INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO SUL
CAMPUS RESTINGA

UM APLICATIVO PARA REGISTRO DE DIÁRIOS DE ENXAQUECA UTILIZANDO FLUTTER E OS PRINCÍPIOS DA ARQUITETURA LIMPA

RAFAEL SOUZA PINTO

Porto Alegre
2021

RAFAEL SOUZA PINTO

UM APLICATIVO PARA REGISTRO DE DIÁRIOS DE ENXAQUECA UTILIZANDO FLUTTER E OS PRINCÍPIOS DA ARQUITETURA LIMPA

Trabalho de Conclusão de Curso apresentado, junto ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal Educação, Ciência e Tecnologia do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Jean Carlo Hamerski

Porto Alegre
2021

RAFAEL SOUZA PINTO

UM APLICATIVO PARA REGISTRO DE DIÁRIOS DE ENXAQUECA UTILIZANDO FLUTTER E OS PRINCÍPIOS DA ARQUITETURA LIMPA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Jean Carlo Hamerski

Aprovado em novembro de 2021.

Documento assinado digitalmente
gov.br
Jean Carlo Hamerski
Data: 25/11/2021 15:35:12-0300
Verifique em https://verificador.iti.br

Jean Carlo Hamerski

Documento assinado digitalmente
gov.br
Gleison Samuel do Nacimento
Data: 02/12/2021 19:02:04-0300
Verifique em https://verificador.iti.br

Gleison Samuel Do Nascimento – IFRS

Documento assinado digitalmente
gov.br
DIEGO MOREIRA DA ROSA
Data: 03/12/2021 08:51:13-0300
Verifique em https://verificador.iti.br

Diego Moreira da Rosa – IFRS

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL
Reitor: Prof. Júlio Xandro Heck
Pró-Reitora de Ensino: Prof. Lucas Coradini
Diretor-geral do *Campus* Restinga: Prof. Rudinei Müller
Coordenador do CST em Análise e Desenvolvimento de Sistemas: Prof. Iuri Albandes Cunha Gomes
Bibliotecária-chefe do *Campus* Restinga: Paula Porto Pedone

Para Tamara

AGRADECIMENTOS

Agradeço ao Instituto Federal do Rio Grande do Sul e ao Campus Restinga por me possibilitarem realizar meus estudos em uma instituição pública, gratuita e de qualidade, onde pude aprender com excelentes professores.

Agradeço aos professores Gleison e Diego pela leitura do trabalho e pelas contribuições.

Agradeço também ao Jean, pelas discussões construtivas e por ter me apresentado a novas tecnologias.

Por todo o resto, agradeço à Tamara.

RESUMO

A enxaqueca é uma doença neurológica crônica que tem como principal sintoma dores de cabeça severas e recorrentes. Em pacientes que fazem tratamento e acompanhamento contínuo da enxaqueca, é recomendada a utilização de diários cuja manutenção pode ajudar pacientes e médicos a fazer um correto diagnóstico ou avaliar o funcionamento do tratamento. Nos últimos tempos, com o crescente número de soluções para rastreamento de aspectos de saúde em dispositivos móveis, tem havido também espaço para esse tipo de diário em formato digital, que pode trazer vantagens para o paciente. O presente trabalho de conclusão de curso tem como objetivo projetar e desenvolver uma solução tecnológica, em dispositivos Android, para o controle e rastreamento de crises de enxaqueca. Para a implementação, foi utilizado o framework Flutter, aliado aos princípios de design da Arquitetura Limpa (*Clean Architecture*). Como resultado final, foi produzido um aplicativo com o qual o usuário é capaz de registrar os dias e horários que teve crises de enxaqueca, com suas intensidades e medicamentos administrados, para visualizá-los em um calendário, por meio de cores. O uso de Flutter com princípios da Arquitetura Limpa trouxeram agilidade para a implementação e manutenção do código.

Palavras-chave: Enxaqueca, aplicativo de saúde, Flutter, Arquitetura Limpa

ABSTRACT

Migraine is a chronic neurological disorder characterized by severe and recurrent episodes of headache. Some migraine patients are required to fill in migraine diaries that can help them and their physicians to make an adequate diagnosis or to evaluate the effectiveness of treatment. In recent times, there has been a growing number of mobile solutions designed to track several health aspects, including digital versions of those migraine diaries. This work aims to design and develop a diary app, for Android devices, in order to help users track migraine episodes. For the implementation, we used the Flutter framework along with the principles of Clean Architecture. The final product was an app with which the user is able to register the days and times of a migraine episode, its intensity and medication taken, in order to view them in a calendar, shown by color. The use of Flutter and the principles of Clean Architecture brought agility to the development and maintenance of the code.

Keywords: Migraine, health app, Flutter, Clean Architecture

LISTA DE TABELAS

TABELA 2.1 - COMPARATIVO ENTRE AS TECNOLOGIAS ANALISADAS 21

TABELA 4.1 - VARIÁVEIS MAIS RELEVANTES PARA RASTREIO DE CRISES DE ENXAQUECA, CONFORME LEVANTAMENTO DE HUNDERT ET AL. (2014) 28

TABELA 4.2 - COMPARAÇÃO ENTRE AS VARIÁVEIS DE ENXAQUECA USADAS PELOS DIÁRIOS RECOMENDADOS 29

TABELA 4.3. HISTÓRIAS DE USUÁRIO DO SISTEMA DE DIÁRIO DE ENXAQUECA 31

LISTA DE FIGURAS

FIGURA 2.1 – TELA INICIAL DO *MIGRAINE BUDDY* 18

FIGURA 2.2 – TELA DE CALENDÁRIO DO *MIGRAINE BUDDY* 18

FIGURA 2.3 – TELA DE INICIAL DO *DIÁRIO DA CEFALEIA* 19

FIGURA 2.4 – TELA INICIAL DO *MIGRAINE MONITOR,* COM CRONÔMETRO 20

FIGURA 3.1 - REPRESENTAÇÃO DO FUNCIONAMENTO DA ARQUITETURA LIMPA, CONFORME MARTIN (2012). 25

FIGURA 4.1 - TELA INICIAL DO SISTEMA. 35

FIGURA 4.2 - TELA DE CADASTRO DE CRISE, COM FLUXO DE SELEÇÃO DE INTENSIDADE E MEDICAMENTO 36

FIGURA 4.3 - CALENDÁRIO MOSTRANDO OS DIAS DE CRISE. 36

FIGURA 4.4 - CALENDÁRIO MOSTRANDO DETALHES DA CRISE 37

FIGURA 4.5 - DIAGRAMA DE ENTIDADES-RELACIONAMENTO. OS TIPOS DE DADOS SÃO: PK – PRIMARY KEY (CHAVE PRIMÁRIA), ARR – ARRAY, NUM – NUMBER, OBJ –OBJECT, STR – STRING, TS – TIMESTAMP 38

FIGURA 4.6 - DIAGRAMA DE CLASSES PARA A SOLUÇÃO CONCEITUAL PROPOSTA 39

FIGURA 5.1 - ESTRUTURA DE DIRETÓRIOS DO PROJETO 42

FIGURA 5.2 - DIAGRAMA DE SEQUÊNCIA DO CADASTRO DE NOVA CRISE 45

FIGURA 5.3 - TELA INICIAL DO DIÁRIO DE ENXAQUECA 47

FIGURA 5.4 - TELA DE CADASTRO DE CRISE 48

FIGURA 5.5 - MENU PARA SELEÇÃO DE DIA E HORA DE INÍCIO 49

FIGURA 5.6 - MENU PARA A SELEÇÃO DE MEDICAMENTO 50

FIGURA 5.7 - TELA DE CALENDÁRIO COM OS DIAS DE CRISE MARCADOS COM CORES 51

FIGURA 5.8 - CAIXA DE DIÁLOGO COM INFORMAÇÕES DA CRISE 52

FIGURA 5.9 TELA COM A LISTA DE CRISES CADASTRADAS 53

FIGURA 5.10 - TELA DE CADASTRO DE MEDICAMENTO 54

SUMÁRIO

1. INTRODUÇÃO 13
1.1. JUSTIFICATIVA 14
1.2 OBJETIVO GERAL 16
1.3 OBJETIVOS ESPECÍFICOS 16
1.4 ESTRUTURA DO TRABALHO 16
2. TRABALHOS CORRELATOS 17
2.1 MIGRAINE BUDDY 17
2.2 DIÁRIO DA CEFALEIA 19
2.3 MIGRAINE MONITOR 19
2.4 COMPARATIVO 21
3. FUNDAMENTAÇÃO TEÓRICA 22
3.1 TECNOLOGIAS 22
3.1.1. Flutter 22
3.1.2. Firebase e Cloud Firestore 23
3.2. DESIGN PATTERNS 23
3.2.1. Arquitetura Limpa 23
3.2.2. BloC 25
4. SOLUÇÃO CONCEITUAL 27
4.1. LEVANTAMENTO DAS VARIÁVEIS DE ENXAQUECA QUE SERÃO UTILIZADAS PELA SOLUÇÃO PROPOSTA 27
4.2. HISTÓRIAS DE USUÁRIO/REQUISITOS FUNCIONAIS 30
4.3. PROTÓTIPOS DE TELAS 35
4.4. DIAGRAMA DE CLASSES 39
5. IMPLEMENTAÇÃO 40
5.1. ESTRUTURA GERAL DO PROJETO 40
5.2 O BLOC E A GERÊNCIA DE ESTADOS 43
5.3 O BANCO DE DADOS 46
5.4 O APLICATIVO 46
5.4.1 A tela inicial 47
5.4.2 A tela de cadastro de crises 48
5.4.3 O Calendário 50
5.4.4 As listas de crises e medicamentos 52
6. CONCLUSÃO 55
REFERÊNCIAS BIBLIOGRÁFICAS 57

1. INTRODUÇÃO

A enxaqueca é uma doença neurológica crônica que tem como principal sintoma dores de cabeça severas, geralmente em um dos lados, que surgem como crises que podem durar de algumas horas a mais de três dias. Estima-se que por volta de 15% da população mundial sofra da doença, que é a terceira maior causa de diminuição da expectativa de vida ajustada à qualidade de vida, isto é, o número de anos que um indivíduo pode esperar viver sem incapacidades (VO et al., 2012). Não há cura conhecida para essa doença (NHS, 2017). Apesar disso, existem diversas formas de tratamento que podem ajudar a diminuir a frequência dos ataques e amenizar a dor e outros sintomas relacionados. Geralmente, o tratamento se dá por uma combinação da administração de medicamentos profiláticos e mudanças comportamentais (GILMORE e MICHAEL, 2011; OSKOUI et al., 2019).

Em pacientes que fazem tratamento e acompanhamento contínuo da enxaqueca com um neurologista, é recomendada a utilização de diários, geralmente em forma de papel, cuja manutenção pode ajudar pacientes e médicos a, por exemplo, fazer um correto diagnóstico, rastrear gatilhos para a dor ou avaliar o funcionamento do tratamento (STANFORD, 2020; THE NATIONAL HEADACHE FOUNDATION, 2021). Diários também têm sido utilizados em estudos para a investigação de fatores como: i) a caracterização dos sintomas da fase final do ataque de enxaqueca (GIFFIN et al., 2016); ii) de que forma as condições atmosféricas podem influenciar a doença (ZEBENHOLZER et al., 2010); iii) e o comportamento da doença durante a gravidez (KVISVIK et al., 2011).

Nos últimos tempos tem havido uma crescente tendência a se desenvolver tecnologias para o rastreamento de diversos aspectos de saúde e bem-estar, com impactos positivos para a saúde e o comportamento de seus usuários. Em 2018, Han et al. conduziram uma revisão sistemática de diversos estudos sobre mudanças comportamentais relacionadas ao uso de aplicativos móveis de saúde para avaliar a efetividade no uso desses aplicativos. Dos 20 estudos analisados, em 17 foram reportados, além de um alto nível de satisfação, mudanças de comportamento positivas para os usuários com relação a aspectos como atividade física, mudanças na dieta e aderência à medição e ao tratamento utilizados.

