UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Guilherme Pretto

DESENVOLVIMENTO EM STARTUP DE UM APLICATIVO MÓVEL PARA CONTROLE DE IRRIGAÇÃO EM PROPRIEDADES RURAIS

Santa Maria, RS
2023

Guilherme Pretto

DESENVOLVIMENTO EM STARTUP DE UM APLICATIVO MÓVEL PARA CONTROLE DE IRRIGAÇÃO EM PROPRIEDADES RURAIS

Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADORA: Profª. Andrea Schwertner Charão
CO-ORIENTADOR: Rômulo Pulcinelli Benedetti

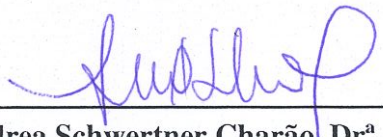
Santa Maria, RS
2023

Guilherme Pretto

DESENVOLVIMENTO EM STARTUP DE UM APLICATIVO MÓVEL PARA CONTROLE DE IRRIGAÇÃO EM PROPRIEDADES RURAIS

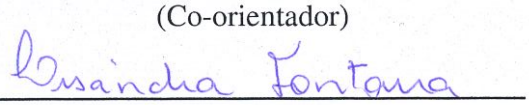
Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 15 de dezembro de 2023:

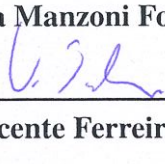


Andrea Schwertner Charão, Drª. (UFSM)
(Presidente/Orientadora)

Rômulo Pulcinelli Benedetti, Bel. (Crops Team)
(Co-orientador)



Lisandra Manzoni Fontoura, Drª. (UFSM)



João Vicente Ferreira Lima, Dr. (UFSM)

Santa Maria, RS
2023

DESENVOLVIMENTO EM STARTUP DE UM APLICATIVO MÓVEL PARA CONTROLE DE IRRIGAÇÃO EM PROPRIEDADES RURAIS

Guilherme Pretto[1], **Andrea Schwertner Charão**[2]

RESUMO

Com o impacto do clima na produtividade agrícola, em relação à falta de chuvas, técnicas como a irrigação se tornam cada vez mais necessárias. A tecnologia contribui automatizando esse processo, também tornando-o mais preciso. Esta área, que associa conhecimentos em Agronomia e Computação, é um dos focos de atuação da empresa CropsTeam, atualmente sediada na Pulsar Incubadora da Universidade Federal de Santa Maria. Em colaboração com a empresa, este trabalho visou desenvolver um aplicativo direcionado ao produtor rural, cliente da empresa, para facilitar o controle do processo de irrigação. O aplicativo utiliza o Framework Flutter e a linguagem Dart, além de integrar-se com o Firebase para gerenciamento de usuários e armazenamento de informações, comunicando-se via API com modelos desenvolvidos pela empresa em Fortran para o cálculo de irrigação. Ele permite ao produtor rural visualizar suas fazendas, talhões e safras, bem como as sugestões de irrigações para cada safra, dando uma visão geral sobre sua estrutura e facilitando o gerenciamento de suas irrigações. Os testes realizados com consultores e produtores rurais, forneceram feedback sobre cada parte do aplicativo, ajudando a empresa a melhorar o resultado final a ser entregue ao cliente.

Palavras-chave: Irrigação. Tecnologia. Desenvolvimento mobile. Flutter.

ABSTRACT

With the impact of climate on agricultural productivity, in relation to the lack of rain, techniques such as irrigation become increasingly necessary. Technology helps by automating this process, and also by making it more precise. This area, which combines knowledge in Agronomy and Computing, is one of the focuses of activity of the company CropsTeam, currently based at the Pulsar Incubadora of the Federal University of Santa Maria. In collaboration with the company, this work aimed to develop an application aimed at rural producers, the company's client, to facilitate control of the irrigation process. The application uses the Flutter Framework and the Dart language, in addition to integrating with Firebase for user management and information storage, communicating via API with models developed by the company in Fortran to calculate irrigation. It allows rural producers to view their farms, plots and crops, as well as irrigation suggestions for each crop, giving an overview of their structure and facilitating the management of their irrigations. Tests carried out with consultants and rural producers provided feedback on each part of the application, helping the company to improve the final result to be delivered to the customer.

Keywords: Irrigation. Technology. Mobile development. Flutter.

[1] Autor: Graduando em Bacharelado em Ciência da Computação pela Universidade Federal de Santa Maria.

[2] Orientadora: Professora do Departamento de Linguagens e Sistemas de Computação da Universidade Federal de Santa Maria.

LISTA DE FIGURAS

LISTA DE TABELAS

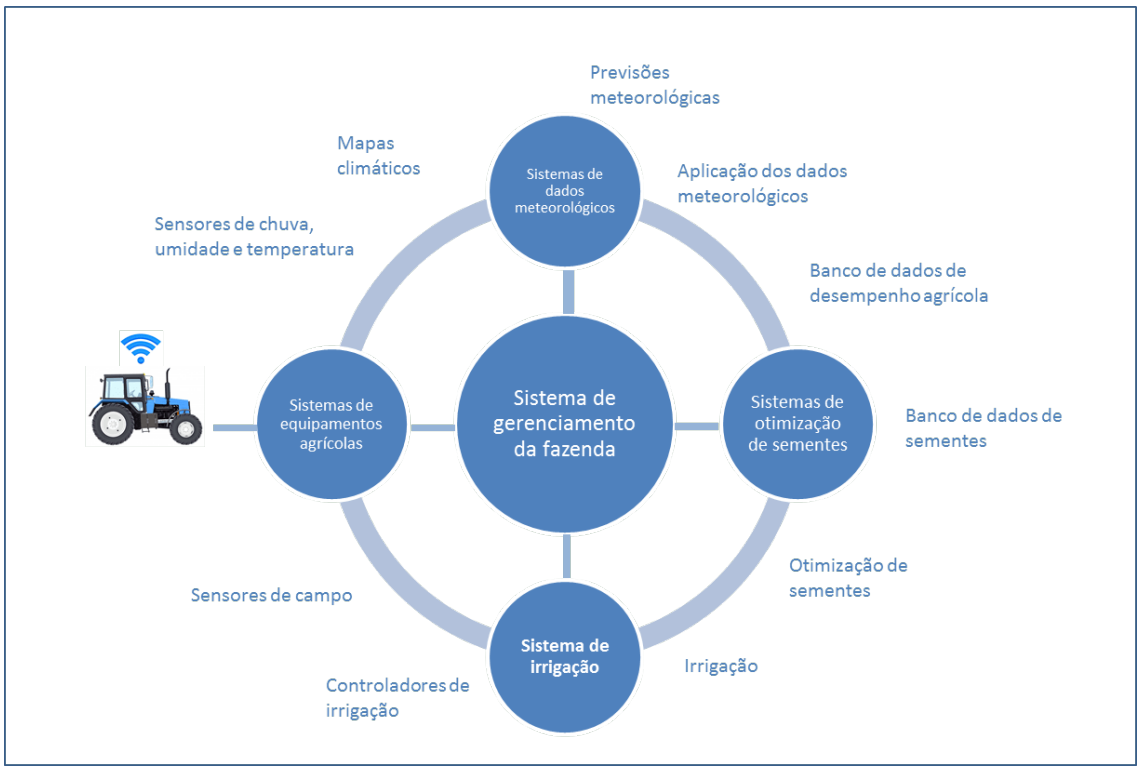
SUMÁRIO

1 INTRODUÇÃO

Com o impacto do clima na produtividade agrícola, principalmente no que diz respeito à falta de chuvas, soluções tecnológicas se tornam cada vez mais necessárias. Visando melhorar a produtividade nas lavouras, uma das estratégias usadas pelos agricultores é a irrigação, visto que a falta de chuvas durante o período de safra é um dos principais problemas e impacta diretamente no produto final (SARTORI et al., 2015).

Na área da irrigação, a tecnologia presente em aplicativos pode mudar a realidade de muitos produtores. Podem ser aplicativos que executam tarefas simples, como visualizar a previsão do tempo em determinada região, até as mais complexas, como automatizar processos de irrigação nas lavouras, conforme ilustrado na Figura 1.

Figura 1 – A Irrigação de Precisão na Agricultura 4.0.



Fonte: Zambon et al., 2019

Para irrigar corretamente, devem ser observadas algumas métricas que impactam em cada cultivar, como umidade do solo, precipitação, entre outras. Esse processo, se feito de forma incorreta, causando excesso ou déficit de irrigação, pode causar danos na produtividade final da lavoura, bem como gastos desnecessários de recursos hídricos (BORTOLUZZI, 2019).

Segundo dados gerados a partir de um censo agropecuário do IBGE, em 2006, a tecnologia foi a responsável por quase 70% do crescimento da produção de grãos, enquanto em 1996, a tecnologia era a responsável por 50% do aumento da produção de grãos (LAMAS, 2017). Além disso, de acordo com (FILHO; GASQUES, 2020), a tecnologia foi responsável por 50,6%, 56,8% e 60,6% do crescimento do valor bruto de produção no anos de 1996, 2006 e 2017, respectivamente. Esses dados enfatizam a

importância da tecnologia para o aumento da produção, onde os resultados são vistos fundamentalmente através do aumento da produtividade.

A partir de ideias e necessidades como essa, foi criada a CropsTeam[3], uma startup incubada na UFSM, composta por mestres e doutores em agronomia, ciências do solo, ecofisiologia, engenharia agrícola, hidrologia, agrometeorologia e ciência da computação, unindo a pesquisa e o conhecimento científico para gerar soluções tecnológicas para produtores rurais e empresas do agronegócio.

Um dos principais projetos da equipe é o WaterCrop, sistema para manejo intensivo e sustentável de irrigação para soja e milho, composto por algoritmos que integram parâmetros de eficiência do uso da água (genético específico) e o potencial de produtividade de cada talhão[4] (ambiente e manejo). Como o WaterCrop utiliza dados das lavouras, principalmente a umidade em cada etapa do manejo, é necessária a presença da equipe no local para coleta dos dados, e, posteriormente, execução do modelo e obtenção de resultados. A maior dificuldade nesse caso é a coleta de dados e também a entrega de resultados aos cliente em cada etapa durante a safra, tendo em vista o deslocamento da equipe a cada lavoura.

A partir disso, em colaboração com a empresa, o objetivo do presente trabalho foi desenvolver um aplicativo mobile para os produtores rurais que aderirem ao projeto WaterCrop, criado pela CropsTeam. O aplicativo tem o intuito de auxiliar os produtores rurais na irrigação da sua lavoura durante o período da safra, visando maior produtividade e uso sustentável da água. Com o mesmo, o produtor pode visualizar suas fazendas, talhões, safras e a quantidade de água que deve ser irrigada em cada talhão para cada período de desenvolvimento da planta, sem ser necessária a presença de um técnico na lavoura a cada etapa da irrigação. Dessa forma, o aplicativo facilita tanto o agricultor na obtenção de dados para sua lavoura, quanto a assistência técnica realizada pela equipe da CropsTeam, possibilitando o auxílio a mais produtores, de forma mais autônoma e com qualidade.

