\$ 128,00 | Fert. Químico | N10P16K20 |
| <input type="checkbox"/> | Bayer | Mythos | 1,00 LT | R\$ 128,00 | Agro. Fungicida | Não informado |

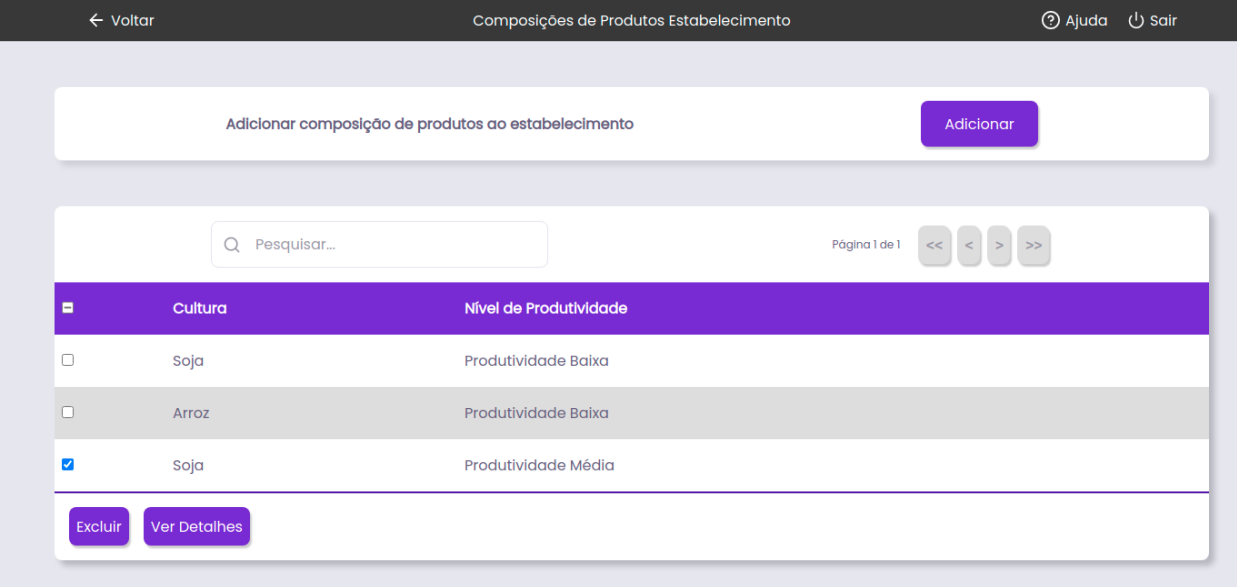
Atualizar **Excluir**

Fonte: Acervo do autor (2021)

Como mencionado anteriormente, o agropecuarista também pode realizar o cadastro de composições no sistema, uma composição pode ser descrita como a união de vários produtos para o manejo de uma cultura. Para realizar o cadastro de composições o usuário deve acessar a funcionalidade “composições”, na tela inicial do sistema. O sistema fará a busca de todas as composições cadastradas, com seu respectivo nível de produtividade e cultura destinada. Também é possível realizar a exclusão ou a visualização destas composições.

Podemos ver através da figura 30, a listagem das composições cadastradas pelo agropecuarista no sistema.

Figura 30 – Listagem de composições do sistema web.



| | Cultura | Nível de Produtividade |
|-------------------------------------|---------|------------------------|
| <input type="checkbox"/> | Soja | Produtividade Baixa |
| <input type="checkbox"/> | Arroz | Produtividade Baixa |
| <input checked="" type="checkbox"/> | Soja | Produtividade Média |

Fonte: Acervo do autor (2021)

Para realizar o cadastro de uma nova composição, basta acessar a funcionalidade “adicionar” acima da listagem, onde o agropecuarista será direcionado a uma tela contendo a listagem de produtos do portfólio da agropecuária, um campo de seleção de cultura e um campo de seleção de nível de produtividade. Cada cultura pode possuir apenas três níveis de produtividade, sendo elas: produtividade baixa, produtividade média e produtividade alta. Fazendo a seleção dessas três informações, cultura, nível de produtividade e produtos, o agropecuarista será direcionado a uma nova tela, onde ele deve informar a quantidade recomendada de cada produto para o cultivo de um hectare de plantio.