O presente trabalho de conclusão de curso propõe projetar e desenvolver uma solução tecnológica para o controle e rastreamento de crises de enxaqueca.

1.1. Justificativa

Com o avanço da tecnologia móvel e a crescente presença de soluções digitais para o registro e controle de aspectos de saúde, os diários utilizados para controle da enxaqueca também têm achado espaço no meio digital. Existem diversas vantagens em substituir os diários de enxaqueca de seu tradicional formato de papel para um formato eletrônico, em dispositivos portáteis. Diários em papel podem ser volumosos e facilmente perdidos, além de incômodos para os pacientes, pois os dados devem ser preenchidos à mão (HUNDERT et al., 2014). Isso causa problemas quanto à aderência ao preenchimento, que ocorrem em número consideravelmente menor se esse preenchimento se der em um meio eletrônico, como já observado na literatura (STONE et al., 2003, ALLENA et al., 2021).

A manutenção de diários por meio de dispositivos eletrônicos portáteis passou a ser utilizada, também, para a coleta de dados clínicos na literatura científica. A partir de 2010, podem ser encontrados diversos estudos que têm como instrumento diários eletrônicos tanto desenvolvidos especificamente para o próprio estudo, quanto desenvolvidos por terceiros, já consagrados e disponíveis no mercado.

Um exemplo de sistema desenvolvido exclusivamente para um estudo foi usado no trabalho de Allena et al. (2012). No artigo, os autores utilizaram um *palmtop*, com um aplicativo desenvolvido sob a supervisão de um dos autores, para avaliar a aplicabilidade de um diário eletrônico como substituto dos registros em papel utilizados no tratamento de enxaquecas causadas por uso excessivo de medicamentos (MOH). Além do bom nível de aceitação dos usuários e do percentual de aderência consideravelmente maior, comparado com registros em papel, os autores citam o potencial que o sistema desenvolvido e utilizado por eles tem de ser usado em contextos clínicos.

Mais recentemente, em 2018, Vo et al. (2018) utilizaram dados coletados de mais de 3 mil usuários do aplicativo *Migraine Buddy* – na época, o aplicativo de diário de enxaqueca mais utilizado no mundo – para avaliar o impacto da enxaqueca em suas atividades diárias, em sua produtividade no trabalho e na quantidade de medicamentos que necessitavam. Apesar do possível viés do estudo, devido principalmente ao fato de que os dados analisados partiram de um aplicativo que tem como finalidade a

autoavaliação dos usuários, os resultados encontrados pelos autores foram consistentes com resultados anteriormente publicados sobre o impacto da enxaqueca.

Tendo em vista os benefícios trazidos por esse tipo de tecnologia, que podem ser vistos até mesmo em contextos clínicos, esse trabalho de conclusão de curso tem como objetivo projetar e desenvolver uma solução tecnológica para o controle e rastreamento de crises de enxaqueca.

Por ser uma doença complexa, que tem causas tanto genéticas quanto ambientais (PIANE et al., 2007), a enxaqueca apresenta diversos fatores envolvidos. Entre os gatilhos (agentes indutores de um ataque de enxaqueca), por exemplo, estão fatores comportamentais, como estresse, sobrecarga mental e sono irregular; ambientais, como odores e ruídos; assim como dietéticos (BORKUM, 2016). Muitos aplicativos já existentes, como os citados acima, contam com equipes multidisciplinares e consultoria de profissionais da saúde para sua implementação e têm a intenção de abarcar o maior número possível de fatores, o que pode até mesmo deixar o registro de uma crise complicado para o usuário.

A solução proposta neste trabalho de conclusão de curso não pretende abarcar todos os fatores envolvidos, assim como não terá a intenção de ser uma ferramenta de diagnóstico. O foco primário da solução é ser um assistente de organização pessoal, com a marcação dos dias e horários dos ataques em um calendário, para visualização do aumento ou não do intervalo de tempo entre eles, da sua intensidade e da quantidade de medicamento analgésico administrada. A simplificação dos registros, usando apenas as características mais importantes das crises, de acordo com a literatura, pode trazer vantagens ao usuário, se comparado com as tecnologias mais utilizadas, que podem ser demasiado complexas. Além disso, essa solução será desenvolvida voltada para pacientes que sofram dos distúrbios relacionados à enxaqueca, como classificados na décima edição da Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde (CID-10 G43) (OMS, 2019), e que façam tratamento acompanhado por profissional especializado.

A escolha pela solução de um aplicativo para uso em dispositivos móveis está relacionada com a questão da praticidade para quem sofre de enxaqueca. Para o desenvolvimento, será utilizado o framework Flutter (FLUTTER, [s.d.]), seguindo os princípios da *Clean Architecture* (Arquitetura limpa) (MARTIN, 2017). Dentre outras vantagens, o Flutter é multiplataforma, pode compilar aplicativos iOS e Android com o mesmo código fonte, e tem uma performance similar a de um aplicativo desenvolvido

nativamente. A *Clean Architecture* tem sido bastante utilizada com Flutter e, além de ajudar na legibilidade do código, traz independência e flexibilidade entre os componentes do sistema (REŠETÁR, 2019; YILDIRIM, 2019; MOURA, 2020), facilitando a manutenção e a inclusão de novas funcionalidades futuras.

1.2 Objetivo Geral

Projetar e desenvolver uma solução tecnológica para o registro de diários para controle e rastreamento de crises de enxaqueca.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho de conclusão são:

- Desenvolver um diário simples, de fácil preenchimento para o usuário com apenas os fatores mais importantes a serem rastreados;
- Projetar o software utilizando conceitos de Arquitetura Limpa (Clean Architecture) (MARTIN, 2017);
- Utilizar o framework Flutter para o desenvolvimento.

1.4 Estrutura do trabalho

No Capítulo 2, são apresentadas e analisadas soluções similares à solução oferecida neste trabalho. O Capítulo 3 traz uma breve apresentação e explicação das tecnologias e padrões de *design* utilizados na implementação da solução. O Capítulo 4 apresenta a solução conceitual, com o projeto do sistema proposto. O Capítulo 5 apresenta o relato e a descrição da implementação da solução. As conclusões finais, com uma análise da solução implementada, são apresentadas no Capítulo 6.

2. TRABALHOS CORRELATOS

Atualmente, existem disponíveis no mercado diversas tecnologias para monitoramento de enxaqueca. Para um levantamento de soluções relacionadas à deste trabalho de conclusão de curso, foram selecionadas três – *Migraine Buddy*, *Diário da Cefaleia* e *Migraine Monitor* – seguindo os critérios abaixo.

- Aplicativos Android disponíveis na Play Store;
- Palavras-chave: “enxaqueca”, “diário”, “saúde”;
- Nota dos usuários maior que 4.0;
- Mais de 10 mil downloads;
- Última atualização a mais de 6 meses.

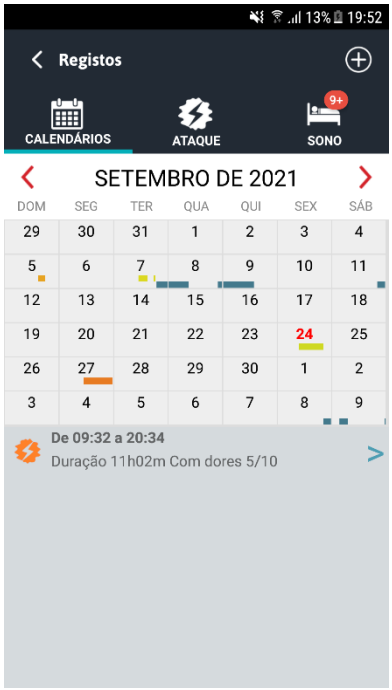
2.1 Migraine Buddy

Com mais de 1 milhão de downloads desde 2014, o Migraine Buddy é o aplicativo para rastreios de crises de enxaqueca mais baixado da Play Store. Para utilizar o aplicativo, o usuário precisa apenas um e-mail e senha de login e sua região de moradia.

O registro da crise se dá a partir de um botão principal na tela inicial (Figura 2.1), e há um número grande de variáveis a serem preenchidas no registro. O usuário seleciona o horário de início e fim da dor, e passa para as próximas variáveis: tipo de ataque, intensidade da dor, local da cabeça onde há dor, medicação administrada, método de alívio, sintomas relacionados, outros sintomas que poderiam ter surgidos (“sentiu que estava para surgir”), como a dor afetou outros fatores da vida, onde o usuário estava quando surgiu a dor, possíveis gatilhos, menstruação e informações adicionais. Apesar de a maioria não ser obrigatória, cada uma dessas variáveis é apresentada em uma tela diferente e exige ao menos um clique do usuário para passar para a próxima tela. As crises podem ser visualizadas em um calendário, que conta com a marcação dos dias de dor com uma cor, de acordo com a intensidade da dor, além dos intervalos de horário de sono marcados pelo usuário (Figura 2.2).

Figura 2.1 – Tela inicial do *Migraine Buddy*

Figura 2.2 – Tela de calendário do *Migraine Buddy*



Além do registro e visualização das crises, o aplicativo conta com outras funcionalidades como previsão de mudanças de pressão atmosférica, visualização e registro de consultas médicas e detecção e rastreamento de padrões de sono.

2.2 Diário da Cefaleia

O Diário da Cefaleia, da Libbs, é um dos aplicativos de diário de enxaqueca mais baixados da Play Store, com mais de 50 mil downloads.

Para registro da crise, é apresentada uma tela com cinco abas, que se abrem para mostrar mais informações que devem ser preenchidas (Figura 2.3). As cinco abas são a data (que inclui local da ocorrência), intensidade da dor, fatores desencadeantes (gatilhos), local da cabeça onde há dor e medicamento.

Figura 2.3 – Tela de inicial do *Diário da Cefaleia*

Como funcionalidades adicionais, o usuário pode enviar um relatório de sua crise a um médico. Há também um indicador da temperatura e condição climática no momento, de acordo com a região do usuário. Porém, esse indicador de condição climática não é utilizado nos registros de crises.

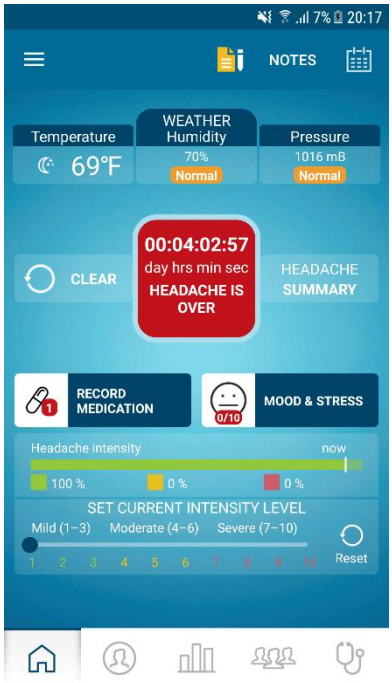
2.3 Migraine Monitor

O Migraine monitor tem mais de 10 mil downloads na Play Store e uma nota de 5.0 (com cerca de 250 reviews).

No registro da crise, após o usuário informar a hora de início, um cronometro é iniciado se não há uma data de fim informada. Na tela inicial, o usuário pode clicar no

mesmo botão utilizado para registrar a crise a fim de parar o cronômetro, o que facilita o registro da duração da crise (Figura 2.4). Para registro, após informar data de início e fim da crise, o usuário deve informar se houve aura, a intensidade da dor, o local da cabeça onde houve dor, gatilhos, sintomas associados e medicação.