O restante deste texto está organizado da seguinte forma: no Capítulo 2, apresenta-se uma fundamentação para o trabalho e discute-se trabalhos relacionados; no Capítulo 3, apresenta-se uma visão geral do projeto do WaterCrop e do aplicativo, seguido pelo capítulo de desenvolvimento (Capítulo 4); após, segue o Capítulo 5, no qual são apresentados os testes com os usuários; ao final, o Capítulo 6 apresenta a conclusão do trabalho.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

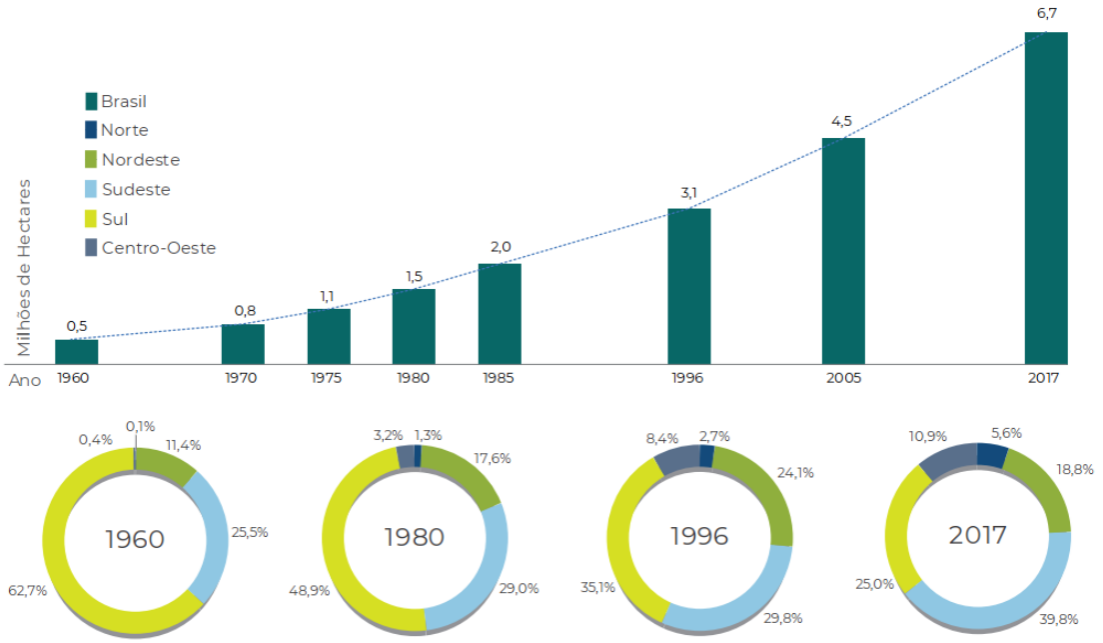
Neste capítulo, é apresentada a fundamentação teórica para o trabalho, apresentando o contexto e embasamento sobre o processo de irrigação, desenvolvimento mobile e multiplataforma. Também é feita uma comparação entre os frameworks Flutter e React Native, pois os dois foram analisados para serem utilizados no desenvolvimento do aplicativo. Como optou-se pelo Flutter, é feita uma análise mais a fundo sobre sua estrutura. Por fim, são apresentados trabalhos relacionados, sejam trabalhos mais abrangentes em relação à agricultura, bem como trabalhos mais voltados à irrigação.

[3] https://www.cropsteam.com/
[4] Lavoura que pertence à fazenda

2.1 Irrigação

Por ser uma das práticas mais antigas realizadas pelo ser humano, o cultivo de plantações sempre recebeu o auxílio de estratégias para aumento da quantidade e qualidade da produção. À medida que o tempo foi avançando, os recursos aumentaram e, principalmente, o conhecimento, ou seja, a tecnologia. Uma das estratégias usadas que se aprimorou muito com o avanço da tecnologia foi a irrigação, como mostra a Figura 2, evidenciando a evolução na área irrigada com o passar dos anos.

Figura 2 – Evolução da Área Irrigada no Brasil.



Fonte: Censos Agropecuários (IBGE, 1960-2017)

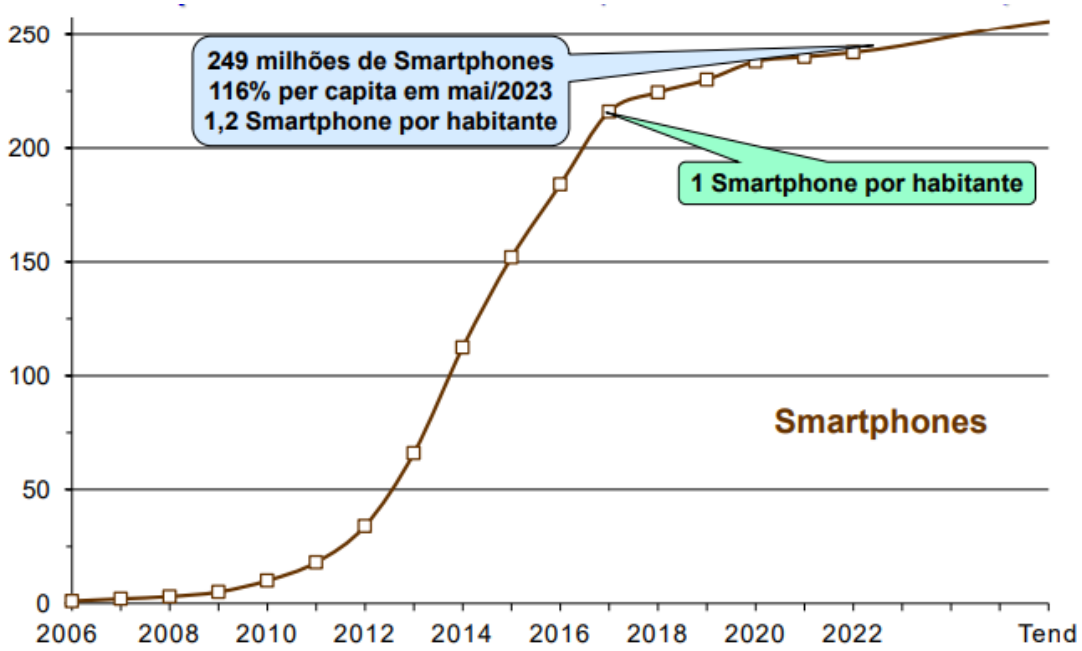
Basicamente, irrigação consiste em técnicas ou meios de aplicar água às plantas de forma artificial, com o intuito de suprir as necessidades das mesmas e, principalmente, aumentar e melhorar a produção final. Pode ser usada em qualquer tipo de região, desde regiões com baixo índice de chuvas anuais, onde a irrigação é fundamental para o desenvolvimento das plantas, até em locais onde os índices pluviométricos são bons, mas que sofrem com um deficit de precipitaçãoque em certas etapas do ano (TESTEZLAF, 2017).

Sabe-se que a falta água é um dos principais riscos que causam prejuízo aos produtores. Em algumas regiões mais áridas, o controle da irrigação deve ser feito principalmente porque o recurso hídrico é muito escasso e pode vir a faltar até mesmo para consumo humano. Além disso, até mesmo em regiões onde a falta de água não é problema, em anos de estiagem, ou seja, quando as chuvas ocorrem em menor quantidade, as irrigações também devem ser bem gerenciadas, a fim de evitar a falta do recurso hídrico. Por outro lado, ao contrário da falta, o excesso de água também pode prejudicar os resultados na hora da colheita. Ou seja, não existe um padrão no ponto de vista da irrigação, cada região possui uma necessidade e essa necessidade pode mudar a cada ano.

2.2 Desenvolvimento Mobile

Nos últimos anos, dispositivos móveis popularmente conhecidos como *smartphones* se desenvolveram tecnologicamente muito rápido. A cada ano surgem novas versões de aparelhos, com novas tecnologias e funcionalidades. Segundo pesquisa de Meirelles (2023), o Brasil possui atualmente mais de 1 smartphone por habitante, totalizado, em 2022, 249 milhões de aparelhos, como pode ser observado na Figura 3. Isso equivale a mais de 51% dos dispositivos digitais (computadores e smartphones) do Brasil, que somou mais de 464 milhões de dispositivos em maio de 2023.

Figura 3 – Smartphones em uso no Brasil (milhões de unidades - FGVcia).



Fonte: Pesquisa do Uso da TI - Tecnologia de Informação nas Empresas

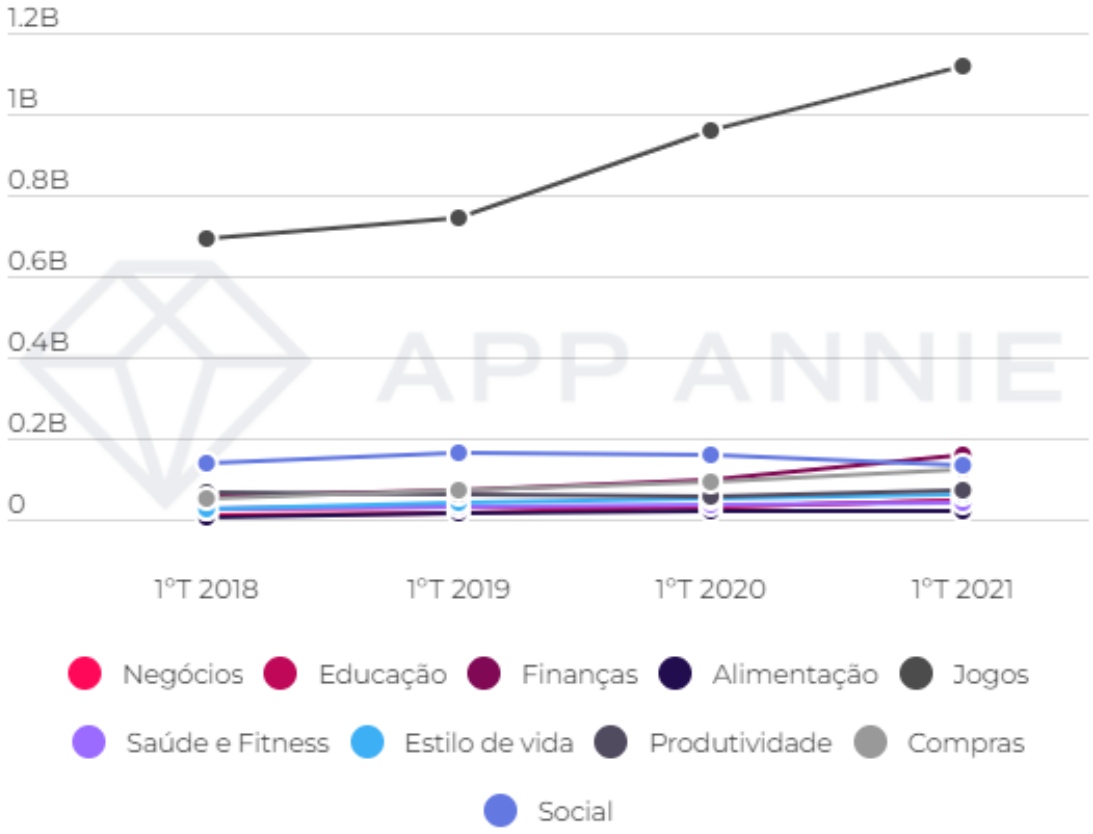
Um dos principais atrativos dos smartphones está nos inúmeros aplicativos. Eles vão desde jogos e redes sociais nas áreas de entretenimento, até aplicativos voltados a área da saúde, trabalho, produtividade, etc. (Figura 4). De acordo com dados trazidos pela AppsFlyer juntamente com a App Annie, foram mais de 700 milhões de instalações de aplicativos no primeiro trimestre de 2019 a 2021, tendo as categorias de jogos, finanças e negócios com uma demanda maior.

Como a demanda aumenta cada vez mais, tornando-se inevitável que as informações cheguem de forma rápida na tela do celular, o desenvolvimento dessas ferramentas também cresceu muito, transformando a programação mobile uma das áreas que mais cresceu nos últimos anos no mercado. Segundo dados da GeekHunter[5], as buscas por desenvolvedor mobile tiveram um aumento de 600% no primeiro semestre de 2021, em relação ao mesmo período no ano de 2020 (CUBOS ACADEMY, 2021).

O desenvolvimento mobile é um processo especializado em que aplicações são concebidas para execução em pequenos dispositivos móveis (SANTOS et al., 2015).

[5] https://www.geekhunter.com.br/

Figura 4 – Finanças, Negócios e Educação tiveram o maior aumento em demanda no 1ºT de 2021.



Fonte: App Annie Intelligence

Essas aplicações podem vir pré-instaladas nos celulares – são os chamados aplicativos nativos, que são específicos para cada plataforma, sendo necessário serem adaptados a cada aparelho. Outro tipo são os aplicativos baseados na web, ou seja, são aqueles que necessitam de um navegador no celular para poderem executar. Além disso, existem os aplicativos híbridos, que têm características semelhantes aos nativos, porém já são adaptados para executar em diferentes plataformas (KAPOS, 2021).