Na figura 31, podemos ver os campos onde deve ser informada a quantidade recomendada do uso do produto. O preenchimento de todos os campos é obrigatório para que

seja possível finalizar o cadastro da composição, como pode ser visualizado na figura a seguir.

Figura 31 – Informando a quantidade recomendada de produto na composição.

| | | |
|---|------|--------------------|
| Premier Plus 1,00 LT R\$ 125,00 | 1,50 | LT(s) em 1 hectare |
| Fox 5,00 LT R\$ 1.280,31 | 8,70 | LT(s) em 1 hectare |
| Aureo 5,00 LT R\$ 453,00 | 5,00 | LT(s) em 1 hectare |
| Ampligo 1,00 LT R\$ 238,00 | 5,50 | LT(s) em 1 hectare |
| Elatus 1,00 LT R\$ 185,00 | 0,00 | LT(s) em 1 hectare |

CADASTRAR

Fonte: Acervo do autor (2021)

Após finalizar o cadastro, a nova composição pode ser visualizada através da ação “ver detalhes” na listagem das composições.

Com a figura 32, é possível observar de forma detalhada todas as informações da composição criada.

Figura 32 – Detalhes da composição no sistema web.

| | |
|---|---|
| Cultura: Soja | Total de itens: 8 |
| Nível de produtividade: Produtividade Baixa | Valor da composição: R\$ 5.496,31 |

| Produto | Tamanho | Recomendação para 1 hectare |
|-----------------|--|---|
| TOP-PHOS | Tamanho: 50,00 KG Valor: R\$ 202,00 | Recomendação: 150,00 KG / 3 unidades Valor total: R\$ 606,00 |
| PhysAlg | Tamanho: 50,00 KG Valor: R\$ 189,00 | Recomendação: 170,00 KG / 4 unidades Valor total: R\$ 756,00 |

Fonte: Acervo do autor (2021)

Quanto ao agricultor, o mesmo deve realizar seu cadastro no aplicativo móvel, informando apenas seu nome, e-mail e senha. Ao fazer o cadastro e realizar *login* na aplicação, o agricultor terá acesso a algumas funcionalidades. A principal delas é a funcionalidade para a realização de um novo orçamento, mas também é possível realizar o cadastro de novas áreas e temporadas de plantio, bem como ter acesso a uma aba de ajuda ou histórico de orçamentos. Todas essas funcionalidades serão descritas a seguir.

Através da figura 33, podemos observar as funcionalidades presentes na tela inicial da aplicação, referindo-se à funcionalidade “ajuda”, a mesma não ficou visível na imagem a seguir pois a aplicação possui uma rolagem vertical, já que todos os itens foram dispostos de forma que ocupassem mais do que o tamanho total da tela.

Figura 33 – Tela inicial da aplicação móvel do agricultor.



Fonte: Acervo do autor (2021)

Através da funcionalidade “minhas áreas” o agricultor pode fazer o cadastro de novas áreas de sua propriedade, para isso basta selecionar um local no mapa e informar o nome, descrição e o tamanho da mesma.

Através da figura 34, podemos ver a listagem das áreas de plantio do agricultor, onde no canto inferior da tela existe a opção para criação de uma nova área.

Figura 34 – Listagem de áreas do agricultor em seu aplicativo.




Fonte: Acervo do autor (2021)

Também é possível realizar o cadastramento de temporadas de plantio, acessando a funcionalidade “minhas temporadas” na tela inicial. Para realizar o cadastro de uma nova temporada basta clicar em “nova temporada” na tela de listagem de temporadas, onde o

agricultor é direcionado para um formulário precisando informar o nome, descrição, data de início e fim da temporada. As datas devem ser escolhidas conforme o período de cultivo mais adequado para determinada cultura no decorrer do ano.

Na figura 35 a seguir, podemos observar o calendário onde são selecionadas as datas de início e fim da temporada.

Figura 35 – Cadastro da temporada de plantio.