Figura 2.4 – Tela inicial do *Migraine Monitor,* com cronômetro



O aplicativo conta também com uma funcionalidade de relatórios que mostra barras coloridas, indicando a duração (tamanho da barra) e a intensidade da dor (cor da barra) a cada dia, além de um relatório diário de variação de temperatura e outro de variação de pressão atmosférica.

2.4 Comparativo

Na Tabela 2.1 podemos observar um comparativo entre os três aplicativos analisados. Quanto ao número de funcionalidades disponíveis, aparte as básicas de visualização de crises, o aplicativo Migraine Buddy é o mais completo, seguido do Migraine Monitor. Para se ter um parâmetro de referência da experiência do usuário no momento do cadastro, foi usado como critério, também, o número mínimo de cliques necessários para o cadastro de uma crise, assim como o número de variáveis apresentadas para o usuário. Apesar de completo, pode-se observar que o Migraine Buddy é o que pode ser mais complexo no cadastro de uma crise.

Tabela 2.1 - Comparativo entre as tecnologias analisadas

Característica	**Migraine Buddy**	**Diário da Cefaleia**	**Migraine Monitor**
Registro da crise			
Nr. min. de cliques	17	12	9
Nr. de variáveis mostradas	15	5	6
Outras características			
Condição climática	Não	Sim	Sim
Pressão atmosférica	Sim	Não	Sim
Monitor de sono	Sim	Não	Não
Agenda de consultas médicas	Sim	Não	Sim
Exportação de dados	Sim	Sim	Não
Forum de discussões	Sim	Não	Sim

3. FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento da solução proposta neste trabalho de conclusão de curso, as tecnologias utilizadas serão o framework Flutter (FLUTTER, [s.d.]), em conjunto com a linguagem de programação Dart (DART, [s.d.]) para o desenvolvimento, e as ferramentas da *Google Firebase*, com o *Firebase Authenticaton* para gerenciamento de autenticação de usuários e o *Cloud Firestore* para gerenciamento da camada de persistência de dados. Um diferencial adicional da solução proposta é o uso de *design patterns* (padrões de projeto) que visam facilitar a posterior manutenção e extensão da solução desenvolvida. Serão utilizados os seguintes *design patterns*: *Clean Architecture* (Arquitetura Limpa), que na sua definição trata-se de um conjunto de *design patterns* a serem utilizados no domínio de arquitetura de software (MARTIN, 2019), e *BLoC* (ANGELOV, 2019), utilizado para o gerenciamento de estados da aplicação. Como as tecnologias e *design patterns* utilizados no desenvolvimento da solução proposta são recentes no mundo do desenvolvimento de aplicativos, eles serão introduzidos nesse capítulo para que possam ser mais bem entendidos nos capítulos seguintes. As tecnologias e padrões serão detalhados nas seções seguintes.

3.1 Tecnologias

3.1.1. Flutter

O Flutter é um kit para desenvolvimento de software (SDK) com foco em dispositivos móveis, gratuito e de código aberto, criado por um dos times de engenheiros de software da *Google* e lançado oficialmente em 2017. Esse SDK evolui do projeto *Sky*, realizado com o objetivo de utilizar a linguagem Dart – também criada por um time da *Google* – para desenvolvimento de aplicativos *Android* como alternativa ao *Java*, focando em velocidade e responsividade (AMADEO, 2015).

A arquitetura do framework Flutter é desenhada em um sistema de camadas, em que cada camada é composta por bibliotecas independentes entre si, mas que dependem da camada mais abaixo. Assim, cada camada do framework é desenvolvida para ser opcional e substituível (FLUTTER, [s.d.]). Uma das principais características dessa arquitetura é o uso, nas camadas superiores, de *widgets* representando os blocos que

compõem um aplicativo. Cada componente da interface de usuário de um aplicativo – botões, textos, imagens, etc. – são um *widget*, e o código forma uma árvore hierárquica, com *widgets* pais e *widgets* filhos. Tudo isso traz ao desenvolvedor as vantagens de personalização e agilidade no desenvolvimento do código.

Entre outras vantagens de se o usar o Flutter, pode-se destacar mais duas. Primeiramente, o framework do SDK, que utiliza a linguagem Dart, é compilado diretamente para código nativo, o que dá ao aplicativo desenvolvido ótima performance. Por fim, o código desenvolvido em Flutter pode ser compilado para dispositivos *Android* e *iOS* sofrendo mudanças apenas nas camadas de apresentação (montagem de telas).

3.1.2. Firebase e Cloud Firestore

O *Firebase* é uma plataforma de *MBaaS* (*Mobile backend as a service*) adquirida pela *Google* em 2014, que fornece serviços de *backend* para aplicativos móveis. Entre os serviços, estão, principalmente o de autenticação de usuários e armazenamento de dados.

Para o armazenamento de dados, o *Firebase* conta com dois serviços: *Realtime Database* e *Cloud Firestore*. Ambos são bancos de dados *NoSQL*, que contam com atualizações dos dados em tempo real. Para a solução desenvolvida nesse trabalho de conclusão de curso, foi escolhido o *Cloud Firestore*, que é uma evolução do *Realtime Database*, e tem um modelo de dados baseado em coleções de documentos, com desempenho aprimorado.

3.2. Design Patterns

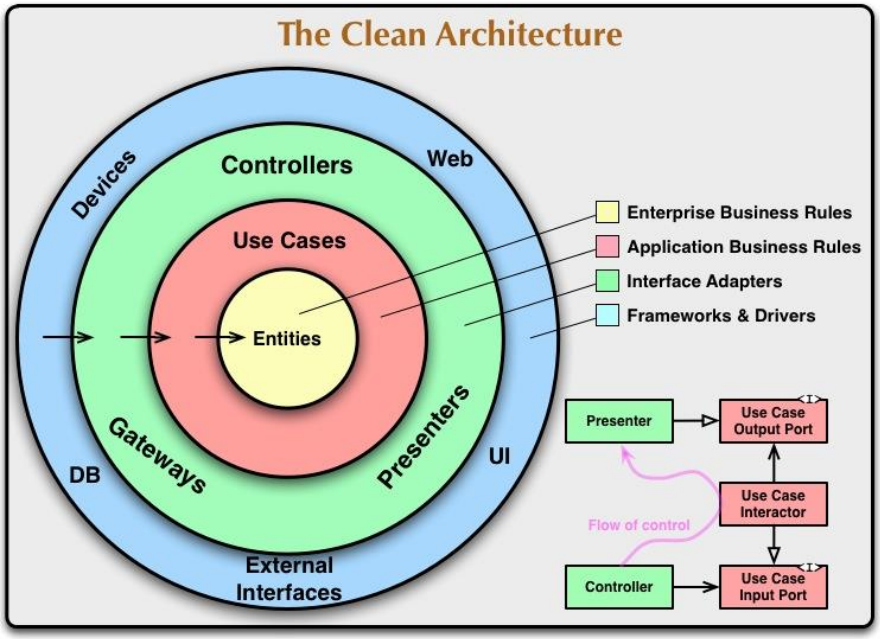
3.2.1. Arquitetura Limpa

Nos últimos anos pode-se observar o surgimento de muitas ideias a respeito da arquitetura de softwares e sistemas, evoluídas dos conceitos de código limpo (*clean code*). Todas elas são similares e têm o mesmo objetivo principal: a separação das funções de cada parte do software, fazendo a divisão do software em camadas (MARTIN, 2012). Entre esses padrões de arquitetura está a Arquitetura Limpa (*Clean Architecture*) idealizada por Robert C. Martin (conhecido como Uncle Bob), e publicada oficialmente pela primeira vez em seu livro “Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software” (MARTIN, 2019).

Martin destaca cinco características que um sistema desenvolvido seguindo os princípios de sua arquitetura deve ter quando finalizado: i) independência de frameworks, ii) independência de interface de usuário, iii) independência de banco de dados, iv) independência de qualquer agente externo e v) testabilidade. Em primeiro lugar, o sistema não deve ficar restrito às limitações de frameworks e bibliotecas externas, e sim usá-los como ferramentas. Da mesma forma, o tipo de banco de dados e interface de usuário devem ser substituíveis sem a necessidade de mudanças no resto do sistema. Em resumo, qualquer fator externo precisa ser independente das regras de negócio do sistema. Isso tudo traz ao sistema a última característica citada: as regras de negócio poderão ser testadas sem a necessidade da interface, banco de dados, ou qualquer outro elemento externo (MARTIN, 2012).

Na Arquitetura Limpa, o projeto do software é dividido em quatro camadas, representadas por círculos concêntricos. De dentro para fora, temos as camadas de Entidades (*Entities*), Casos de Uso (*Use Cases*), Adaptadores de Interface (*Interface Adapters*) e Frameworks e Drivers. Todas essas camadas são permeadas por uma regra prevalecente, a Regra da Dependência, que diz que as dependências do código apontam de fora para dentro, isto é, nada de um círculo mais interno sabe sobre um círculo mais externo. As Entidades, por exemplo, não sabem ou dependem de nada dos Casos de Uso, porém os Casos de Uso dependem e referenciam as Entidades. Martin (2012) representa esses conceitos básicos da Arquitetura Limpa com o diagrama reproduzido na Figura 3.1.

Figura 3.1 - Representação do funcionamento da Arquitetura Limpa, conforme Martin (2012).



Fonte: Disponível em https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, acesso em 06/09/2021.

A camada das Entidades é onde ficam as regras de negócio empresariais. Nela, definimos as propriedades de cada entidade, seus tipos e acessos. A comunicação com as entidades é feita pela camada de Casos de Uso, onde ficam as regras de negócio da aplicação. Os *controllers*, *presenters* e repositórios, que fazem a tradução dos casos de uso para os dispositivos externos, ficam na camada de Adaptadores, e os dispositivos externos em si (frameworks, bancos de dados, integração com outros sistemas, etc.) ficam na camada de Frameworks e Drivers.

3.2.2. BloC

O *BLoC* (*Business Logic Component*) é um dos principais padrões de design recomendados por desenvolvedores da *Google* para o gerenciamento de estados no Flutter (SURI, 2018). O *BloC* ajuda a separar a interface de usuário das regras de negócio usando conceitos de programação reativa e *streams* de dados, quer dizer, um fluxo de dados assíncrono.

Utilizando *BLoC* com Flutter, o fluxo de dados entre a camada de interface de usuário e a camada mais interna é feita através de um controlador. A camada de *BLoC*

“escuta” por eventos, disparados pelos *widgets* da interface de usuário e, então, executa as regras de negócio chamando a camada de casos de uso (no caso de um sistema estruturado com a Arquitetura Limpa). A execução das regras de negócio poderá alterar o estado do sistema (por exemplo, a atualização de uma variável). Essa mudança de estado será propagada para a interface do usuário, que pode ou não alterar o que está sendo exibido em tela.

Durante o desenvolvimento da solução proposta neste trabalho de conclusão de curso, será utilizado o *package flutter_bloc* (ANGELOV, 2019), que tem o objetivo de facilitar a integração do padrão *BLoC* no Flutter.

4. SOLUÇÃO CONCEITUAL

Este trabalho de conclusão de curso apresenta o projeto e desenvolvimento de um aplicativo assistente de organização pessoal, voltados a indivíduos que sofrem e fazem tratamento de enxaqueca. O aplicativo tem como principal funcionalidade a marcação dos dias e horários de crises em um calendário, apresentando para o usuário a visualização de fatores como o aumento ou não do intervalo de tempo entre as crises, a intensidade das crises e a quantidade de medicamento analgésico administrada em cada crise. Apesar de a solução não ter intenção de ser uma ferramenta de diagnóstico ou coleta de dados para análise clínica, sua temática e foco são voltados ao tratamento da doença. Por isso, para a concepção das funcionalidades, em especial das variáveis que o usuário poderá rastrear, se faz necessária uma rápida referência à literatura especializada.