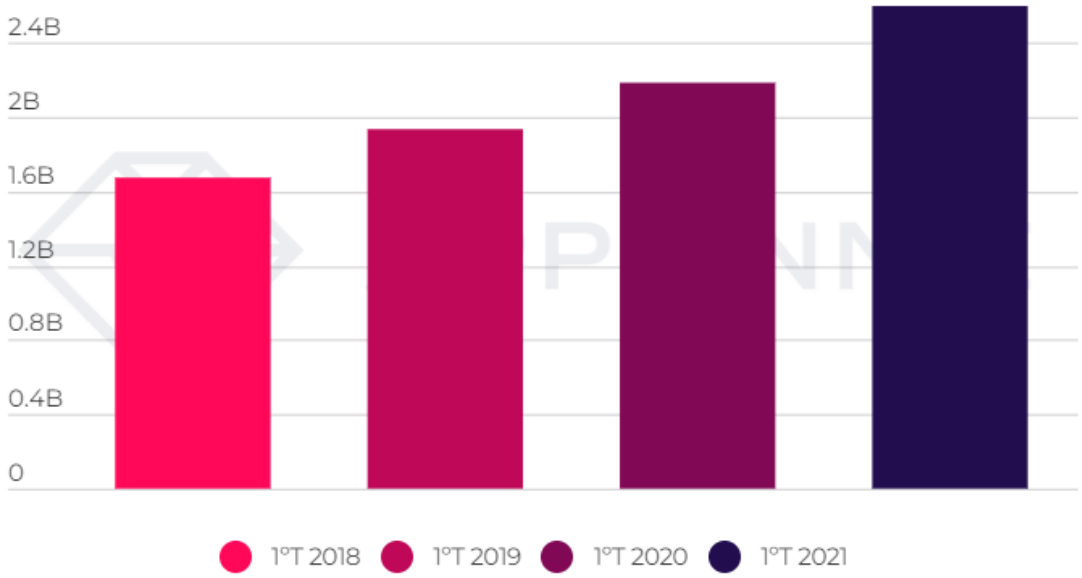
2.3 Desenvolvimento Multiplataforma

Com a popularidade dos aplicativos móveis, a demanda de desenvolvimento aumentou muito nos últimos anos. Segundo pesquisas feitas pela AppAnnie[6], plataforma de dados e análise mobile, em 2021 o download de apps chegou a 2,6 bilhões no primeiro trimestre, isso aumentando 55% em relação a 3 anos atrás, considerando as lojas do iOS e Google Play, como evidencia a Figura 5.

Em paralelo ao crescimento no número de aplicativos, também aumenta a diversidade de aparelhos e plataformas. De acordo com Correa e Pinto (2020), dois eventos ocorridos em 2007 e 2008 tiveram grande contribuição para a indústria da tecnologia e principalmente nas mudanças no modo de comunicação da população. Em 2007 houve

[6] https://www.data.ai/en/go/state-of-mobile-2023/

Figura 5 – Downloads do iOS e Google Play no Brasil



Fonte: App Annie Intelligence

o lançamento do primeiro iPhone, pela Apple. No ano seguinte, o primeiro smartphone com o sistema operacional Android, da Google. Com essas novas tecnologias no mercado, surgiram também novos problemas, visto que os sistemas operacionais dos aparelhos da Apple e da Google utilizam arquiteturas diferentes.

Dessa forma, tornou-se necessário o desenvolvimento de aplicações separadas e individuais para cada sistema, demandando mais custos e, principalmente, mais tempo, devido a todo o planejamento necessário para poder englobar as duas plataformas. Como alternativa, e para tentar amenizar o problema, surgiram os aplicativos híbridos. Basicamente, eles são desenvolvidos com frameworks específicos que permitem criar apenas uma aplicação que pode ser usada em diversos dispositivos (CHARLAND; LEROUX, 2011).

Dentre os frameworks mais conhecidos e utilizados na atualidade, estão o Flutter[7] e o React Native[8]. Com o Flutter, é possível desenvolver um aplicativo que funcionará em qualquer tela, podendo ser executado tanto em dispositivos móveis, como aplicações desktop[9]. Já com o React Native, uma versão do framework React que aproveita a implementação da biblioteca de interface do usuário JavaScript, é possível a execução em qualquer dispositivo móvel (CONSULTING, 2022).

2.4 Flutter x React Native

Escolher uma linguagem de programação, plataforma ou framework vai muito além de apenas comparar as opções e decidir entre uma delas. Na grande parte das vezes, o ponto de vista do produto a ser desenvolvido é que define a melhor opção dentre as disponíveis. No presente trabalho, a necessidade de escolha se dá por alguns requisitos

[7] https://flutter.dev/
[8] https://reactnative.dev/
[9] https://flutter.dev/multi-platform

definidos pela equipe da empresa CropsTeam, a saber: um aplicativo que fique disponível tanto iOS quanto Android, que seja de desenvolvimento simples e rápido, com integração com apenas alguns recursos nativos do aparelho como o giroscópio e também com o Firebase para armazenamento de dados. Diante disso, a seguir são discutidos alguns pontos de comparação entre os dois frameworks, que foram estudados durante este trabalho e culminaram com a escolha do Flutter para o desenvolvimento do aplicativo.

Iniciando pela linguagem por trás de cada framework, do lado do Flutter temos a linguagem Dart[10], uma linguagem multiparadigma, multiplataforma e orientada a objetos, lançada pela Google, em 2011. Tendo uma sintaxe que se assemelha com a da linguagem C, ela permite criar soluções tanto mobile, quanto desktop e web (SILVA, 2023). Já no React Native, a linguagem base é o JavaScript, que é uma linguagem voltada para a criação de aplicativos web, interpretada e orientada a objetos, conhecida como a linguagem de scripting para páginas web e que também está disponível para outros ambientes (FOUNDATION, 2023).

Quando a métrica de comparação é desempenho, Flutter pode ser mais vantajoso à orimeira vista, pois é compilado em bibliotecas nativas para ARM ou x86, enquanto o React Native não é compilado para código nativo e, além disso, passa por uma camada JavaScript, que pode diminuir sua performance (KHAN, 2021). Porém, segundo (TANAKA, 2023), executando testes de desempenho com algoritmos de busca, foram obtidos resultados onde o React Native obteve melhor desempenho em 75% dos casos testados. Dessa forma, a questão desempenho pode variar muito para cada situação e não ser ponto principal na escolha de um ou outro framework.

Do ponto de vista da arquitetura, segundo Khan (2021), Flutter possui um SDK avançado, o que permite uma arquitetura de camadas, garantindo uma alta personalização, enquanto o React Native depende de outro software para construir seus componentes, além de utilizar JavaScript como ponte. Com a arquitetura de camadas do Flutter, é possível realizar grandes personalizações, de forma fácil e rápida.

Em relação aos componentes de UI e visualização, no Flutter a principal estrutura que controla todo o desenvolvimento são os Widgets, que basicamente são compostos por outros Widgets, ou seja, eles formam uma estrutura hierárquica, o que é chamado de Widget Tree, que possui um nó pai, o qual normalmente é MaterialApp ou CupertinoApp. Analisando o React Native, ele possui estruturas básicas adaptativas à plataforma, chamados de React Elements, como view, button, text, entre outros. Além das estruturas básicas, ele possui muitos componentes de terceiros, que ajudam a personalizar a interface do app (FREITAS, 2022).

2.5 Framework Flutter

Esta seção traz uma visão mais aprofundada sobre o framework Flutter, escolhido para este trabalho. Flutter foi lançado em meados de 2017 pela empresa Google, apresentando-se como um framework de código aberto para a criação de aplicativos móveis nativos para múltiplas plataformas, a partir de uma única base de código. Ele rapidamente tornou-se amplamente utilizado para desenvolver aplicativos para iOS, Android, web e desktop. Utiliza a linguagem de programação Dart, que foi também desenvolvida pela Go-

[10] https://dart.dev/

ogle. Embora o Dart possa ser menos conhecido do que JavaScript, ele é fácil de aprender e oferece recursos como tipagem estática que ajudam a evitar erros de código (CAPPELLI, 2018). De acordo com a própria documentação do Flutter, ele é um kit de ferramentas de UI multiplataforma, que é projetado para permitir a reutilização de código em sistemas operacionais como iOS e Android, enquanto permite que os aplicativos interajam com os serviços da plataforma subjacente.

Flutter foi projetado para ser um sistema extensível em camadas, como mostra a Figura 6. Seu funcionamento se dá a partir de bibliotecas que funcionam separadamente e independentes, dependendo apenas da camada anterior. O sistema operacional subjacente trata os aplicativos Flutter da mesma forma que trata os aplicativos nativos convencionais. Para isso, utiliza-se um integrador específico da plataforma, que serve como ponto de entrada, coordena a interação com o sistema operacional para acessar recursos como renderização, acessibilidade e entrada, e gerencia o ciclo de eventos de mensagens. Cada integrador é desenvolvido na linguagem mais adequada para a plataforma específica, como Java e C++ para Android, Objective-C/Objective-C++ para iOS e macOS, e C++ para Windows e Linux. Com a ajuda desse integrador, o código Flutter pode ser incorporado a um aplicativo existente como um módulo adicional ou até mesmo constituir o conteúdo completo do aplicativo. O Flutter inclui diversos integradores para plataformas populares, mas também é possível encontrar outros integradores disponíveis.

As subseções seguintes descreverão melhor sua linguagem, engine, interface e seus widgets, de acordo com a documentação fornecida pelo Flutter e também no site da linguagem Dart (FLUTTER, 2023).

2.5.1 Linguagem Dart

Dart é uma linguagem otimizada para desenvolver aplicativos rápidos em qualquer plataforma. Seu objetivo é oferecer a linguagem de programação mais produtiva para desenvolvimento multiplataforma, combinada com uma plataforma de execução flexível para estruturas de aplicativos. Nos dias atuais, o principal uso da linguagem Dart se dá nos aplicativos Flutter.

A linguagem Dart é segura quanto a tipos; ela utiliza verificação de tipo estático para garantir que o valor de uma variável sempre corresponda ao tipo estático da variável. Embora os tipos sejam obrigatórios, as anotações de tipo são opcionais devido à inferência de tipos. O sistema de tipagem do Dart também é flexível, permitindo o uso de um tipo dinâmico combinado com verificações em tempo de execução, o que pode ser útil durante experimentação ou para código que precisa ser especialmente dinâmico.

Outro aspecto importante é que o Dart possui nulidade segura integrada. Isso significa que os valores não podem ser nulos, a menos que isso seja definido explicitamente. Com a nulidade segura, o Dart pode proteger o código contra exceções de nulidade em tempo de execução, por meio de análise de código estático. Ao contrário de muitas outras linguagens seguras quanto a nulos, quando o Dart determina que uma variável é não nula, essa variável nunca pode ser nula.

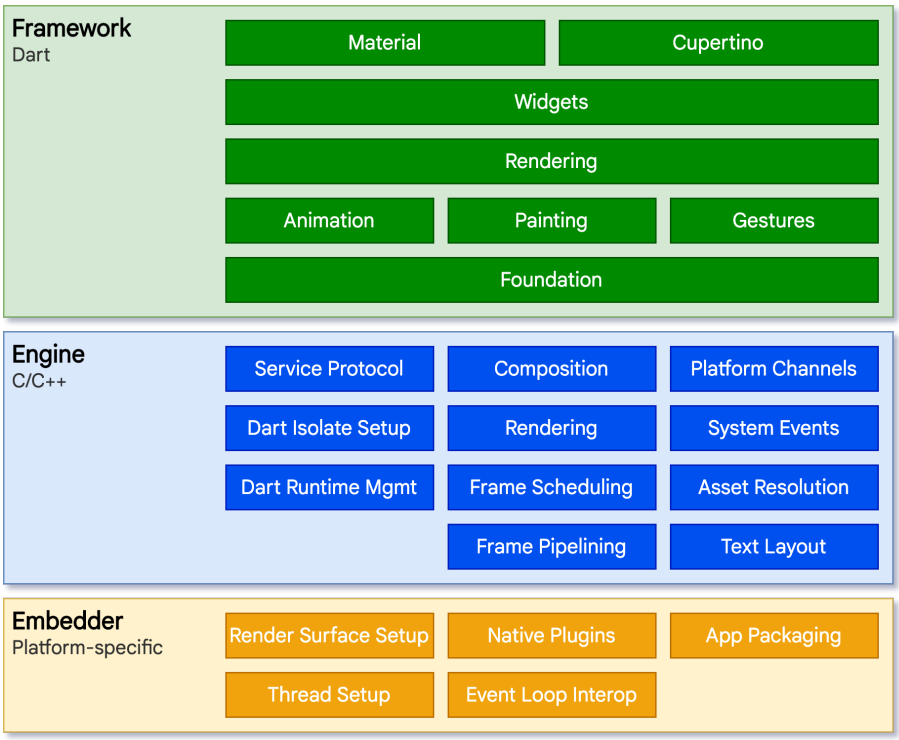
Um ponto que também merece destaque é o conjunto de bibliotecas fornecidas pela linguagem, que executam muitas funções comuns no dia a dia da programação. São bibliotecas como a core, collection, convert, math, io, async, entre outras. Elas fornecem

tipos integrados, coleções básicas e mais elaboradas como filas, listas vinculadas, hash-maps e árvores binárias, codificadores, decodificadores, funções matemáticas, arquivos, programação assíncrona, entre muitas outras funcionalidades.