Fonte: Acervo do autor (2021)

Para realizar o cadastro de um novo orçamento em seu aplicativo, o agricultor deve selecionar a funcionalidade “novo orçamento”, o próximo passo é fazer a seleção de uma área listada no mapa, seguindo para a seleção da cultura que será cultivada na respectiva área. O próximo passo é definir em qual temporada de plantio acontecerá, seguindo para a seleção do

nível de produtividade esperado no final da lavoura. Ao selecionar essas informações a aplicação fará a busca de fornecedores que contenham composições de produtos disponíveis para a cultura e nível de produtividade selecionados. Ao selecionar o fornecedor desejado, o usuário poderá ver os detalhes da composição do mesmo. Os orçamentos serão calculados através do tamanho da área selecionada e o valor dos produtos presentes na composição do fornecedor, onde os mesmos, serão multiplicados pelas recomendações. Dessa forma, se o agricultor possui uma área de dez hectares, e a recomendação de um produto “x” é de 1 litro por hectare, este produto será multiplicado por dez para que seja possível fazer o cálculo do orçamento, e obter-se o valor final.

Na figura 36, é possível observar a disposição das informações de detalhes da composição.

Figura 36 – Detalhes da composição do fornecedor no aplicativo do agricultor.



Fonte: Acervo do autor (2021)

Ao finalizar o orçamento no fornecedor desejado, o agricultor pode fazer a visualização de seus orçamentos através da funcionalidade “meus orçamentos” na tela inicial

da aplicação. Essa funcionalidade tem como foco principal gerar um histórico de orçamentos do agricultor, para que ele possa saber de forma precisa dados de gastos e investimentos feitos em lavouras anteriores.

Dessa forma, conclui-se o desenvolvimento do protótipo de aplicação, onde na próxima sessão serão abordados assuntos referentes as considerações finais do projeto.

5 CONSIDERAÇÕES FINAIS

O presente projeto constitui-se no desenvolvimento de um protótipo de aplicação para gestão orçamentária de lavouras. Pode-se observar com base nas entrevistas e questionamentos bem como na fundamentação descrita, que existem oportunidades para o uso de soluções tecnológicas no campo, e que isso poderá afetar diretamente as próximas gerações, por meio da diminuição e o uso eficiente de matérias primas na produção agrícola.

Com as funcionalidades apresentadas anteriormente, o protótipo foi projetado e poderá aumentar os lucros financeiros, reduzir os gastos de tempo e o desperdício de matérias primas adotadas na produção, trazendo mais eficiência aos agricultores. De outra forma, o projeto também poderá possibilitar melhorias nos serviços e produtos das agropecuárias, aumentando a competitividade no mercado.

As ferramentas e linguagens de programação utilizadas no projeto descritas no trabalho foram essenciais para o desenvolvimento deste protótipo de forma funcional. Com o Javascript, HTML e CSS foi possível desenvolver as camadas visuais da aplicação, também conhecidas como *front-end*. Com o Javascript também foi possível construir a *API* de consumação de dados, essa camada é comumente chamada de *back-end*. Para o armazenamento de dados relacionais o banco de dados PostgreSQL desempenhou seu papel.

Em relação a aplicação do modelo de avaliação heurística de usabilidade, o objetivo foi alcançado, onde o mesmo foi descrito e exemplificado com imagens do protótipo em seu desenvolvimento.

Com base na identificação das oportunidades, a especificação dos recursos necessários, o estudo de tecnologias envolvidas e as facilidades de uso pretendidas, pode-se afirmar que a implementação do protótipo, foi realizada com sucesso.

Desta forma, os objetivos apresentados foram alcançados, contudo, considerando as limitações desta pesquisa, a seguir serão apresentadas algumas recomendações de trabalhos futuros.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Como recomendação de continuidade desse projeto, a implantação de funcionalidades como permitir que o agricultor selecione sua área de plantio no mapa, demarcando o contorno da mesma, traria uma visualização mais detalhada e informações mais precisas em relação ao seu tamanho e localização.

Fazendo uma integração com meios de pagamentos, a ferramenta poderia permitir que o agricultor não só realizasse os orçamentos de seus insumos como também, a compra dos mesmos nas agropecuárias.

Para que o agricultor tivesse maior liberdade na criação de seus orçamentos, seria interessante permitir que o mesmo alterasse os produtos das composições que o sistema gera, essa melhoria poderia aumentar a liberdade e a satisfação do agricultor.

A aplicação móvel poderia contar com funcionalidades que permitissem seu uso em modo *off-line* de forma temporária. Sendo assim, ao estar na lavoura e longe de uma conexão com a internet, o agricultor poderia continuar fazendo o uso da ferramenta.