4.1. Levantamento das variáveis de enxaqueca que serão utilizadas pela solução proposta

Em um estudo de Hundert et al. (2014), foi feita uma análise comparativa de diversas soluções para dispositivos móveis voltadas à manutenção de diários de enxaqueca disponíveis na época. Um dos critérios utilizados pelos autores para a comparação foi a relevância das variáveis medidas por cada aplicativo analisado. Para determinar quais variáveis seriam relevantes, foram consultados diversos profissionais especializados, e as variáveis mencionadas por 50% ou mais dos profissionais foram as consideradas pelos autores para a avaliação dos critérios. Na Tabela 4.1 temos um resumo das variáveis levantadas pelos autores.

Tabela 4.1 - Variáveis mais relevantes para rastreio de crises de enxaqueca, conforme levantamento de Hundert et al. (2014)

Variável	**Porcentagem dos especialistas que recomendaram**
Intensidade da dor	100
Gatilhos para a crise	90
Medicamento administrado	90
Sintomas associados	70
Frequência de crises	60
Incapacidade relacionada à dor	50
Duração da dor	50

Fonte: Hundert et al. (2014). Disponível em: ttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4147710/

Juntamente com o levantamento de Hundert et al. (2014), foi feita no âmbito do levantamento dos requisitos nesse trabalho de conclusão de curso, uma análise de modelos de diários de enxaqueca disponíveis na Web. Para se atribuir a relevância dos diários analisados, foi utilizado como critério de inclusão somente os modelos de diário disponibilizados pelo site de uma organização dedicada ao tratamento e conscientização sobre a enxaqueca, tendo médicos especializados como parte do conselho de membros. Foram selecionados diários disponibilizados por três instituições: *The National Headache Foundation*, *National Migraine Centre* e *The Migraine Trust*. A Tabela 4.2 apresenta um quadro comparativo das variáveis recomendadas nos diários disponibilizados.

Tabela 4.2 - Comparação entre as variáveis de enxaqueca usadas pelos diários recomendados. (a presença da variável no diário é marcada com um X).

Variável	NHF	NMC	TMT
Dia	X	X	X
Hora do início	X	X	
Hora do fim	X		
Duração	X		X
Intensidade da dor	X	X	X
Sintomas relacionados	X		X
Medicamento	X	X	X
Hora do medicamento		X	
Gatilhos	X		X
Nível de alívio	X		

Fontes: https://headaches.org/wp-content/uploads/2018/08/HEADACHE-DIARY.pdf, https://www.nationalmigrainecentre.org.uk/migraine-and-headaches/migraine-and-headache-diary/, https://migrainetrust.org/live-with-migraine/self-management/keeping-a-migraine-diary/#page-section-2

Fazendo uma comparação com as variáveis dos três diários e as recomendadas em Hundert et al. (2014), podemos observar que as mais frequentes – além do dia, variável presente em qualquer diário – são a intensidade da dor, horas de início e fim ou duração (variável composta das horas de início e fim), medicamento utilizado, gatilhos e sintomas associados.

Para a solução desenvolvida nesse trabalho de conclusão de curso, as variáveis principais serão aquelas que podem ser representadas em unidades discretas e apresentadas em um calendário de maneira visual, por meio de cores. Sendo assim, foram selecionadas as seguintes variáveis: horas de início e fim, duração, intensidade e medicamento administrado. Apesar de haver muitos medicamentos que podem ser utilizados por indivíduos que sofrem de enxaqueca, a variável "medicamento administrado" será utilizada por ser de bastante importância, sendo que o número de medicamentos listados ao usuário poder ser reduzido, extraído da listagem dos principais medicamentos recomendados. Outras duas variáveis também presentes na maioria dos diários recomendados são os *gatilhos* que podem ter causado a crise e os *sintomas* relacionados à crise. Essa duas varáveis são bastante específicas de cada indivíduo e, em um primeiro momento, serão adicionadas ao registro da crise, na solução desenvolvida, apenas como observações adicionais (em alguns diários, como o da Migraine Trust, esses

fatores também são adicionados como comentários adicionais).

A partir desse levantamento inicial, será apresentado, nas seções seguintes, o projeto da solução conceitual proposta neste trabalho de conclusão de curso, com o levantamento das histórias de usuário/requisitos funcionais (Seção 4.2), protótipo de telas (Seção 4.3), diagrama entidade-relacionamento (Seção 4.4) e diagrama de classes (Seção 4.5).

4.2. Histórias de usuário/requisitos funcionais

As histórias de usuário com as funcionalidades desejadas para a solução serão organizadas por ordem de prioridade. Primeiramente, serão listadas as histórias de cadastro e edição das crises e variáveis atribuídas pelo usuário para cada crise, essenciais para a existência do sistema. Depois, serão listadas as histórias de visualização das crises e estatísticas para rastreamento da evolução do quadro de enxaqueca. Por fim, as histórias de edição de atributos de visualização não essenciais, login e sincronização de dados. Para melhor visualização, as histórias de usuário estão dispostas em um quadro (Tabela 4.3). O sistema terá um único ator, que executará todas as funcionalidades.

Tabela 4.3. Histórias de usuário do sistema de Diário de Enxaqueca

NR.	HISTÓRIA	O USUÁRIO DESEJA	PARA
1	REGISTRAR CRISE	Clicar em um botão e registrar um episódio de crise de enxaqueca.	Poder saber hora de início, hora de fim, intensidade, entre outras características da sua crise naquele dia.
2	ADICIONAR MEDICAMENTO	Adicionar um medicamento à lista de medicamentos apresentados no registro da crise.	Ter em sua lista algum medicamento diferente dos listados pelo sistema.
3	EXCLUIR MEDICAMENTO DA LISTA DE MEDICAMENTOS	Excluir um medicamento da lista de medicamentos apresentados no registro da crise.	Excluir de sua lista algum medicamento que não utilize mais ou tenha registrado por engano.
4	EDITAR MEDICAMENTO DA LISTA DE MEDICAMENTOS	Editar um medicamento da lista de medicamentos apresentados no registro da crise.	Editar algum medicamento que tenha registrado incorretamente.
5	ADICIONAR GATILHO À CRISE	Adicionar um gatilho à crise registrada.	Ter registrado na crise um possível fator causador, caso tenha percebido.
6	VISUALIZAR LISTA DE GATILHOS	Visualizar uma lista de gatilhos para as crises já cadastradas por ele.	Poder ter uma lista de gatilhos frequentes e não precisar digitar o mesmo gatilho caso ocorra em crise futura.
7	EXCLUIR GATILHO DA LISTA DE GATILHOS	Excluir um gatilho da lista de gatilhos já cadastrados.	Excluir de sua lista algum gatilho que tenha registrado por engano.
8	EDITAR GATILHO DA LISTA DE GATILHOS	Editar um gatilho da lista de gatilhos já cadastrados.	Editar algum gatilho que tenha registrado incorretamente.
9	ADICIONAR SINTOMA RELACIONADO À CRISE	Adicionar um sintoma relacionado à crise registrada.	Ter registrado na crise um sintoma relacionado que tenha percebido.

10	VISUALIZAR LISTA DE SINTOMAS RELACIONADOS	Visualizar uma lista de sintomas relacionados às crises já cadastrados por ele.	Poder ter uma lista de sintomas frequentes e não precisar digitar o mesmo sintoma caso ocorra em crise futura.
11	EXCLUIR SINTOMA DA LISTA DE SINTOMAS RELACIONADOS	Excluir um sintoma da lista de sintomas já cadastrados.	Excluir de sua lista algum sintoma que tenha registrado por engano.
12	EDITAR SINTOMA DA LISTA DE SINTOMAS RELACIONADOS	Editar um sintoma da lista de sintomas já cadastrados.	Editar algum sintoma que tenha registrado incorretamente.
13	VISUALIZAR CALENDÁRIO	Visualizar, em um calendário, os dias em que teve uma crise marcados.	Poder ter uma visualização espacial da frequência e período entre crises, conforme as semanas e meses
14	VISUALIZAR CRISE	Clicar em um dia, a partir do calendário.	Ver as informações da crise de enxaqueca que teve nesse dia.
15	EDITAR CRISE	Editar uma crise que já tenha cadastrado.	Editar alguma variável da crise que tenha registrado incorretamente.
16	EXCLUIR CRISE	Excluir uma crise que já tenha cadastrado.	Excluir alguma crise que tenha registrado incorretamente.
17	VISUALIZAR CRISES POR INTENSIDADE	Que o calendário marque os dias em que houve crise conforme a intensidade da crise registrada, marcando cada nível de intensidade com uma cor.	Poder saber, visualmente, a frequência com que cada nível de intensidade ocorre.

18	VISUALIZAR CRISES POR TURNO	Que o calendário marque os dias em que houve crise conforme o turno em que foi registrado o início da crise, marcando cada turno com uma cor.	Poder saber, visualmente, a frequência com que cada turno de início ocorre.
19	VISUALIZAR CRISES POR MEDICAMENTO ADMINISTRADO	Que o calendário marque os dias em que houve crise conforme o medicamento que foi administrado, marcando cada tipo de medicamento com uma cor.	Poder saber, visualmente, a frequência com que toma cada medicamento
20	VISUALIZAR CRISES POR DURAÇÃO DA CRISE	Que o calendário marque os dias em que houve crise conforme o tempo de duração da crise, marcando cada intervalo de tempo com uma cor.	Poder saber, visualmente, a frequência com que ocorrem crises de curta, média ou longa duração.
21	VISUALIZAR MÉDIA DE QUANTIDADE DE CRISES POR PERÍODO	Selecionar um tipo de período (ex. semanas, meses, anos) e visualizar a média de crises que teve dentro daquele período.	Poder rastrear a evolução de seu quadro por meio do aumento ou diminuição da frequência de crises.
22	VISUALIZAR MÉDIA DE MEDICAMENTOS ADMINISTRADOS	Selecionar um período e visualizar a média de medicamentos administrados, de um ou mais medicamentos específicos ou de todos os medicamentos, em geral.	Poder rastrear a evolução de seu quadro por meio dos medicamentos administradas.

23	VISUALIZAR MÉDIA DE DURAÇÃO DE CRISES	Selecionar um período e visualizar a média de duração das crises dentro daquele período.	Poder rastrear a evolução de seu quadro por meio do aumento ou diminuição da duração das crises.
24	ATRIBUIR CORES PARA INTENSIDADES	Atribuir cores para cada nível de intensidade de dor.	Poder editar a visualização do calendário por intensidade com as cores de sua preferência.
25	ATRIBUIR CORES PARA MEDICAMENTOS	Atribuir cores para cada medicamento cadastrado.	Poder editar a visualização do calendário por medicamento com as cores de sua preferência.
26	ATRIBUIR CORES PARA TURNOS	Atribuir cores para cada turno de início de crise.	Poder editar a visualização do calendário por turno com as cores de sua preferência.
27	ATRIBUIR CORES PARA INTERVALOS DE TEMPO	Atribuir cores para intervalos de tempo escolhidos.	Poder editar a visualização do calendário por duração com as cores dos intervalo de sua preferência.
28	FAZER LOGIN NO SISTEMA	Fazer login no sistema com um usuário e senha.	Poder ter suas configurações e preferências salvas.
29	SINCRONIZAR DADOS	Sincronizar seus dados em um dispositivo novo	Poder ter suas preferências e registros já cadastrados em um novo dispositivo.

4.3. Protótipos de telas

A tela principal do sistema (Figura 4.1) terá um botão de ação flutuante, facilmente clicável com o polegar direito, como o principal botão para cadastro de uma crise. Além dele, algumas informações úteis serão disponibilizadas por meios de widgets clicáveis.

Figura 4.1 - Tela inicial do sistema.