2.5.2 Engine

No centro do Flutter está a engine. Escrita principalmente em C++, ela oferece suporte aos elementos essenciais necessários para garantir o funcionamento de todos os aplicativos Flutter. Esse mecanismo desempenha a função de renderizar as cenas compostas sempre que é necessário exibir um novo quadro na tela. Também abrange a implementação de baixo nível da principal API do Flutter, envolvendo gráficos (utilizando o Impeller no iOS e se integrando ao Android, e o Skia em outras plataformas), layout de texto, E/S de arquivo e rede, acessibilidade, arquitetura de plug-ins e um ambiente de execução Dart, além do conjunto de ferramentas de compilação. A biblioteca dart:ui, é a responsável por interligar a estrutura da engine com a do Flutter. Ela encapsula o código C++ subjacente em classes Dart. Além disso, expõe os elementos de nível mais baixo, como classes para interagir com subsistemas de entrada, renderização gráfica e renderização de texto.

Figura 6 – Camadas da Arquitetura Flutter



Fonte: Flutter

2.5.3 Interface

De acordo com a própria documentação, o Flutter é uma estrutura de desenvolvimento de interfaces de usuário que segue uma abordagem declarativa e reativa. Os desenvolvedores definem como o estado do aplicativo se relaciona com a interface do

usuário, e o Flutter se encarrega automaticamente de atualizar a interface quando o estado do aplicativo muda. Essa abordagem foi inspirada no React, framework desenvolvido pelo Facebook, e visa superar os desafios encontrados em estruturas de interface de usuário tradicionais.

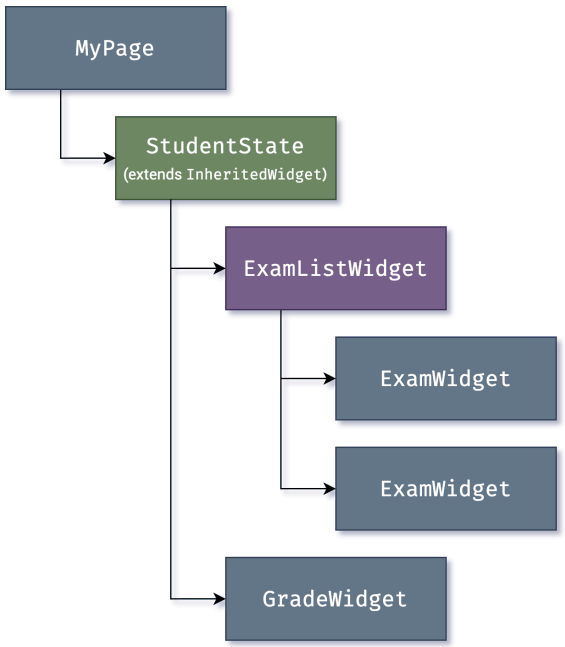
Em estruturas tradicionais, a interface do usuário é definida uma vez e depois atualizada manualmente pelo código do desenvolvedor em resposta a eventos. Isso pode se tornar complexo quando as mudanças de estado precisam ser refletidas em toda a interface do usuário. O Flutter resolve esse problema desacoplando a interface do usuário do estado subjacente. Ele utiliza widgets para criar uma descrição da interface e, em seguida, a estrutura se encarrega de atualizar a interface de acordo com essa descrição.

O Flutter adota uma abordagem eficiente, utilizando árvores de objetos para representar widgets e gerenciar o layout e a composição da interface. O método build() é fundamental, pois é chamado pela estrutura sempre que necessário para atualizar a interface do usuário. Além disso, o Flutter se beneficia das características de execução da linguagem Dart para garantir um desempenho eficiente, tornando-o adequado para o desenvolvimento de aplicativos de alto desempenho.

2.5.4 Widgets

Widgets são basicamente blocos de construção da interface. Eles formam uma hierarquia, encapsulando-se um dentro do outro e podendo receber o contexto do seu Widget pai, como ilustra a Figura 7. Os widgets podem alterar a aparência da sua interface em resposta a eventos, como interações do usuário, por exemplo, solicitando à estrutura que substitua um widget por outro na hierarquia. A atualização é feita por meio da comparação do Widget antigo com o novo, tornando-a mais eficiente.

Figura 7 – Árvore de Widgets



Fonte: Flutter

Existem vários tipos de Widgets. A ideia por trás deles é que sejam pequenos e principalmente, combináveis. Os principais e, de certa forma, "maiores", costumam ser mais abstratos, incluindo partes básicas, como alinhamento e cor. Essas partes básicas são preenchidas com outros Widgets simples, criados para executar cada função específica.

2.6 Trabalhos relacionados

Nesta seção, exploraremos uma seleção de trabalhos relacionados que abordam tópicos semelhantes ou complementares ao projeto: IZagro, SmartPivo e IrrigaFácil. Esses trabalhos abrangem desde aplicativos voltados à área da agricultura, sejam relacionados à irrigação ou até mesmo com o intuito de levar mais informações aos produtores, além de plataformas ligadas inteiramente ao processo de irrigação. A Tabela 1, mais adiante, sintetiza estes trabalhos comparativamente ao WaterCrop.

Fundada em 2016 por engenheiros agrônomos, administradores e graduados em ciência da computação, a IZagro[11] é uma plataforma, disponível tanto para Web quanto para dispositivos mobile, que leva informações direto para o agricultor, de maneira gratuita e rápida. Com o aplicativo, o agricultor pode acessar informações sobre insumos agrícolas, pragas, doenças e também buscar insumos para a compra, de acordo com a doença encontrada na propriedade (STARTAGRO, 2018). Segundo dados do próprio site, o aplicativo conecta, atualmente, mais de 65 mil usuários, tornando o acesso a informações mais democrático a fim de tornar a agricultura mais rentável e sustentável. A partir dessas informações, podemos ver que tanto a IZagro quanto o WaterCrop visam melhorar e aumentar o acesso à informação para o agricultor durante a safra, mas de formas diferentes, onde a IZagro é focada no processo produtivo em geral e o WaterCrop diretamente e exclusivamente focado na irrigação.

SmartPivo é um aplicativo móvel desenvolvido em linguagem de programação voltada a web, juntamente com o framework Cordova, permitindo a criação de um aplicativo multiplataforma, para iOS, Android, entre outros. Com o aplicativo, é possível informar dados meteorológicos, topográficos e da cultura e obter o dimensionamento do equipamento de irrigação do tipo pivô central necessário para irrigar determinada área, além da vazão de água para a área em questão (OLIVEIRA; OLIVEIRA; FIGUEIREDO, 2018). Dessa forma, o SmartPivo tem um viés relacionado à estrutura necessária para que a irrigação seja efetuada da melhor maneira possível, enquanto o WaterCrop tem a ideia de gerenciar a irrigação durante a safra. Os dois aplicativos poderiam ser usados em conjunto, em diferentes etapas, o SmartPivo sendo útil no início da safra para montar a estrutura de irrigação e, posteriormente, o WaterCrop para o gerenciamento da mesma.

Com o foco inteiramente no manejo da irrigação, o IrrigaFácil é um aplicativo computacional criado com a finalidade de gerar dados preditos confiáveis de evapotranspiração de referência (ETo), a partir de dados de série histórica de variáveis climáticas, com o intuito de obter um simples calendário de irrigação que mostra datas e quantidade de água a ser irrigada. Se durante o período estipulado pelo calendário de irrigação ocorrerem chuvas, um novo calendário pode ser gerado, computando-se os eventos de precipitação. Além disso, outros critérios podem ser adotados para esse manejo, como medir ou estimar dados de ETo por qualquer método para acompanhar a irrigação dia a dia (ALBUQUER-

[11] https://izagro.com.br/sobre-nos

QUE; FARIA; COELHO, 2011). Assim como o WaterCrop, o IrrigaFácil é voltado para o gerenciamento de irrigações para uma determinada safra. No entanto, os dois possuem algumas diferenças, como a plataforma em que são disponibilizados, pois o IrrigaFácil é um software para computadores, enquanto o WaterCrop é direcionado para o mercado mobile. Além disso, no IrrigaFácil, são usadas séries temporais, com dados pluviométricos de anos anteriores e inteligência artificial para a geração do calendário de irrigações, enquanto no WaterCrop, os cálculos são baseados nos dados climáticos e do solo, da safra em questão, fornecendo irrigações futuras e se adaptando às mudanças climáticas.

	IZagro	SmartPivo	IrrigaFácil	WaterCrop
Plataforma	Web, Mobile	Mobile	Desktop	Mobile
Área de atuação	Processo produtivo de soja em geral	Criação da estrutura para o processo de irrigação	Gerenciamento do processo de irrigação	Gerenciamento do processo de irrigação
Benefícios	Leva informação sobre doenças e insumos agrícolas	Ajuda na escolha de Pivôs para irrigação	Fornece calendário de irrigação a partir de dados históricos	Fornece sugestão de irrigação a partir de dados meteorológicos e da cultivar

Tabela 1 – Comparativo de trabalhos relacionados

3 PROJETO WATERCROP

Este capítulo visa apresentar mais a fundo tanto a ideia do WaterCrop como um todo, explicando sua estrutura e finalidade, quanto o aplicativo, mostrando uma visão geral de suas funcionalidades, bem como a arquitetura, requisitos e ferramentas utilizadas durante o desenvolvimento.

3.1 WaterCrop

Pensado e desenvolvido pela CropsTeam, o WaterCrop é um sistema para controle de irrigação para as culturas de soja e milho. Com ele, é possível que o produtor obtenha a melhor quantidade de água a ser irrigada na sua lavoura, no estágio atual da cultura que está sendo produzida, visando maior produtividade e principalmente, fazer um bom uso da água, evitando gastos em excesso. Pensando de uma maneira prática, é comum que uma fazenda possua vários talhões, que são as lavouras que compõem a fazenda. Cada talhão terá uma cultivar sendo produzida em cada etapa do ano, o que é chamado de safra. Com o WaterCrop, é possível fazer o gerenciamento da irrigação de cada safra em cada talhão de uma fazenda. Esse controle é feito separadamente para cada talhão, pois é muito comum que os talhões possuam diferentes datas de plantio, bem como diferentes propriedades do solo, umidade e também dados meteorológicos.

Durante o período de uma safra, a cultivar passa por vários estágios, que são tratados pelo modelo do WaterCrop de diferentes maneiras, retornando valores específicos a serem irrigados em cada etapa. Dessa forma, o modelo que calcula as irrigações necessita inicialmente das informações básicas referentes à safra, como cultura, data da semeadura, localização, dados sobre o solo, tais como profundidade da raiz e também sobre a cultivar semeada. Além disso, uma parte muito importante e necessária para os modelos são os dados climáticos, que devem ser referentes a cada dia da safra em andamento e para cada talhão, pois seus valores podem variar. São informações ligadas à temperatura mínima, máxima, umidade, radiação solar e, principalmente, quantidade de água precipitada, seja ela a partir das chuvas ou das irrigações.