Referente ao cadastro e manutenção de produtos na agropecuária, a solução poderia ser integrada a base nacional de produtos, conhecida como GS1. Para facilitar a gestão interna da agropecuária, o sistema poderia também, contar com um gerenciador de estoque próprio.

Quanto as notificações, rotinas de cadastro de composições por parte do agropecuarista, e cadastro de orçamentos provindos do agricultor, poderiam exibir notificações, como por exemplo: “nova composição de alta produtividade para a soja foi cadastrada pela agropecuária x”.

Ao administrador do sistema, poderia ser criada uma página *web* onde o mesmo fizesse o cadastro dos produtos, culturas e demais informações que são de sua responsabilidade. Atualmente o mesmo faz o cadastro dessas informações diretamente no banco de dados, o pode dificultar suas atividades.

Em ambos os cenários de uso do sistema, tanto móvel quanto *web*, poderiam apresentar recursos de responsividade, operando em vários tamanhos de tela, sendo que atualmente, o sistema foi pensando somente para telas HD.

REFERÊNCIAS

AEGRO. **Aegro**: evoluir a agricultura é o que nos move. Por isso, somos parceiros de quem produz. 2021. Disponível em: < <https://aegro.com.br/> > Acesso em: 21 mar. 2021.

AMARAL, Luiz Fernando. **Tic e Desenvolvimento Rural**: Impactos da Tecnologia de Informação e Comunicação na Agricultura. 2017. 234 f. Tese (Doutorado) - Curso de Relações Internacionais, Universidade de São Paulo (USP), São Paulo, 2017. Disponível em: <<http://www.funag.gov.br/ipri/btd/index.php/9-teses/4507-tic-e-desenvolvimento-rural-impactos-da-tecnologia-de-informacao-e-comunicacao-na-agricultura>>. Acesso em: 12 set. 2019.

AMCHAM, Campinas. **Agronegócio é responsável por 21% do PIB nacional, afirma especialista da PWC**. Disponível em: <<https://g1.globo.com/sp/campinas-regiao/especial-publicitario/amcham/noticia/agronegocio-e-responsavel-por-21-do-pib-nacional-afirma-especialista-da-pwc.ghtml>> Acesso em: 13 mar. 2019.

BANKS, Alex, PORCELLO, Eve. **Learning React: functional web development with React and Redux**. Gravenstein Highway North, Sebastopol: O'Reilly Media, 2017. *E-Book*.

CARVALHO, Vinicius. **PostgreSQL**: banco de dados para aplicações web modernas. São Paulo: Casa do código, 2017. *E-Book*.

EISENMAN, Bonnie. **Learning React Native: building native mobile apps with Javascript**. Gravenstein Highway North, Sebastopol: O'Reilly Media, 2016. *E-Book*.

FLANAGAN, David. **Javascript**: o guia definitivo. Porto Alegre: Bookman, 2013. *E-Book*

LAMAS, Fernando Mendes. **A tecnologia na agricultura**. Disponível em: < <https://www.embrapa.br/agropecuaria-oeste/busca-de-noticias/-/noticia/30015917/artigo-a-tecnologia-na-agricultura> > Acesso em: 12 set. 2019.

NIELSEN, Jakob. **10 Usability Heuristics for User Interface Design**. Disponível em: <<https://www.nngroup.com/articles/ten-usability-heuristics/>> Acesso em: 13 set. 2019

PAPP, Ana Carolina, CHIARA, Márcia De. **Agronegócio ignora crise e bate recordes**. Disponível em: <<https://economia.estadao.com.br/noticias/geral,nova-noticia,1845680>> Acesso em: 14 de jun. 2019.

PEREIRA, Rogério. **User Experience Design**: como criar produtos digitais com foco nas pessoas. 1. ed. São Paulo: Casa do código, 2018. *E-Book*.

POWERS, Shelley. **Learning Node: moving to the server side**. Gravenstein Highway North, Sebastopol: O'Reilly Media, 2012. *E-Book*

ROB, Peter, CORONEL, Carlos. **Sistema de Banco de Dados**: Projeto, implementação e gerenciamento. 1. ed. São Paulo: Cengage learning, 2014.