Ao clicar no botão de cadastro, a tela de cadastro de crise (Figura 4.2) aparecerá para o usuário. Nela haverá 5 botões com as funcionalidades representadas por ícones: hora de início, hora de fim, intensidade da dor, medicamento administrado e observações adicionais (adicionar gatilho ou sintoma relacionado). Os botões de data e hora de início e fim apresentarão um *date picker*. O botão de intensidade apresentará um *slider* com os níveis possíveis para escolha. O botão de medicamento apresentará uma lista dos medicamentos já cadastrados, para escolha, assim como a possibilidade de cadastrar novo medicamento. Por fim, o botão de “mais” apresentara as opções de adicionar um gatilho para a crise ou sintoma relacionado.

Figura 4.2 - Tela de cadastro de crise, com fluxo de seleção de intensidade e medicamento

Ao clicar no widget do calendário, na página inicial, o usuário é levado ao calendário, onde pode ver os dias de crise marcados com cores (Figura 4.3). Ao clicar em um dos dias de crise, são apresentados os detalhes daquela crise (Figura 4.4).

Figura 4.3 - Calendário mostrando os dias de crise.

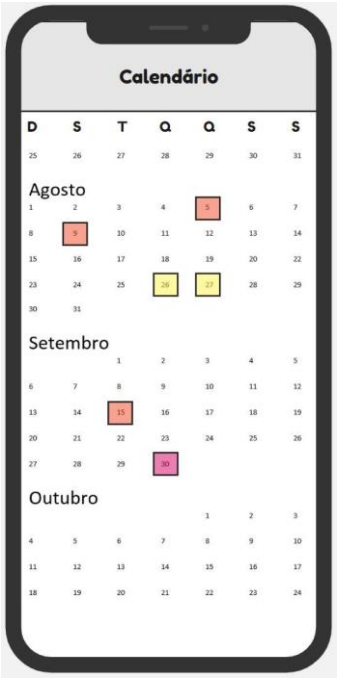
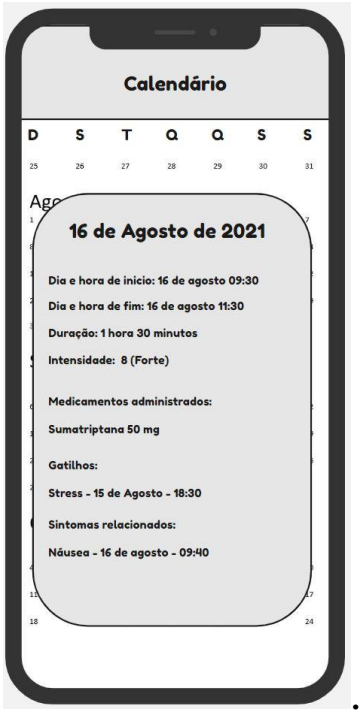


Figura 4.4 - Calendário mostrando detalhes da crise.



4. Diagrama Entidade-Relacionamento

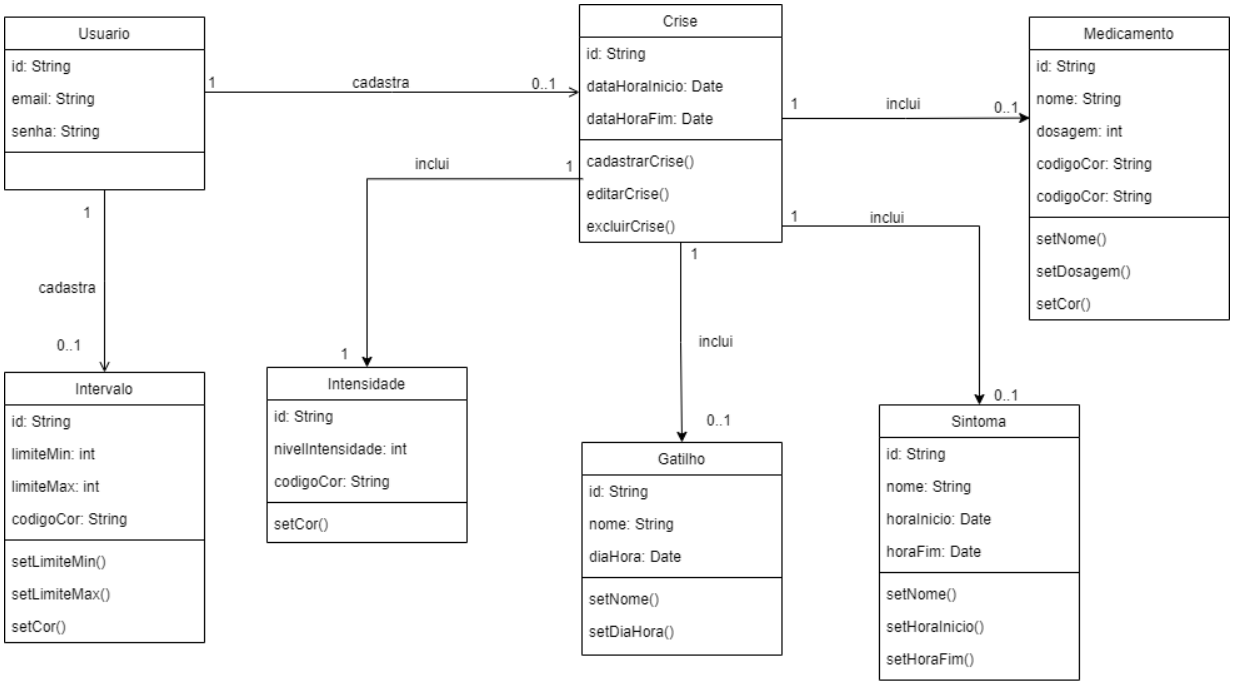
Para a solução proposta neste trabalho de conclusão de curso, será utilizado o banco de dados *NoSQL Cloud Firestore*. Por isso, a modelagem das entidades do sistema e seus relacionamentos foi feita utilizando o padrão de projeto de banco de dados não relacional. Em um banco de dados não relacional, cada entidade é uma coleção de documentos, e os documentos são as instâncias das entidades, com seus atributos organizados em árvores de dados. O diagrama entidade-relacionamento correspondente a uma modelagem de banco de dados não relacional da solução proposta está apresentado na Figura 4.5.

Figura 4.5 - Diagrama de entidades-relacionamento. Os tipos de dados são: pk – primary key (chave primária), arr – array, num – number, obj –object, str – string, ts – timestamp

4.4. Diagrama de classes

Baseando-se nas entidades, pode ser gerado o diagrama de classes para o sistema, conforme a Figura 4.6.

Figura 4.6 - Diagrama de classes para a solução conceitual proposta



5. IMPLEMENTAÇÃO

Neste capítulo, discutiremos a implementação do aplicativo, seguindo o projetado na fase de concepção, conforme o descrito no Capítulo 4 (Solução Conceitual).

Devido ao fato de a solução ter sido implementada utilizando-se conceitos e estrutura baseada na Arquitetura Limpa (introduzida na Seção 3.2.1), é importante iniciarmos com uma descrição detalhada da estrutura geral do projeto e de como as classes e packages se comunicam entre si. A seguir, é importante detalhar como acontece o funcionamento do BLoC (introduzido na Seção 3.2.2) neste projeto, pois é a camada do BLoC a principal ligação entre as ações do usuário e a lógica da aplicação. Por fim, será demonstrado o produto final com especificidades do código, assim como desafios enfrentados no uso do framework Flutter e da linguagem Dart.

5.1. Estrutura geral do projeto

Um dos principais conceitos que regem a Arquitetura limpa é o da independência que as camadas do projeto têm entre si e entre quaisquer agentes externos. Para atingir essa independência na implementação do aplicativo, que começa a ser detalhada a seguir, o projeto principal é dividido em três camadas: *Presentation* (Apresentação), *Domain* (Domínio) e *Data* (Dados). O fluxo de chamada do sistema, desde a interação do usuário com o aplicativo até a comunicação do aplicativo com o banco de dados externo segue a seguinte ordem: *Presentation => Domain => Data*.

A camada de apresentação é onde são implementados os componentes visuais do aplicativo: as telas e *widgets*. É nela que encontramos os recursos específicos do *framework* Flutter. Além das telas e *widgets*, também ficam na camada de apresentação os componentes que fazem a gerência de estados do aplicativo; no caso da solução implementada neste trabalho, os componentes do BLoC. O BLoC é o componente que fará a comunicação entre a camada de apresentação e a camada de domínio, “escutando” por eventos despachados pelos *widgets* e, de acordo com o evento disparado, chamando os casos de uso.

Na camada de domínio ficam a lógica de negócio (casos de uso) e os objetos (entidades) do sistema. Essa camada é a mais estável, pouco suscetível a mudanças da camada de dados e de apresentação. Além das classes de casos de uso e de entidades, a camada de domínio apresenta uma classe abstrata de repositório, que é implementada na camada de dados. Essa classe define o contrato do que deverá ser feito pela implementação do repositório, na camada de dados.

Para a declaração dos métodos que representam os casos de uso, foram utilizado recursos do paradigma de programação funcional, com ajuda da biblioteca *dartz* (DARTZ, 2019), de modo a tratar os erros que possam ocorrer quando os casos de uso são acionados. O *dartz* funciona fornecendo dois tipos de retornos possíveis a um método, declarando-o como no exemplo a seguir, do caso de uso de cadastro de nova crise:

```
Future<Either<Failure, String>> newCrise(Crise crise);
```

O método, declarado dessa forma, retornará ou o valor da esquerda (a falha) ou o valor da direita (no caso do cadastro com sucesso no banco de dados, o id único do registro da crise, no tipo *String*).

A camada de dados contém, além da implementação do repositório, as classes de modelo das entidades (*Model)* e as fontes de dados (*Datasources*). Nesta última, é onde o código específico do banco de dados se encontra. No caso deste projeto, todo o código e regras específicos do Firestore. A conversão das mensagens recebidas do banco de dados, em formato Json, para objetos em linguagem Dart é feita por métodos das classes *Model*. Desta forma, o protocolo de comunicação entre a aplicação e o repositório se restringe à camada de dados, não "vazando" para a camada de domínio.

Além dos diretórios citados, é importante mencionar dois outros componentes que se encontram na raiz do projeto: o diretório Core e o arquivo injection_container.

O diretório *Core* agrega classes e funcionalidades compartilhadas por componentes de todo o projeto. Nele estarão definidos os erros e exceções disparados durante a execução da aplicação, uma classe para a checagem da conexão à rede e uma classe abstrata definindo um método de chamada para todos os casos de uso do sistema. Na linguagem Dart, é possível criar um método *call* em uma classe, para simplificar sua chamada, sem a necessidade de inclui-lo na declaração. No caso deste projeto, o método *call* de um objeto getMedicamento, por exemplo, pode ser chamado apenas utilizando *getMedicamento()*, no lugar de *getMedicamento.call()*. Assim, a classe abstrata *UseCase*, que todas as classes de caso de uso do projeto estenderão, ajuda a padronizar esse método e a manter o código dentro dos princípios da arquitetura limpa.

Por fim, no arquivo injection_container.dart, é feita toda a injeção de dependências da aplicação, com ajuda da biblioteca *get_it* (Burkhart, 2021). O get_it é um *service locator* (localizador de serviços) desenvolvido para uso com Dart e Flutter. Os objetos são instanciados, na inicialização da aplicação, como *Lazy*S*ingletons* – classes que são instanciadas apenas uma

vez (*singleton*) e inicializadas apenas quando requisitadas pela aplicação (*lazy*) – e podem, então ser acessados em qualquer parte da aplicação sem a necessidade de se criar *widgets* herdados.