Com essas informações, o modelo retorna valores a serem irrigados diariamente, de acordo com a quantidade de dados meteorológicos diária informada, além de informações como deficit hídrico, entre outras. É importante ressaltar que os valores referentes às quantidades irrigadas podem variar de acordo com as mudanças climáticas. Por exemplo, quando era prevista uma quantidade de chuva para um dia e na prática ocorreu uma precipitação maior ou menor, esses dados devem ser informados ao modelo que corrigirá a quantidade a ser irrigada.

Um ponto importante a ser destacado é a finalidade do uso do WaterCrop. Além de visar melhorar e aumentar a produtividade das safras, um bom gerenciamento das irrigações favorece o uso mais adequado da água, recurso hídrico tão importante para a vida e que tem se tornado cada vez mais escasso.

3.2 Visão geral do aplicativo

Conforme já mencionado, o objetivo deste trabalho é desenvolver um aplicativo que facilite o controle do processo de irrigação tanto para os agricultores quanto para os técnicos da CropsTeam. A partir dele, o agricultor terá com mais facilidade as informações referentes às irrigações das safras de cada talhão de sua fazenda. Para os consultores da equipe da CropsTeam, o aplicativo será útil pois expande o poder de consultoria para áreas maiores e mais clientes.

No aplicativo, o cliente faz o login em sua conta e pode acessar suas fazendas, talhões e safras de cada talhão. Na safra, serão listadas as irrigações sugeridas para cada dia, de acordo com os dados meteorológicos cadastrados. É possível também que o produtor modifique as informações de cada irrigação, alterando a quantidade de chuva e confirmando se uma irrigação sugerida foi realmente efetuada, lembrando que cada alteração feita pode acarretar em mudanças nas sugestões futuras. A ideia é que o produtor tenha o menor trabalho possível para poder visualizar as irrigações sugeridas. O trabalho de cadastrar fazendas, talhões, safras e informar os dados meteorológicos ficará a cargo do consultor, que inicialmente fará o cadastro dessas informações diretamente no banco de dados, pois boa parte dos dados são informações técnicas sobre as quais o produtor não possui conhecimento.

3.3 Estrutura do aplicativo

Após a etapa inicial de estudos sobre os frameworks React Native e Flutter e criação dos protótipos utilizando as duas plataformas, foi optado por utilizar o Flutter para

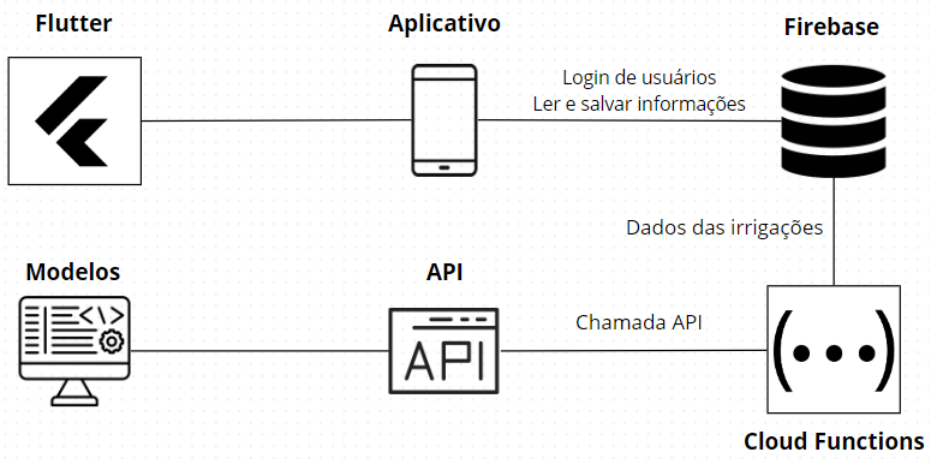
construção do aplicativo, juntamente com o Firebase como banco de dados. Dessa forma, o aplicativo se conecta ao Firebase para realizar a autenticação de usuário, bem como leitura e gravação de dados referentes às irrigações.

Além do banco de dados, a estrutura mais importante para o funcionamento do aplicativo são os modelos de dados desenvolvidos em Fortran pela CropsTeam. É nele que são feitos os cálculos para sugestão de irrigações a partir das informações fornecidas. Para a conexão desses modelos com o aplicativo, foi desenvolvida também pela CropsTeam uma API, por meio da qual é possível fazer requisições informando os dados da safra e meteorológicos, tendo como retorno as sugestões de irrigações a serem realizadas.

Para facilitar a conexão com a API e abstrair essa função do lado do cliente, optou-se por utilizar o Firebase Cloud Functions, que é um ambiente de execução, sem servidor, para criar e conectar serviços em nuvem. Ele permite executar código de back-end a partir de eventos acionados por recursos do Firebase. Dessa forma, é possível criar um código JavaScript ou TypeScript que será armazenado na nuvem do Google e executado em um ambiente gerenciado.

Na Figura 8 é possível visualizar a estrutura final completa.

Figura 8 – Estrutura final do aplicativo



Fonte: Próprio Autor

3.4 Metodologia Scrum

Como metodologia para desenvolvimento de todo o projeto, optou-se pelo uso da metodologia ágil denominada Scrum. Nessa metodologia, o projeto é dividido em requisitos e tarefas, onde essas tarefas são divididas em períodos curtos que podem variar de duas a quatro semanas. No caso do aplicativo do WaterCrop, foi utilizado um período mais curto, realizando reuniões semanais. Nas reuniões eram discutidas as tarefas a serem desenvolvidas na semana seguinte e revisadas as tarefas entregues na semana anterior.

Este processo ajudou a estruturar o projeto e manter um controle do tempo para as tarefas, mas ao mesmo tempo permitia flexibilidade caso surgissem empecilhos que bloqueassem o desenvolvimento de alguma delas.

Durante o desenvolvimento, o processo foi dividido em algumas etapas maiores, as quais eram divididas em tarefas, distribuídas durante as sprints semanais. Na primeira etapa, o foco principal foi estruturar o processo de autenticação de usuários. Após isso, foi voltada a atenção para leitura e escrita de dados no Firebase. Como próximo passo, foi feita a conexão com a API. Ao final, destinou-se um tempo para ajustes de layout e visualização do aplicativo.

3.5 Requisitos do aplicativo

Sobre um sistema ou aplicativo, seus requisitos são basicamente a descrição do que o mesmo é capaz de fazer, quais os serviços que ele oferece, bem como suas restrições de funcionamento para cada usuário. Os requisitos têm a finalidade de refletir as necessidades dos clientes dentro do sistema, como enviar ou receber uma informação ou controlar um dispositivo. Para o aplicativo do WaterCrop, foi elaborada uma lista de requisitos para especificar as funcionalidades necessárias do aplicativo e facilitar o processo de desenvolvimento.

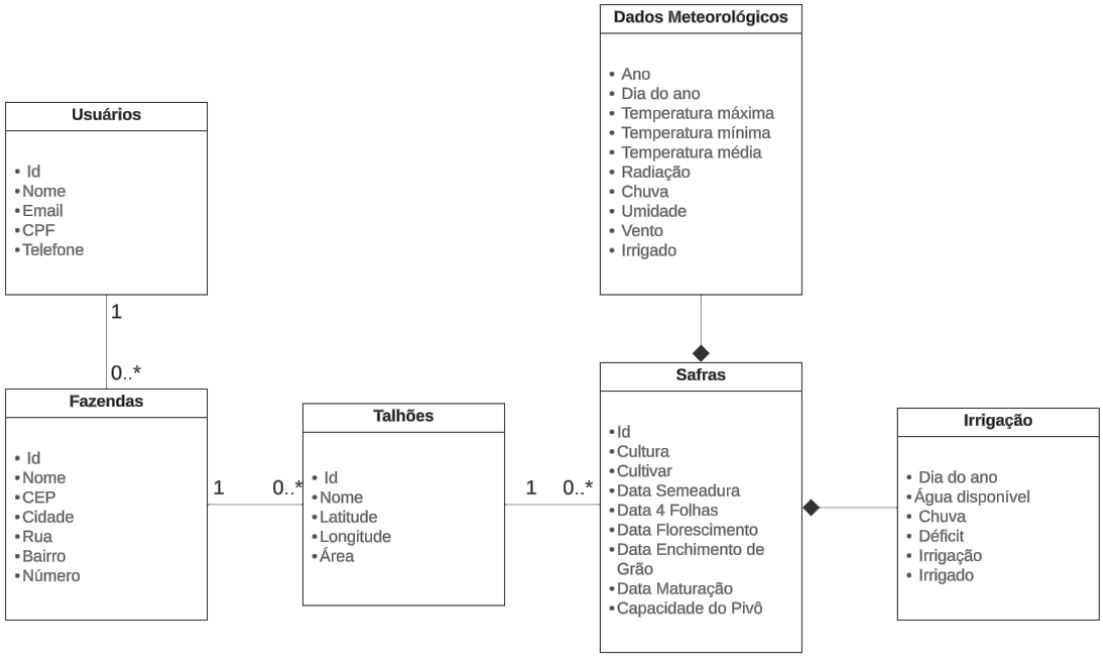
- Autenticação do usuário - Login e Logout
- Página de termos de uso do aplicativo
- Página de perfil do cliente
- Página de informações sobre a empresa
- Listagem de fazendas do usuário
- Listagem de talhões de cada fazenda
- Listagem de safras de cada talhão
- Listagem de irrigações para cada safra
- Atualização de informações da irrigação

Além dos requisitos, para ajudar na estruturação do projeto, foi montado o diagrama de classes, especificado na Figura 9.

Descrição das classes:

- Usuários: classe destinada a guardar as informações pessoais dos clientes.
- Fazendas: classe destinada a guardar as informações das fazendas pertencentes aos clientes.
- Talhões: classe destinada a guardar as informações dos talhões de cada fazenda.
- Safras: classe destinada a guardar as informações das safras de cada talhão. Possui relação de composição com as classes de Dados Meteorológicos e Irrigação.
- Dados Meteorológicos: classe destinada a guardar as informações de dados meteorológicos associados a uma safra.

Figura 9 – Diagrama de Classes



Fonte: Próprio Autor

- Irrigações: classe destinada a guardar as informações de irrigações associadas a uma safra.

3.6 Ferramentas utilizadas

Nessa seção, são apresentadas algumas ferramentas utilizadas durante o processo de desenvolvimento do aplicativo. São ferramentas utilizadas para gerenciar o projeto, fazer o controle do versionamento do código e também desenvolvimento do mesmo.

Para fazer o gerenciamento do projeto como um todo, visando um controle de cada etapa, separação de tarefas, foi utilizada a ferramenta Projects, do GitHub. Com ela é possível definir a visualização Kanban de acordo com o status de cada tarefa, como pendente, em andamento, em teste e finalizada. Além disso, é possível atribuir outras propriedades às tarefas, como dificuldade, tempo, etc.

Para o controle e versionamento do código, foi utilizado o Git, por ser um sistema de controle de versão muito conhecido e usado, além de ser usado pelo GitHub, uma plataforma também de gerenciamento de código e muito usada que, principalmente, fornece um ambiente de colaboração entre desenvolvedores (SILVEIRA, 2023). A partir disso, foi criado um repositório no GitHub para o aplicativo dentro do domínio da empresa CropsTeam, pois ela já utilizava o mesmo para gerenciar e manter seus códigos de outros produtos. Para manter uma estrutura e separar cada etapa do processo, foi criada uma branch[12] de desenvolvimento, a partir da branch principal. Dessa forma, o desenvolvimento de cada feature é feito separadamente, sem afetar o código que já está na branch principal.

[12] https://git-scm.com/book/pt-br/v2/Branches-no-Git-Branches-em-poucas-palavras

Outro ponto importante utilizado para controle e versionamento do código são os commits, que reúnem uma série de alterações realizadas no código e montam um pacote com as mesmas. Com essa estrutura, é possível criar diversas versões do código e acessá-las quando e caso seja necessário (SILVEIRA, 2023). Para manter maior clareza a cada versão, os commits foram feitos após o desenvolvimento de cada funcionalidade ou também correção, por mais simples que fosse, sempre especificando na mensagem tudo o que foi alterado.

Como ferramenta para o desenvolvimento do código, foi utilizada a IDE Visual Studio Code, pois já havia sido utilizada anteriormente e possui diversas funcionalidades que facilitam o desenvolvimento. Uma das funcionalidades são as extensões, que permitem facilitar o desenvolvimento. São extensões voltadas ao GitHub, facilitando tanto o acesso às branches quanto o controle de commits, além de extensões ligadas ao próprio Flutter, melhorando a visualização e criação do código.

Por ser um aplicativo mobile, a empresa CropsTeam disponibilizou um celular com a plataforma Android para execução do mesmo na etapa de desenvolvimento, unido ao computador do autor, com o sistema operacional Windows 10.

4 DESENVOLVIMENTO

Por se tratar de um trabalho diretamente ligado à empresa CropsTeam, no qual tanto as necessidades do projeto, quanto os resultados são de total controle e interesse da mesma, a primeira etapa tratou de conhecer o que era o projeto WaterCrop, quais suas funcionalidades e, principalmente, como o aplicativo poderia acrescentar no produto final. Dessa forma, inicialmente, foram realizadas reuniões entre alguns membros da equipe CropsTeam, juntamente com a professora orientadora e o autor para apresentação básica do projeto, além de reuniões mais técnicas na própria sede da empresa, com mais membros, incluindo tanto membros relacionados à área de Agronomia quanto de Computação.

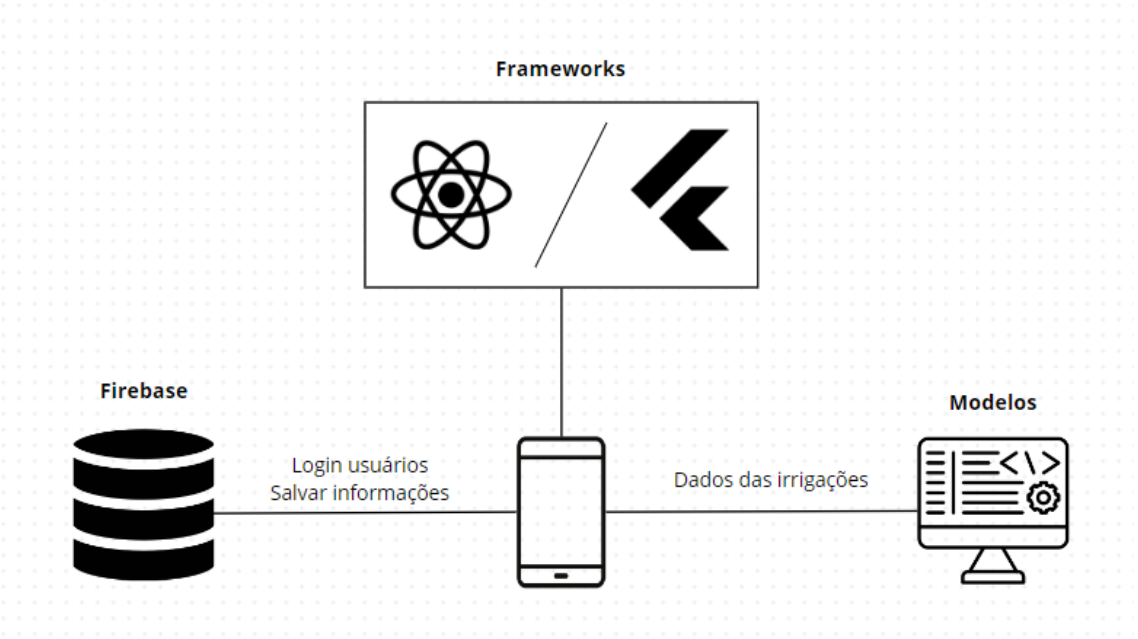
Para o controle de autenticação e gerenciamento de usuários, bem como armazenamento de informações referentes às irrigações, a empresa optou por utilizar o Firebase, uma plataforma criada pelo Google que oferece uma estrutura robusta direcionada a aplicativos, fácil de usar e com diversos serviços disponíveis, o que facilita e acelera o desenvolvimento (RIBEIRO, 2023).

Além da comunicação com o Firebase para autenticação e manutenção dos dados, era necessário que o aplicativo se comunicasse com os modelos desenvolvidos em Fortran, para obtenção dos dados referentes à irrigação de cada safra.

Estabeleceu-se, no início do projeto, que a escolha do framework para criação do aplicativo se daria por meio de testes de funcionalidades, utilizando React Native e Flutter. Como já mencionado anteriormente, são plataformas que possuem muitas funcionalidades e estão sendo muito usadas na atualidade.

Portanto, as reuniões iniciais realizadas, além da finalidade de entendimento do projeto WaterCrop, tiveram como resultado um esboço inicial sobre o aplicativo, já incluindo as possíveis tecnologias a serem usadas, de acordo com a Figura 10.

Figura 10 – Esboço inicial da estrutura



Fonte: Próprio Autor

4.1 Protótipos exploratórios em Flutter e React Native

Após o estabelecimento de uma estrutura básica para o aplicativo, com o Firebase como banco de dados para autenticação de usuários e armazenamento de informações referentes às irrigações, o próximo passo no desenvolvimento foi criar aplicativos simples de teste usando os frameworks React Native e Flutter, pois, como já foi mencionado anteriormente, são, atualmente, os mais conhecidos e usados.

Para os testes, foi montado um conjunto de tarefas a serem realizadas, de acordo com funcionalidades que seriam necessárias no aplicativo final. Conforme já mencionado, para auxiliar no controle do desenvolvimento foi utilizada a ferramenta Projects, do GitHub. Nela foram criadas tarefas e, em cada uma delas, foi especificado o trabalho a ser feito. A lista de requisitos e funcionalidades testadas foi igual para os dois frameworks, pois o intuito era testar tanto a funcionalidade quanto o esforço necessário para implementar cada etapa. Os requisitos implementados foram a autenticação de usuários com o Firebase, adição de botões, formatação de textos, navegação entre páginas, listar informações, integração com sensores do smartphone, como giroscópio, animações de linhas e curvas e conexão com o Google Maps.

Como eram apenas aplicativos simples de teste, não houve tanta com design de interface, pois o foco principal eram os testes das funcionalidades.

Ao final do desenvolvimento dos dois protótipos, levou-se em consideração a capacidade dos dois frameworks de atender aos requisitos e funcionalidades impostas e, principalmente, o esforço para realizar tais tarefas, bem como a dificuldade de adaptação do desenvolvedor com cada plataforma. Em relação aos requisitos, tanto o React Native quanto o Flutter apresentam boas soluções para cada item, tendo o React uma maior flexibilidade, por ser uma plataforma que está há mais tempo no mercado. Em relação a

esforço e adaptação, pelo fato da linguagem Dart se assemelhar com a linguagem Java, que já era de conhecimento do desenvolvedor deste trabalho, a adaptação ao framework Flutter ocorreu mais facilmente, sendo o ponto mais importante na decisão. Dessa forma, por atender aos requisitos necessários e uma melhor adaptação à plataforma, optou-se pelo Flutter como framework para desenvolvimento do aplicativo do WaterCrop.

4.2 Implementação do aplicativo

A implementação começou a partir da prototipagem das telas, realizada pela própria equipe da CropsTeam no Figma, uma plataforma para construção e design de interfaces.

Uma das principais etapas do desenvolvimento foi a integração com o Firebase para autenticação de usuários. Como já havia sido desenvolvido um protótipo exploratório em Flutter, um pouco da estrutura do mesmo pôde ser reutilizada, principalmente a parte de autenticação do usuário no Firebase. Durante esse processo, foi necessário o uso de um pacote de gerenciamento de estados.

Em definição básica, um gerenciamento de estados tem a função de observar o estado de um objeto e notificar os demais objetos e componentes que ocorreu alguma mudança, para que eles atualizem seus valores (LIMA, 2021). Recomendado pela própria documentação[13] do Flutter, um dos mais simples, fáceis, porém que supre as necessidades do aplicativo, é o pacote Provider. Com ele é possível notificar a partir da classe controladora de autenticação, se houve alguma mudança com o usuário, para todos os componentes que dependem do usuário autenticado.

Ainda sobre a autenticação de usuários, como o Firebase mostra em sua própria documentação, existem várias formas para autenticação, podendo ser realizada por email, número de celular, ou até mesmo pela conta do Google ou redes sociais. A forma escolhida pela empresa foi via email e senha. A Figura 11 mostra a página de login, utilizando email e senha.

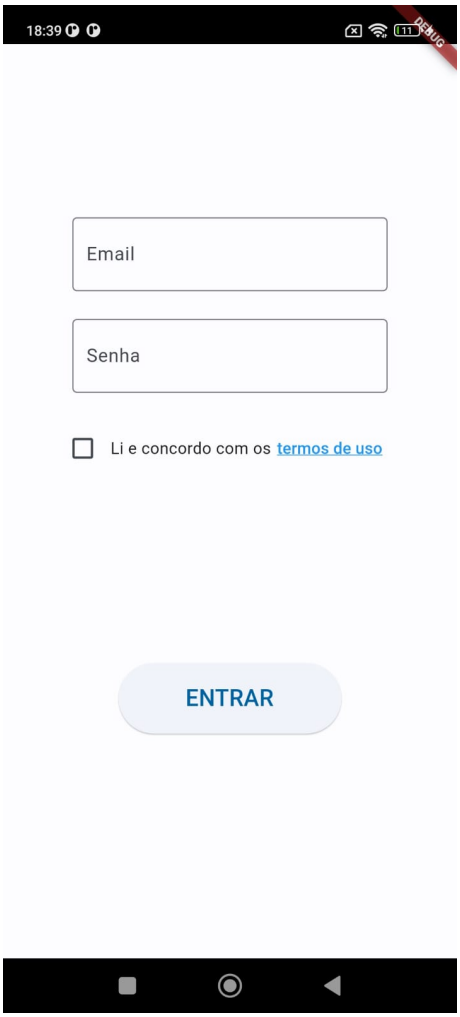
Após realizar o login, o usuário é direcionado para a página de fazendas, conforme a Figura 12. Nesta página, ele pode visualizar as informações de cada fazenda, tocando no ícone de informação, de acordo com a Figura 12b.

Para essa visualização, que exige leitura de dados, era necessário que o aplicativo atualizasse suas informações em tempo real, de acordo com qualquer mudança no Firebase. Dessa forma, foi utilizada a widget StreamBuilder[14], que se constrói a cada nova atualização de dados da Stream[15] a qual ela se refere. Stream é uma estrutura do Flutter que fornece uma sequência assíncrona de dados, incluindo eventos, que podem ser gerados pelo usuário, ou como nesse caso, atualização no banco de dados. Dessa forma, qualquer alteração nos dados é refletida automaticamente no aplicativo. A widget StreamBuilder é utilizada tanto na página de fazendas, quanto talhões, safras e também irrigações.

Ao clicar em uma fazenda, a página de talhões é exibida, listando os talhões referentes àquela fazenda. Para cada talhão, também é possível visualizar suas informações.