Os nomes dos diretórios foram criados em inglês, seguindo o padrão utilizado no mercado e na maior parte dos projetos criados utilizando Flutter e a Arquitetura Limpa (por exemplo: Rešetár, 2019 e Cadic, 2020). Da mesma forma, salvo os nomes das entidades e objetos derivados dela (projetados em português), os nomes de classes, variáveis e métodos do código usam expressões em inglês, para manter consistência com a semântica da linguagem Dart. A estrutura dos diretórios do projeto pode ser visualizada na Figura 5.1.

Figura 5.1 - Estrutura de diretórios do projeto

5.2 O BLoC e a Gerência de Estados

Da mesma forma que temos as classes *Repository* da camada de domínio fazendo a comunicação com as implementações das *Repository* na camada de dados, a comunicação entre a camada de apresentação e os casos de uso da camada de domínio se dá, neste sistema, através dos componentes BLoC. Os casos de uso – que são, em essência, os métodos que representam as operações sofridas pelas entidades do banco de dados – não são chamados diretamente da camada de apresentação, onde são implementados os *widgets* do aplicativo. Em vez disso, os *widgets* disparam eventos, gerenciados pela camada de BLoC que, por sua vez, chama os casos de uso de acordo com o evento, devolvendo estados para a camada de apresentação. A camada de apresentação trata cada mudança de estado gerada pelo BLoC com a finalidade de alterar as telas ou mensagens que aparecem para o usuário.

Para implementar o BLoC e suas funcionalidades, neste projeto, foi utilizada a biblioteca flutter_bloc (ANGELOV, 2019). Essa biblioteca consiste em *widgets* integrados à camada de apresentação, que fazem a gerência de estados.

Para cada entidade, é criado um componente BLoC, que é separado em três partes: *Event*, onde são definidas as classes de eventos, *State*, onde são definidas as classes de estado e *Bloc*, que faz toda a lógica, recebendo os eventos, tratando-os, e devolvendo os estados. O fluxo da camada de BLoC pode ser explicado mais detalhadamente com um exemplo de um caso de uso deste projeto: o cadastro de uma nova crise, principal funcionalidade do aplicativo.

Quando o aplicativo é inicializado, a página inicial é montada utilizando o *widget* *BlocProvider*, que fornece aquele Bloc para as *widgets* "filhas". A instância de *CriseBloc*, instanciada pelo *Service Locator* é então inicializada com todos os casos de uso registrados para a entidade Crise, incluindo o caso *newCrise()*:

```
BlocProvider<CriseBloc>(
  create: (BuildContext context) => di.sl<CriseBloc>(),
)
```

Na página inicial, ao clicar no botão de ação flutuante, o usuário é apresentado com a tela de cadastro de uma crise, informa os dados e clica no botão de cadastro. No momento do clique, após algumas validações dos dados pela camada de apresentação o evento de nova crise, que recebe um objeto *Crise* com os parâmetros informados pelo usuário, é adicionado ao BLoC por meio da seguinte chamada de método:

```
BlocProvider.of<CriseBloc>(context).add(NewCriseEvent(newCrise));
```

Dentro do *stream* de dados do componente *CriseBloc*, ainda na camada de apresentação, é verificado que o evento é o de nova crise, e retornados os estados, em sequência, com os quais a aplicação deverá lidar para alterar o que é exibido ao usuário. Primeiramente, é retornado um *widget* indicando ao usuário que a página está carregando:

```
yield CriseLoading();
```

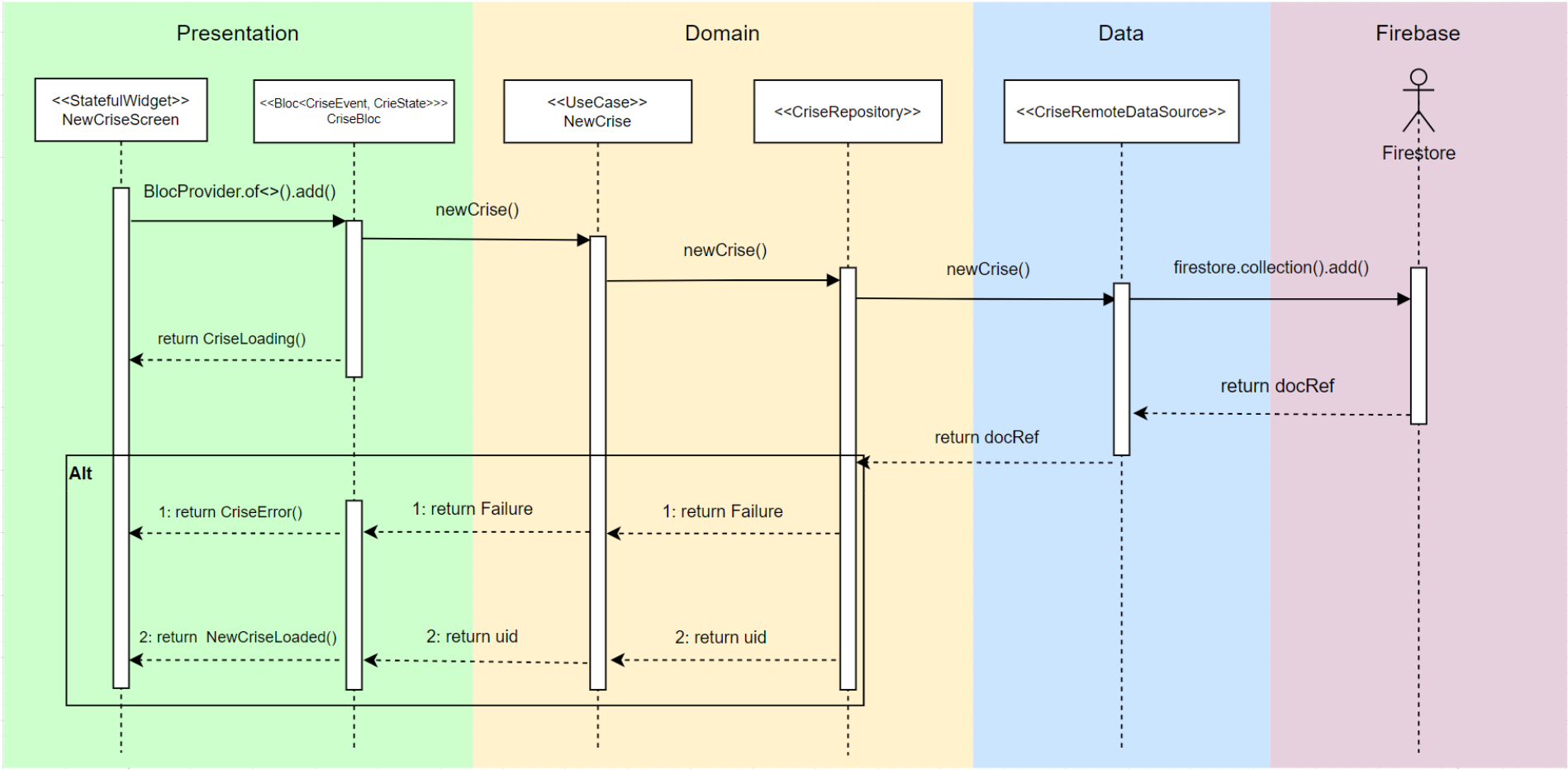
Em seguida, o caso de uso é chamado, de forma assíncrona, e seu retorno é tratado. Em caso de falha, é retornado o estado de erro; em caso de sucesso, é retornado o estado de crise carregada:

```
final failureOrUid = await newCrise(
  NewCriseParams(crise: event.crise),
);

yield failureOrUid.fold(
      (failure) => CriseError(message: _mapFailureToMessage(failure)),
      (uid) => NewCriseLoaded(uid: uid),
);
```

O processo de cadastro de nova crise está esquematizado no diagrama de sequência da Figura 5.2. O diagrama indica os passos do processo com as classes envolvidas (fluxo horizontal) conforme o tempo (fluxo vertical). As classes envolvidas no processo estão dispostas nos quadros brancos, com as classes abstratas estendidas por elas marcadas com as aspas francesas (<< >>). As setas sólidas indicam o fluxo de chamada, com os métodos utilizados pelas classes, enquanto que as setas pontilhadas indicam os valores retornados. As vias coloridas indicam a que camada do sistema a classe pertence.

Figura 5.2 - Diagrama de sequência do cadastro de nova crise



5.3 O banco de dados

Nesta implementação, foi utilizado o banco de dados *NoSQL Firestore* do serviço de *back-end* em nuvem *Google Firebase*. Cada entidade do sistema é armazenada no *Firestore* como uma coleção, e suas instâncias como documentos.

Toda a comunicação com o banco de dados é realizada no diretório *datasources* da camada de dados. Foi utitlizada a biblioteca *cloud_firestore* (CLOUD_FIRESTORE, 2021), que possui classes e métodos específicos para essa comunicação da aplicação com o *Firestore*.

No cadastro de uma nova crise, por exemplo, quando os dados do objeto crise montados pelos parâmetros informados pelo usuário em tela chegam à última camada (após o fluxo detalhado na seção 5.2), é chamado o método *collection().add()* para criar o novo registro:

```
final docRef = firestore.collection("crises").add({
  'diaHoraInicio': crise.diaHoraInicio,
  'diaHoraFim': crise.diaHoraFim,
  'intensidade': crise.intensidade,
  'medicamento': crise.medicamento,
});

return docRef.toString();
```

Cada documento é criado com um número de identificação único alfa-numérico, gerado automaticamente. Esse número é retornado pelo método de criação de registro com o tipo *Document Reference*, que é transformado em uma *String* para ser usado pelas camadas superiores em transações que envolvam apenas um documento específico da coleção (uma transação de exclusão de registro, por exemplo).

Outros exemplos de métodos utilizados pela implementação para comunicação com o banco de dados são: *get()*, que pode retornar todos os documentos da coleção em uma lista, e *delete()*, que exclui um documento específico de uma coleção.

5.4 O Aplicativo

Nesta seção, serão apresentadas as telas desenvolvidas para o aplicativo, com suas funcionalidades e especificidades em sua construção. O código-fonte do aplicativo pode ser acessado livremente, para mais detalhes, em: https://github.com/rzapin/enxaqueca-tcc.

5.4.1 A tela inicial

Ao abrir o aplicativo, o usuário é apresentado à tela inicial (Figura 5.3), que contém alguns *widgets* em formatos de caixas, com informações importantes e que servem como botões para acesso às outras telas dos aplicativos. Um calendário simples apresenta uma visualização do dia e mês atuais, e dá acesso ao calendário que apresenta os dias de crises. Abaixo, o usuário tem uma visão geral de quantos dias está sem dores de cabeça, quais foram as últimas crises e os últimos medicamentos administrados, e alguns dados estatísticos de suas crises. O botão de ação flutuante com o símbolo de adição é utilizado para cadastrar uma nova crise.