[13] https://docs.flutter.dev/data-and-backend/state-mgmt/simple
[14] https://firebase.flutter.dev/docs/firestore/usage/
[15] https://api.flutter.dev/flutter/dart-async/Stream-class.html

Figura 11 – Página de Login



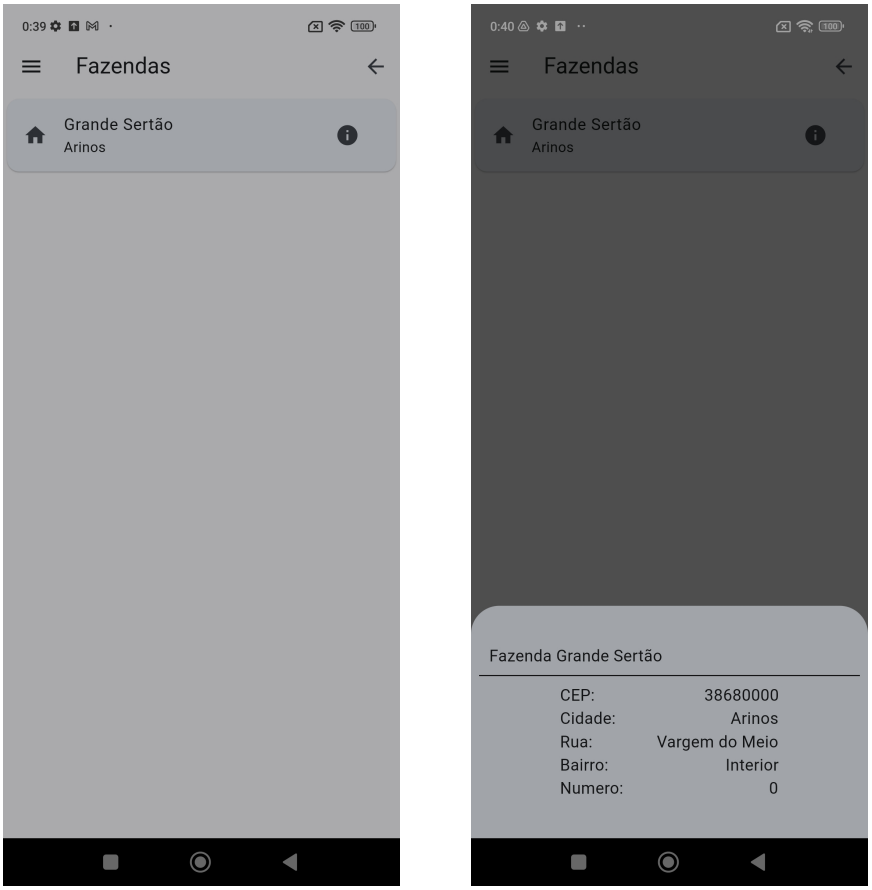
Fonte: Próprio Autor

Clicando sobre eles, o usuário é direcionado à página de safras daquele talhão. Para as safras, o processo é mesmo, sendo que, ao clicar em uma safra, o usuário é direcionado à página de irrigações daquela safra, de acordo com a Figura 13.

Na página de irrigações, é mostrada a safra atual e sua cultivar. Na lista de irrigações, aparecem todas para as quais foram informados os dados meteorológicos. Dessa forma, pode ser exibida uma lista com todas as irrigações sugeridas e realizadas desde o início da safra até a data que possuir dados meteorológicos cadastrados. O dia atual fica destacado para facilitar a visualização do produtor. Em relação à confirmação de uma irrigação sugerida, é exibida uma checkbox para que o produtor marque caso tenha realmente realizado essa irrigação. Ao marcar a checkbox, uma confirmação é solicitada ao usuário, de acordo com a Figura 14a.

A confirmação de uma irrigação é necessária pois os modelos em Fortran também consideram a quantidade irrigada ao efetuar os cálculos das irrigações futuras. Ele considera tanto as irrigações já feitas e confirmadas de dias anteriores, como as futuras irrigações sugeridas pelo próprio modelo, tratando-as como se já tivessem sido feitas.

Figura 12 – Página de fazendas



(a) Lista de fazendas (b) Informações da fazenda

Fonte: Próprio Autor

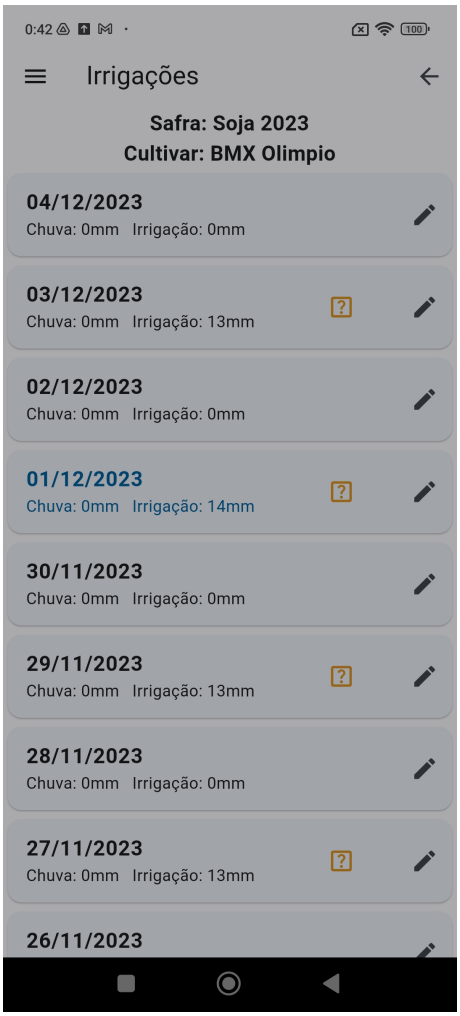
Portanto, a confirmação é necessária pois, se ela não for confirmada quando realizada, o modelo não considerará o valor para os cálculos futuros.

Além disso, como as previsões meteorológicas podem ser imprecisas em determinados momentos, para cada item da lista de irrigação, é possível alterar a quantidade de chuva, caso tenha chovido mais ou menos no talhão, como mostra a Figura 14b. Para salvar essas informações no Firebase, foram utilizadas as estruturas de coleções e documentos, onde cada coleção representa uma entidade, como fazenda, talhão e safra, as quais possuem documentos, que são as unidades de armazenamento de dados do Firebase.

Nessa etapa do desenvolvimento, para obter as sugestões de irrigações, foi necessário realizar a conexão com a API dos modelos de irrigação. A conexão foi realizada por meio de requisições HTTPS utilizando o método POST, informando os parâmetros da safra, como datas de semeadura e de cada estágio da planta, dentre outras informações técnicas, além dos dados meteorológicos, no formato JSON. Os resultados, retornados também no formato JSON, foram convertidos e armazenados dentro da safra em questão.

Para realizar a chamada da API, bem como tratamento e armazenamento dos resultados, foi utilizado o Firebase Cloud Functions. Com ele, foi possível escrever funções em TypeScript que realizam todo o processo, acionadas quando uma alteração no campo

Figura 13 – Página de Irrigações



Fonte: Próprio Autor

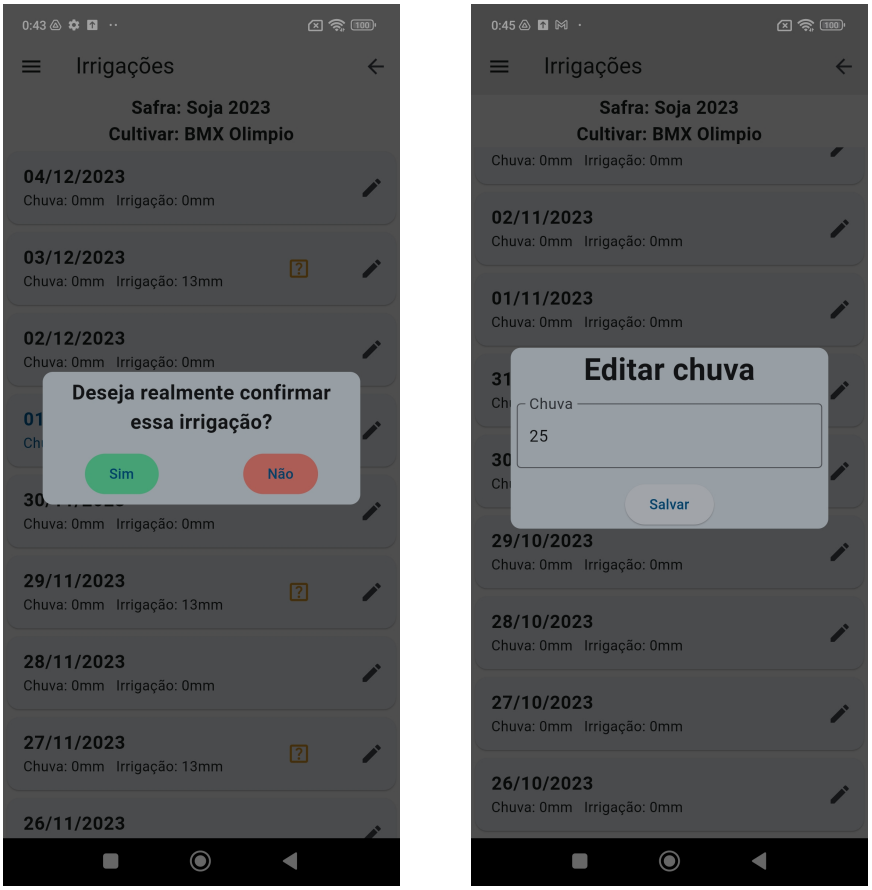
de irrigações, dentro da safra, ocorrer. Dessa forma, quando a irrigação sugerida para um determinado dia for confirmada, ou quando a quantidade de chuva for alterada, uma nova chamada na API é feita e os novos dados são armazenados no Firebase. Utilizando essa estrutura, é possível que o aplicativo no lado do cliente apenas leia e salve informações no Firebase, sem que seja necessária a conexão com outras estruturas.

Até o presente momento, para fins de teste, os dados dos produtores estão sendo inseridos diretamente no Firebase, através de telas acessíveis apenas pelo autor do projeto, como forma de automatizar o processo. Em uma versão futura, será implementada a interface do aplicativo para o administrador e para os consultores da empresa. Dessa forma, será possível gerenciar os clientes e suas informações diretamente pelo aplicativo.

5 TESTES COM USUÁRIOS

Por se tratar de um aplicativo voltado diretamente ao produtor rural, foi aplicado um teste de usabilidade qualitativo (MORAN, 2019) com produtores rurais e consultores da empresa, visando obter um feedback em relação à usabilidade do mesmo. A criação

Figura 14 – Edição de Irrigações



(a) Confirmação de Irrigação (b) Edição de Chuva

Fonte: Próprio Autor

do teste se baseou no método SUS (BROOKE, 1996), porém não seguiu à risca todo o processo, visando torná-lo mais simples para os testadores, que não fazem parte do meio acadêmico.

A adaptação ao método consistiu em criar perguntas relacionadas às tarefas executadas pelo usuário durante o teste, para obter um feedback sobre cada parte do aplicativo. Para responder a cada pergunta, o usuário marca sua resposta em uma escala de 1 a 5, onde 1 corresponde a Discordo Totalmente e 5 corresponde a Concordo Totalmente.

Após a realização dos testes, as respostas coletadas serviram para analisar separadamente qual parte do aplicativo possui boa interface e usabilidade, e qual parte necessita de correções ou adaptações posteriores.

5.1 Aplicação dos testes

Inicialmente, o objetivo era realizar os testes junto com os produtores clientes da empresa CropsTeam, assinantes do projeto WaterCrop, por serem o público alvo real, no qual o aplicativo é destinado. Como a maioria dos clientes da empresa se localiza na região centro-oeste do país, a aplicação precisaria ser realizada via chamada online. No entanto, alguns pontos inviabilizaram a aplicação do teste para esses usuários. Primeiramente, o

desenvolvimento do aplicativo se estendeu além do inicialmente previsto, de forma que a etapa de testes coincidiu com o período de plantio da safra de soja no Brasil. Assim, o tempo disponível tanto dos produtores, quanto dos consultores da empresa CropsTeam tornou-se muito escasso e não foi possível encontrar uma agenda disponível para ambas as partes.

Dessa forma, optou-se por realizar os testes com consultores da equipe, que estão familiarizados com o processo de irrigação e também com alguns produtores rurais da região. Ao todo, foram realizados testes com 5 usuários. Mesmo sendo um pequeno número, esta quantidade de testadores pode gerar percepções relevantes sobre artefato em teste (NIELSEN, 2012). Os testes realizados com os consultores foram úteis para obter uma avaliação crítica sobre as irrigações propostas pelo aplicativo. Por outro lado, os testes realizados com os produtores da região retornaram um feedback do ponto de vista do usuário final em relação à interface e usabilidade.

Para os testes, foi disponibilizado um APK, arquivo contendo todos os dados do aplicativo, para ser instalado no celular dos usuários. Pela pouca disponibilidade de tempo, foi possível apenas testar o aplicativo em sua versão para Android. Após ter instalado o aplicativo, o usuário era orientado a realizar uma série de tarefas:

- Login
- Navegar pelas fazendas e visualizar suas informações
- Navegar pelos talhões e visualizar suas informações
- Navegar pelas safras e visualizar suas informações
- Navegar pelas irrigações
- Confirmar uma irrigação sugerida
- Editar a quantidade de chuva de uma irrigação

Depois de realizar as tarefas, os usuários responderam um formulário disponibilizado. Além das perguntas sobre a usabilidade, onde era possível marcar respostas de 1 a 5, foram disponibilizadas perguntas discursivas para receber possíveis pontos de melhoria no aplicativo, bem como entender como foi a experiência do usuário com o mesmo.

Na Tabela 2, encontram-se os resultados dos testes com 5 usuários, sendo 2 deles consultores da empresa e 3 deles produtores rurais da região.

5.2 Resultados obtidos

Diante das respostas obtidas, foi possível notar que, para a maioria dos testadores, a interface foi percebida como sendo clara e organizada. A navegação pelos menus foi simples e intuitiva, sendo que a maioria dos testadores conseguiu navegar facilmente entre fazendas, talhões e safras, bem como visualizar suas informações.

Para as informações sobre irrigações, tanto na visualização como na edição, a maioria teve uma boa experiência, mas obtivemos respostas onde um usuário relatou ter

Perguntas	1	2	3	4	5
A interface da aplicação é clara e organizada.			1	1	3
As funcionalidades relacionadas à navegação pelo aplicativo pelos menus são simples e intuitivas.				2	3
Consegui navegar facilmente pelas fazendas, talhões e safras.				1	4
Consegui visualizar facilmente as informações de fazendas, talhões e safras.			1	1	3
As funcionalidades relacionadas a visualização das irrigações são simples e intuitivas.		1			4
As funcionalidades relacionadas a confirmação de uma irrigação são simples e intuitivas.	1			2	2
As funcionalidades relacionadas a alteração da quantidade das chuvas para cada irrigação são simples e intuitivas.	1				4

Tabela 2 – Perguntas do formulário

dificuldade para realizar a confirmação de irrigação e alteração da quantidade de chuvas. O mesmo sugeriu que a interface poderia ser mais clara para edição das chuvas e confirmação das irrigações.

6 CONCLUSÃO

O presente trabalho teve como objetivo desenvolver um aplicativo para execução do projeto WaterCrop para a empresa CropsTeam. O aplicativo tem como finalidade principal facilitar o acesso das informações referentes à irrigação aos produtores clientes da empresa. Com o aplicativo, é possível que o usuário visualize suas fazendas como um todo, podendo ter seus talhões, safras e, principalmente, uma lista de irrigações para cada dia da safra. Por ser um aplicativo destinado ao produtor rural, o intuito foi deixá-lo o mais simples possível, necessitando cadastrar ou editar um mínimo de informações.

Além do desenvolvimento, uma etapa muito importante foi o estudo dos frameworks React Native e Flutter, analisando seus prós e contras a fim de encontrar o que melhor atendia aos requisitos necessários para o aplicativo WaterCrop. Vale lembrar que algumas funcionalidades testadas nos protótipos, como integração com o Google Maps e sensores do smartphone e animações visuais não foram empregadas na versão atual do aplicativo. Durante o decorrer do tempo e do desenvolvimento, notou-se que não haveria tempo hábil para realização de tais tarefas. As mesmas ficam como trabalho futuro, visando agregar mais funcionalidades. Além de acrescentar funcionalidades, correções poderão ser feitas na área de listagem de irrigações, conforme foi apontado na seção de testes com usuários.

A partir da aplicação dos testes com usuários, foi possível obter um feedback em relação à usabilidade e interface do aplicativo. Dessa forma, melhorias na página de listagem de irrigações podem ser feitas futuramente. Uma delas seria tornar cada item da lista clicável. Ao clicar, abrir uma nova página que mostre mais informações

referentes à irrigação daquele dia, além dos botões para edição de chuva e confirmação da irrigação. Para esses botões, adicionar o texto descritivo em cada um, de acordo com a ação executada, tornando mais fácil de visualizar o que cada um faz.

Um APK do aplicativo WaterCrop, juntamente com um vídeo de demonstração, está disponível em uma pasta do drive, podendo ser acessada a partir do link <https://drive.google.com/drive/folders/1Y1FfP5ii_nNhC5wel4BVNArlbEziffjZ?usp=drive_link>. Para realizar o login no aplicativo, é necessário solicitar as credenciais ao autor, via email, para o endereço gpretto@inf.ufsm.br. O código fonte do aplicativo está disponibilizado em um repositório no GitHub, podendo ser acessado pelo link <https://github.com/GuilhermePretto/TCC-WaterCrop>.

Levando em conta todos os pontos citados acima, pode-se afirmar que o aplicativo do WaterCrop é um aplicativo simples, focado em ter uma interface objetiva e de fácil utilização, voltado para produtores rurais. Pode ser aprimorado facilmente, agregando novas funcionalidades, além de ajustes, visando torná-lo cada vez mais importante para o produtor rural no processo de irrigação.

REFERÊNCIAS

ALBUQUERQUE, P. E. P. de; FARIA, C. M. de; COELHO, E. A. Utilização do software irrigafácil para manejo de irrigação. *Embrapa Milho e Sorgo.*, p. 36, mar. 2011. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/56166/1/doc-128.pdf>. Citado na página 19.

BORTOLUZZI, M. P. Risco de ocorrência de excesso e déficit hídrico na soja em terras baixas. In: . [s.n.], 2019. Disponível em: <https://repositorio.ufsm.br/handle/1/20743>. Citado na página 8.

BROOKE, J. Sus – a quick and dirty usability scale. In: ____. [S.l.: s.n.], 1996. p. 189–194. Citado na página 30.

CAPPELLI, E. *Desenvolvimento Híbrido com Flutter: Prós e Contras*. 2018. Disponível em: <https://medium.com/@devmob/desenvolvimento-h%C3%ADbrido-com-flutter-pr%C3%B3s-e-contras-6f3f422c480c>. Citado na página 15.

CHARLAND, A.; LEROUX, B. Mobile application development: Web vs. native. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 5, p. 49–53, may 2011. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1941487.1941504>. Citado na página 13.

CONSULTING, M. *Flutter vs React Native, afinal, qual o melhor em 2022?* 2022. Disponível em: <https://mindconsulting.com.br/2022/08/flutter-vs-react-native-afinal-qual-o-melhor-em-2022/#:~:text=O%20Flutter%20assumiu%20que%20voc%C3%AA,execu%C3%A7%C3%A3o%20em%20qualquer%20dispositivo%20m%C3%B3vel.> Citado na página 13.

CORREA, F.; PINTO, G. Desafios no desenvolvimento de aplicaÇÕes para dispositivos mÓveis e os frameworks multiplataformas. *Revista Interface Tecnológica*, v. 17, p. 91–102, 12 2020. Citado na página 12.

CUBOS ACADEMY. *Flutter: por que aprender o framework da Google é uma boa ideia em um mercado mobile crescente*. 2021. Acesso em: 23 set. 2023. Disponível em: <https://blog.geekhunter.com.br/flutter/#A_procura_por_desenvolvedores_mobile_esta_aumentando>. Citado na página 11.

FILHO, J. E. R. O. V.; GASQUES, J. G. O. Uma jornada pelos contrastes do brasil: Cem anos do censo agropecuário. Instituto de Pesquisa Econômica Aplicada (Ipea), 2020. Citado na página 8.

FLUTTER. *Flutter architectural overview*. 2023. Disponível em: <https://docs.flutter.dev/resources/architectural-overview#architectural-layers>. Citado na página 15.

FOUNDATION, M. *Sobre JavaScript | MDN*. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript>. Citado na página 14.

FREITAS, B. C. d. Flutter e react native: uma análise comparativa entre dois frameworks de desenvolvimento mobile multiplataforma. Universidade Federal do Rio de Janeiro, 2022. Citado na página 14.

KAPOS, G. *Quais são os diferentes tipos de aplicativos e como eles funcionam?* 2021. Disponível em: <https://www.segs.com.br/info-ti/326122-quais-sao-os-diferentes-tipos-de-aplicativos-e-como-eles-funcionam>. Citado na página 12.

KHAN, S. *Flutter vs React Native: Which is the Best Choice for 2022?* 2021. Disponível em: <https://javascript.plainenglish.io/flutter-vs-react-native-which-is-the-best-choice-for-2021-e695e79c6707>. Citado na página 14.

LAMAS, F. M. A tecnologia na agricultura. 2017. Disponível em: <https://www.embrapa.br/en/busca-de-noticias/-/noticia/30015917/artigo-a-tecnologia-na-agricultura#:~:text=Segundo%20dados%20gerados%20a%20partir,aumento%20da%20produ%C3%A7%C3%A3o%20de%20gr%C3%A3os>. Citado na página 8.

LIMA, R. *O que é um gerenciador de estados?* 2021. Disponível em: <https://www.alura.com.br/artigos/como-gerenciar-estados-com-flutter-provider>. Citado na página 26.

MEIRELLES, F. *Pesquisa do Uso da TI - Tecnologia de Informação nas Empresas*. 2023. Disponível em: <https://eaesp.fgv.br/producao-intelectual/pesquisa-anual-uso-ti>. Citado na página 11.

MORAN, K. *Usability Testing 101*. 2019. Disponível em: <https://www.nngroup.com/articles/usability-testing-101/>. Citado na página 29.

NIELSEN, J. *How Many Test Users in a Usability Study?* 2012. Acesso em: 3 dez. 2023. Disponível em: <https://www.nngroup.com/articles/how-many-test-users/>. Citado na página 31.

OLIVEIRA, L. V. de; OLIVEIRA, F. G.; FIGUEIREDO, F. P. Aplicativo multiplataforma para dimensionamento de irrigação por pivô central. *IRRIGA*, v. 1, n. 1, p. 40–47, jun. 2018. Disponível em: <https://irriga.fca.unesp.br/index.php/irriga/article/view/2812>. Citado na página 18.

RIBEIRO, A. L. S. *O que é Firebase? Para que serve, principais característica e um Guia dessa ferramenta Google*. 2023. Disponível em: <https://www.alura.com.br/artigos/firebase>. Citado na página 24.

SANTOS, A. R. et al. Combining challenge-based learning and scrum framework for mobile application development. In: . New York, NY, USA: Association for Computing Machinery, 2015. (ITiCSE '15), p. 189–194. ISBN 9781450334402. Disponível em: <https://doi.org/10.1145/2729094.2742602>. Citado na página 11.

SARTORI, G. M. S. et al. Rendimento de grãos de soja em função de sistemas de plantio e irrigação por superfície em planossolos. *Pesquisa Agropecuária Brasileira*,

v. 50, n. 1, p. 1139–1149, 2015. Disponível em: <https://www.scielo.br/j/pab/a/Gh9CDRrp5qZgJ4MvBNLp67G/?lang=pt>. Citado na página 8.

SILVA, G. *O que é Dart?* 2023. Disponível em: <https://coodesh.com/blog/dicionario/o-que-e-dart/>. Citado na página 14.

SILVEIRA, P. *Por que usar Git e GitHub?* 2023. Disponível em: <https://www.alura.com.br/artigos/o-que-e-git-github>. Citado 2 vezes nas páginas 23 e 24.

STARTAGRO. *Perfis STARTAGRO: Conheça a startup IZagro*. 2018. Disponível em: <https://www.startagro.agr.br/perfis-startagro-conheca-izagro/>. Citado na página 18.

TANAKA, C. C. G. e S. Comparação entre o desempenho de aplicações para smartphones desenvolvidas em flutter e react native: uma análise utilizando algoritmos de ordenação. *Revista Terra & Cultura: Cadernos de Ensino e Pesquisa*, v. 39, n. especial, p. 7–17, 2023. ISSN 2596-2809. Disponível em: <http://periodicos.unifil.br/index.php/Revistateste/article/view/2796>. Citado na página 14.

TESTEZLAF, R. *Irrigação: métodos, sistemas e aplicações*. [S.l.: s.n.], 2017. ISBN 978-85-99678-10-7. Citado na página 10.