Figura 5.3 - Tela inicial do Diário de Enxaqueca

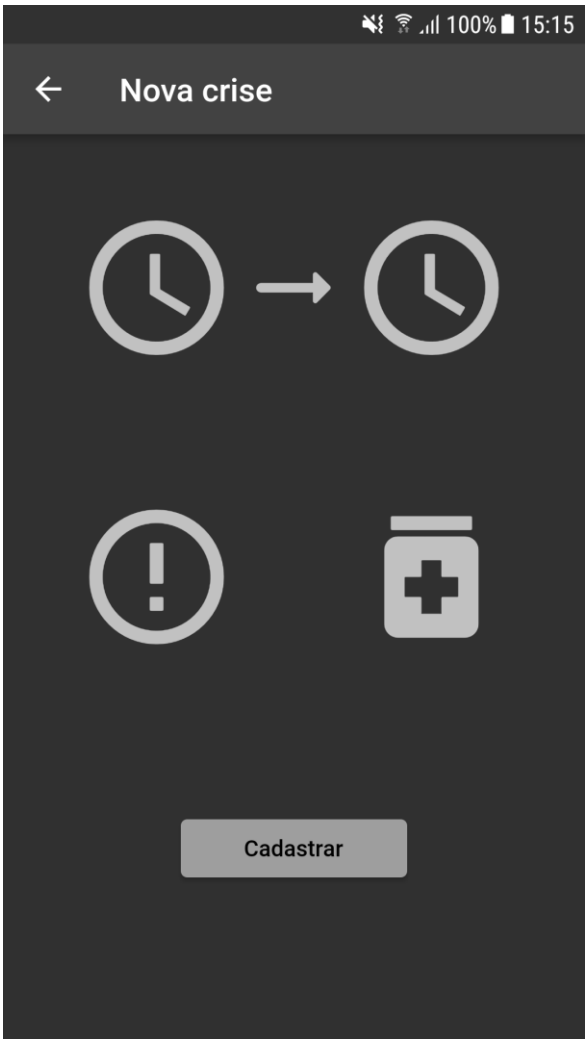
Cada caixa é construída com um *BlocBuilder*, que é a classe da biblioteca *flutter_bloc* que constrói *widgets* a partir de estados retornados. A caixa de últimas crises, por exemplo,

recebe o estado retornado do evento *getAllCrises()* para utilizar os dados das crises na visualização da lista.

5.4.2 A tela de cadastro de crises

Na tela de cadastro (Figura 5.4), optou-se pela utilização somente de ícones, para manter a simplicidade do cadastro e não atrapalhar o usuário. Os botões são: a) dia e hora de início, inicializado com o dia e hora do momento do cadastro; b) dia e hora de fim, inicializado em 6 horas após o dia e hora de início; c) intensidade; d) medicamento, que carrega a lista de medicamentos cadastrados para seleção do usuário.

Figura 5.4 - Tela de cadastro de crise



Quando o usuário clica nos ícones de relógio, um menu se abre na parte inferior da tela para a seleção do dia e horário de início (Figura 5.5) ou fim da crise. O ícone com o ponto de exclamação seleciona a intensidade da crise, em uma escala de 1 a 10. O ícone com o medicamento abre a lista de medicamentos para que seja selecionado o medicamento administrado, caso o usuário tenha utilizado medicamento para alívio da dor (Figura 5.6). A crise é cadastrada com sucesso, após validações das datas (datas anteriores à data e hora atuais e data de fim posterior a data de início) quando é pressionado o botão de cadastro.

Figura 5.5 - Menu para seleção de dia e hora de início

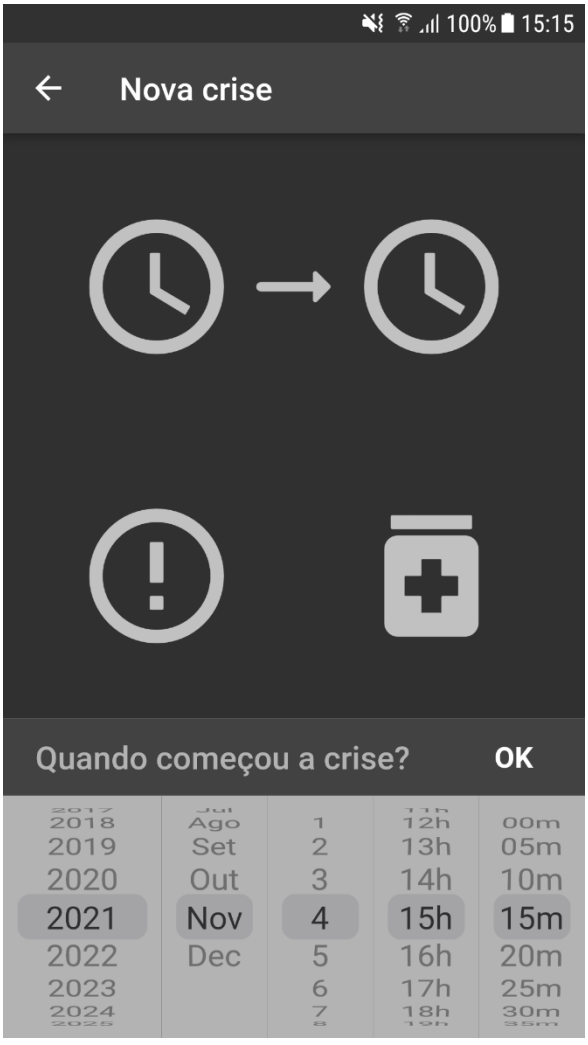
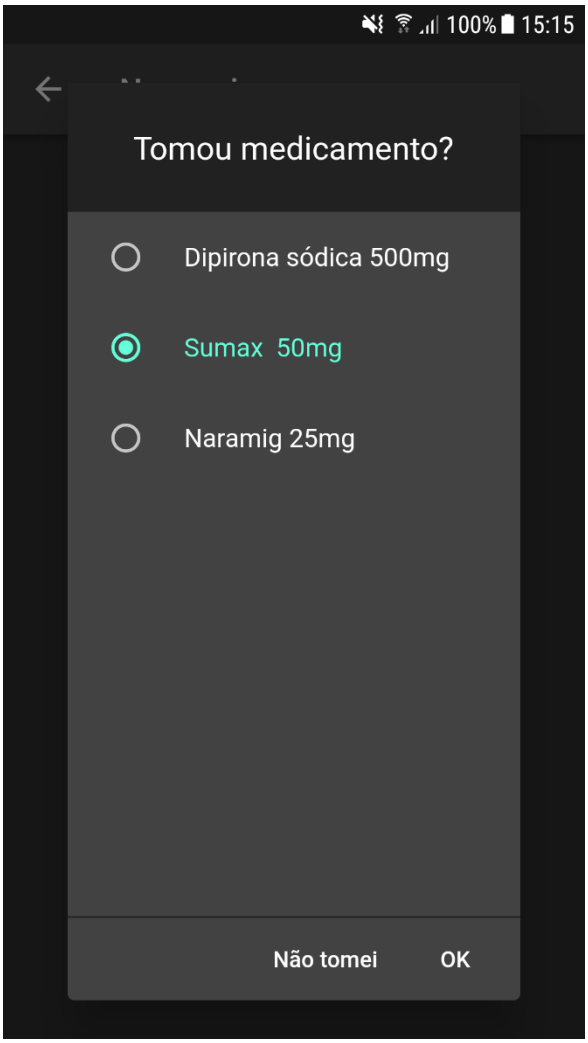


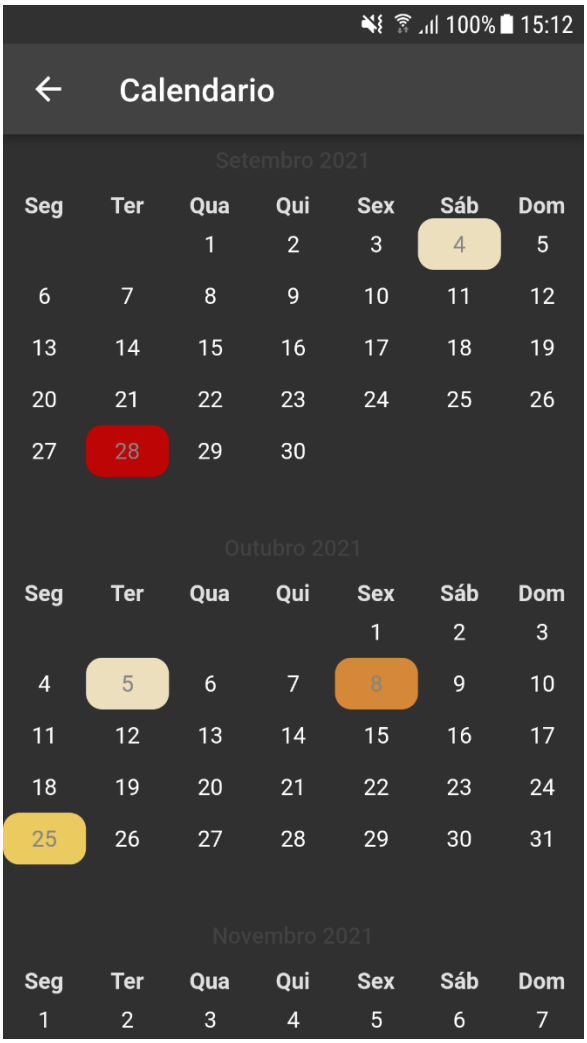
Figura 5.6 - Menu para a seleção de medicamento



5.4.3 O Calendário

Na tela de calendário (Figura 5.7), acessada quando o usuário clica no botão de calendário da tela inicial, é apresentado ao usuário um calendário de rolagem vertical, com os dias de crise marcados com cores de acordo com as intensidades. Um arranjo de cores padrão já é definido no banco de dados: amarelo creme para as intensidades 1 e 2 (muito leve), amarelo alaranjado para as intensidades 3 e 4 (leve), laranja para as intensidades 5 e 6 (moderada), vermelho rosado para as intensidades 7 e 8 (forte) e vermelho para as intensidades 9 e 10 (muito forte). Como trabalho futuro, poderá ser implementada a funcionalidade de atribuição de cores para cada intensidade, pelo usuário.

Figura 5.7 - Tela de calendário com os dias de crise marcados com cores



Para a construção do calendário, foi utilizada a biblioteca *scrollable_clean_calendar* (FIUZA et al., 2021). A biblioteca foi alterada localmente para adicionar métodos e funcionalidades apropriadas à implementação deste projeto. Entre outras modificações, foram adicionados métodos para selecionar a cor de fundo que uma data deve ter e definir regras para a aplicação da cor de fundo.

Quando uma data é clicada, é apresentada uma caixa de diálogo com as informações da crise, conforme Figura 5.8.

Figura 5.8 - Caixa de diálogo com informações da crise

5.4.4 As listas de crises e medicamentos

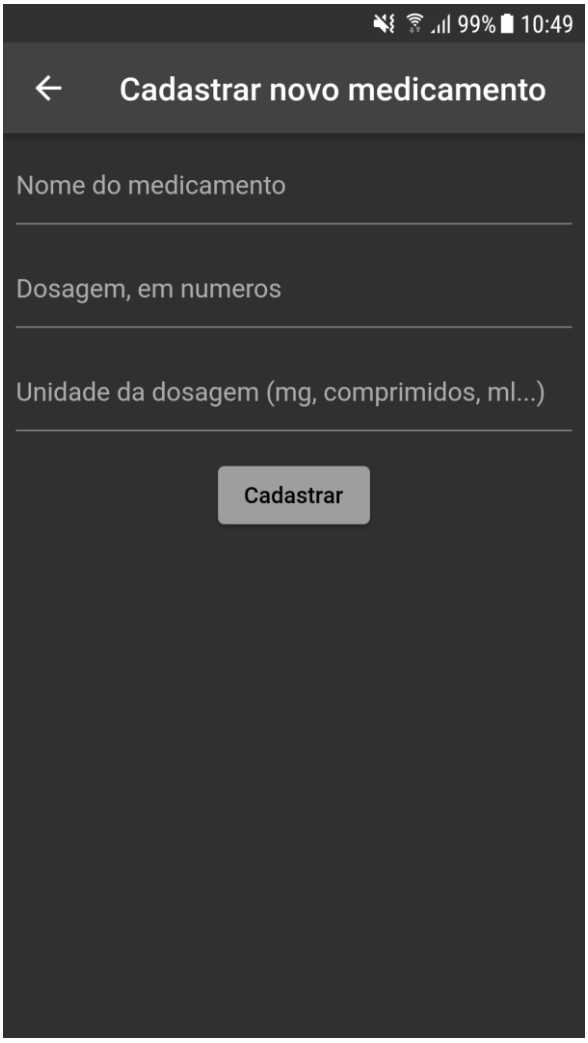
A lista de crises é aberta quando o usuário clica na caixa de últimas crises. É apresentada em uma lista de *tiles* (itens) que podem ser expandidos quando clicados, mostrando mais informações sobre as crises (Figura 5.9). Para excluir uma crise, os *tiles* podem ser arrastados para o lado; a crise é excluída após confirmação do usuário. Na barra superior, o usuário poderá adicionar uma nova crise clicando no botão com o símbolo de adição.

Figura 5.9 Tela com a lista de crises cadastradas

A tela de lista de medicamentos, acessada quando o usuário clica na caixa de medicamentos, da tela inicial, é construída de forma similar à lista de crises, com itens estáticos, não expansíveis. Para excluir um medicamento, o usuário pode arrastar o item para o lado. Para adicionar um novo medicamento, assim como na tela de crises, o usuário poderá clicar no botão de adição, na barra superior.

A tela de cadastro de medicamentos (Figura 5.10) é composta por um formulário de texto simples, contendo os campos necessários para cadastro. Ao clicar no botão Cadastrar, o usuário é levado de volta à tela com a lista de medicamentos cadastrados.

Figura 5.10 - Tela de cadastro de medicamento



6. CONCLUSÃO

Este trabalho teve como objetivo projetar e desenvolver uma solução para dispositivos móveis que permita ao usuário cadastrar dias em que teve crises de enxaqueca, com informações como horas de início e fim, intensidade, medicamento administrado, sintomas e fatores causadores relacionados. Associado a isso, ao longo do desenvolvimento do aplicativo pretendeu-se utilizar tecnologias relativamente novas no mercado para a implementação, com o framework Flutter no desenvolvimento, o BLoC para a gerência de estados do aplicativo e a Arquitetura Limpa como padrão de arquitetura de software. A utilização dessas tecnologias se provou o maior desafio na implementação da solução planejada, devido ao tempo demandado pela curva de aprendizagem.

Das 29 histórias de usuário projetadas para a solução, conforme detalhado na seção 4.2, foram implementadas 9. O desenvolvimento desta solução ainda é um trabalho em andamento, com a possibilidade de implementação de diversas melhorias futuras. Entre elas, podemos destacar a adição de:

1. **Entidades de gatilho e sintomas relacionados no cadastro da crise** – Essas entidades e seus casos de uso básicos já foram implementados nas camadas inferiores de domínio e dados, sendo necessário apenas sua adição na interface de usuário. Essa facilidade é um exemplo de como a modularização característica da Arquitetura Limpa pode ajudar na manutenção do sistema.
2. **Funcionalidades para personalizar cores –** Na implementação, foi dada prioridade para as funcionalidades básicas do cadastro de crises e montagem e visualização do calendário. Para a visualização do calendário, foram implementadas cores padrão para a demonstração das crises de acordo com as intensidades.
3. **Operações de atualização de crises e medicamentos já cadastrados** – Assim como com as entidades Gatilho e Sintoma, e seus casos de uso básicos, já implementados nas camadas inferiores, os casos de uso para *update* das crises e medicamentos já estão implementados, faltando sua adição na interface de usuário. Como a atualização de uma crise ou medicamento pode ser feita excluindo-os e cadastrando novamente, a implementação das operações de update não são necessárias para se ter um produto minimamente viável (MVP).

4. **Consulta aos dados cadastrados quando sem conexão à Internet** – Pelo fato de os dados do aplicativo estarem armazenados em nuvem, no momento, as operações sobre o banco de dados, incluindo consulta, no momento, acontecem apenas se o usuário está conectado à Internet. Isso ainda limita o uso do aplicativo.
5. **Funcionalidade de autenticação de usuário –** Para produzir um MVP, também foi deixada a funcionalidade de autenticação do usuário como uma implementação futura.

Apesar da curva de aprendizagem mencionada, durante o processo de desenvolvimento da solução, a utilização do framework Flutter trouxe as vantagens esperadas, especialmente na implementação e manutenção da camada de apresentação. A modularização concedida pela linguagem facilita a troca e ajuste dos *widgets*, uma vez que os componentes iniciais já estão implementados. De forma análoga, do ponto de vista da estrutura de *back-end* do sistema, o uso da Arquitetura Limpa permitiu, por exemplo, que transações com componentes externos, implementação de novos casos de uso ou o ajuste de casos de uso já implementados fossem realizadas com agilidade.

Com o projeto executado neste trabalho de conclusão de curso, foi possível produzir um MVP que alcançou o objetivo principal proposto. Um usuário que sofre e faz tratamento para enxaqueca, público-alvo da solução, pode registrar suas crises, administrá-las e visualizá-las em um calendário, marcadas com cores. Como já pontuado, as tecnologias e padrões utilizados para o desenvolvimento da solução, implementados com sucesso, permitem a modularização e fácil manutenção do código. Devido a isso, o software desenvolvido neste trabalho sofrerá um processo de melhoria contínua.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLENA, M.; CUZZONI, M. G.; TASSORELLI, C.; NAPPI, G.; ANTONACI, F. An Electronic Diary on a Palm Device for Headache Monitoring: a Preliminary Experience. **The Journal of Headache and Pain**, v. 13, n. 7, p. 537–541, 2012. Disponível em: https://doi.org/10.1007/s10194-012-0473-2. Acesso em: 23, Ago, 2021.

AMADEO, R. Google's Dart Language on Android Aims for Java-free, 120 FPS apps. **Ars Technica**, 2015. Disponível em: https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/ . Acesso em: 28, Ago, 2021.

ANGELOV, F. flutter_bloc. Versão 3.0.0. **Pub.dev**, 2019. [Pacote de códigos-fonte]. Disponível em: https://pub.dev/packages/flutter_bloc. Acesso em: 06, Set, 2021.

BORKUM, J. M. Migraine triggers and oxidative stress: A narrative review and synthesis. **Headache**, v. 56, p. 12-35, 2016.

BURKHART, T. get_it. Versão 7.2.0. **Pub.dev**, 2021. [Pacote de códigos-fonte]. Disponível em: https://pub.dev/packages/get_it. Acesso em: 03, nov, 2021.

CADIC, E. A Flutter BLoC + Clean Architecture journey to release the #1st Idean Flutter app. **Ideas by Idean**. Disponível em: https://medium.com/ideas-by-idean/a-flutter-bloc-clean-architecture-journey-to-release-the-1st-idean-flutter-app-db218021a804. Acesso em: 11, nov, 2021.

CLOUD_FIRESTORE. Versão 3.1.0. **Pub.dev**, 2021. [Pacote de códigos-fonte]. Disponível em: https://pub.dev/packages/cloud_firestore. Acesso em: 03, nov, 2021.

DART Programming Language. **Dart.dev**, [s.d.]. Disponível em: https://dart.dev/ Acesso em: 06, Set, 2021.

DARTZ. Versão 0.8.9. **Pub.dev**, 2019. [Pacote de códigos-fonte]. Disponível em: https://pub.dev/packages/dartz. Acesso em: 03, nov, 2021.

FIUZA, F. scrollable_clean_calendar. Versão 0.5.2. **Pub.dev**, 2021. [Pacote de códigos-fonte]. Disponível em: https://pub.dev/packages/scrollable_clean_calendar. Acesso em: 06, Set, 2021.

FLUTTER architectural overview. **Flutter.dev**, [s.d.]. Disponível em: https://flutter.dev/docs/resources/architectural-overview. Acesso em: 28, Ago, 2021.

GIFFIN, N. J.; LIPTON, R. B.; SILBERSTEIN, S. D.; OLESEN, J.; GOADSBY, P. J. The migraine postdrome: An electronic diary study. **Neurology**, v. 87, p. 309-313, 2016.

GILMORE, B.; MICHAEL, M. Treatment of acute migraine headache. **American Family Physician**, v. 83, n. 3, p. 271-280, 2011.

HAN, M.; LEE, E. Effectiveness of mobile health application use to improve health behavior changes: A systematic review of randomized controlled trial. **Healthcare Informatics Research**, v. 24, n. 3, p. 207-226, 2016.

HUNDERT, A. S.; HUGUET, A.; MCGRATH, P. J.; STINSON, J. N.; WHEATON, M. Commercially available mobile phone headache diary apps: a systematic review. **JMIR mHealth and uHealth**, v. 2, n. 3. 2014. Disponível em: https://doi.org/10.2196/mhealth.3452. Acesso em: 23, Ago, 2021.

INTERNATIONAL HEADACHE SOCIETY. The international classification of headache disorders, 3rd edition. **Cephalalgia**, v. 38, n. 1, p. 1-211, 2018.

KVISVIK, E. V.; STOVNER, L. J.; HELDE, G.; BOVIM, G.; LINDE, M. Headache and migraine during pregnancy and puerperium: the MIGRA-study. **The Journal of Headache and Pain**, v. 12, p. 443-451, 2011.

MARTIN, R. C. The Clean Architecture. **The Clean Code Blog**, 2012. Disponível em: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. Acesso em: 29, Ago, 2021.

MARTIN, R. C. **Arquitetura Limpa: O Guia do Artesão para Estrutura e Design de Software**. 1ª Edição. Rio de Janeiro: Alta Books, 2019.

MOURA, J. A. Sua Arquitetura Está Limpa? (Clean Architecture no Flutter). **Flutterando**. 2020. Disponível em: https://blog.flutterando.com.br/sua-arquitetura-est%C3%A1-limpa-clean-architecture-no-flutter-458c68fad120. Acesso em: 23, Ago, 2021.

NATIONAL HEADACHE FOUNDATION. **Headache diary: Keeping a diary can help your doctor help you**. Disponível em: https://headaches.org/resources/headache-diary-keeping-a-diary-can-help-your-doctor-help-you/. Acesso em: 18, mar, 2021.

NHS. National Health Service. **Overview**: Migraine. Disponível em: https://www.nhs.uk/conditions/migraine/#:~:text=Treating%20migraines,brain%20that%20may%20cause%20migraines. Acesso em: 18, mar, 2021.

OMS. Organização Mundial da Saúde. **International statistical classification of diseases and related health problems 10th revision**. 2019. Disponível em: https://icd.who.int/browse10/2019/en. Acesso em: 18, mar, 2021.

OSKOUI, M.; PRINGSHEIM, T.; BILLINGHURST, L. et al. Practice guideline update summary: Pharmacologic treatment for pediatric migraine prevention. **Neurology**. v. 93, p. 500-509, 2019.

PIANE, M.; LULLI, P.; FARINELLI, I. et al. Genetics of migraine and pharmacogenomics: Some considerations. **The Journal of Headache and Pain**, v. 7, p. 334-339, 2007.

REŠETÁR, M. Flutter TDD Clean Architecture Course 1 - Explanation & Project Structure. **Reso Coder**. 2019. Disponível em: https://resocoder.com/2019/08/27/flutter-tdd-clean-architecture-course-1-explanation-project-structure/ . Acesso em: 23, Ago, 2021.

SURI, S. Architect your Flutter project using BLOC pattern. **Code Chai**, 2018. Disponível em: https://medium.com/codechai/architecting-your-flutter-project-bd04e144a8f1. Acesso em: 29, Ago, 2021.

VO, P.; PARIS N.; BILITOU, A. Burden of Migraine in Europe Using Self-Reported Digital Diary Data from the Migraine Buddy Application. **Neurology and Therapy**, v. 7, p. 321–332, 2012.

YILDIRIM, U. Using Clean Architecture in Flutter. **Codeburst**. 2020**. Disponível em:** https://codeburst.io/using-clean-architecture-in-flutter-d0437d0c7f87. Acesso em: 23, Ago, 2021.

ZEBENHOLZER, K.; RUDEL, E.; BRANNATH, W. Migraine and weather: A prospective diary-based analysis. **Cephalalgia.** v. 31, p. 391-400.