"BoraMarcar" - Um Aplicativo para Marcar Todo Tipo de Evento, de Reuniões a Festas

Relatório de Projeto Final do Curso de Ciência da Computação

Gustavo Barros Marchesan
1521500

Profa. Orientadora: Simone Diniz Junqueira Barbosa

Agradecimentos

À professora Simone Diniz Junqueira, pela disponibilidade e paciência durante esse ano.

Ao Hugo Martins Tonette, Marcello Nunes Bernades, Vinicius Cavalcante, Hugo Ribeiro e outros amigos que testaram e contribuíram com esse aplicativo e que vão me perdoar por não citá-los por nome.

À minha mãe, vó, tia e todos os membros da minha família que me ajudaram e possibilitaram a jornada que foi a faculdade.

E aos professores, alunos ou curiosos que estão dedicando tempo para ler esse relatório. Espero que ele seja iluminador.

Resumo

Marchesan, Gustavo. Barbosa, Simone. **"Bora Marcar" - Um Aplicativo para Marcar Todo Tipo de Evento, de Reuniões a Festas**. Rio de Janeiro, 2021. 44p. Relatório de Projeto Final II do CTC – Centro Técnico Científico – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Neste projeto, visamos criar um aplicativo para plataforma Android que facilitaria a organização de eventos e ajudaria na organização da vida profissional e pessoal do usuário. O aplicativo armazena o horário fornecido pelo usuário e, ao criar um evento, define o melhor dia e hora para o evento considerando os horários dos usuários convidados.

Palavras-chave:

Aplicativo, Android, Agenda, Flutter, Firebase

Sumário

Lista de Figuras

1. Introdução

1.1. Propósito

O propósito deste aplicativo é facilitar marcar eventos e reuniões. Usuários do aplicativo poderão criar eventos e convidar outros usuários, os usuários têm seus horários cadastrados no aplicativo e ao aceitarem o convite o aplicativo irá determinar o melhor dia e hora para o evento baseado nos horários fornecidos pelos convidados para o evento.

Todo usuário tem o cadastro de um horário de trabalho e um horário livre, e o criador do evento determina qual horário usar. Essa demarcação do horário ajuda a controlar a carga horária de trabalho de um usuário, evitando que ele trabalhe em excesso.

1.2. Motivação

A inspiração do “BoraMarcar” veio de uma situação muito familiar para todos; tentar marcar algum evento acaba se tornando um malabarismo de horários conflitantes. A dificuldade em encontrar uma boa data para um evento aumenta progressivamente com o número de convidados, afinal todos têm horários de trabalho e de lazer diferentes, e o “BoraMarcar” foi primeiro envisionado como uma solução para essas situações. Até a quarentena de 2020.

Durante a quarentena muitos se viram obrigados a trabalhar e estudar de casa. Esse fim da diferenciação entre o ambiente de trabalho e o ambiente de lazer afetou também a diferenciação dos horários de trabalho e lazer. Essa situação inspirou a outra funcionalidade do aplicativo, ajudar a manter um horário de trabalho mais saudável e menos intrusivo na vida do trabalhador.

1.3. Público Alvo

O "Bora Marcar" é destinado a qualquer pessoa que deseja ter um controle melhor do seu horário de trabalho, atingir um melhor equilíbrio entre trabalho e lazer ou apenas queira manter o seu horário mais organizado.

O aplicativo pode ser especialmente útil para pessoas que trabalham remotamente; e possuem dificuldade em limitar o seu horário de trabalho, assim como pessoas que trabalham como *freelancers* ou com empregos com horários mais erráticos.

Apesar de a proposta ser voltada para pessoas no mercado de trabalho, nada impede que qualquer pessoa, independente da situação empregatícia, possa tirar proveito do "Bora Marcar". Um dos focos do aplicativo é ter um estilo mais casual do que outros aplicativos de agendamento

1.4. Comparações

Existem outros aplicativos focados em marcar eventos como Calendly, Boomerang Calendar e o *When2Meet*, mas o “BoraMarcar” se diferencia em alguns aspectos;

- Assim como o When2Meet, o “BoraMarcar” determina o melhor horário para um evento, mas como o “BoraMarcar” armazena o horário do usuário, o planejamento de novos eventos é mais simplificado.
- O “BoraMarcar” exige que o usuário se cadastre para usar o aplicativo, o que pode ser considerado um ponto negativo se comparado a maioria dos serviços similares (normalmente o cadastro é opcional, só exige o e-mail), mas ele compensa esse incômodo facilitando o planejamento de eventos com convidados recorrentes, já que como o horário do usuário já está armazenado.
- A funcionalidade de Grupos também permite que eventos com múltiplos convidados sejam criados com mais facilidade.
- Ele tem uma proposta mais casual. A maioria dos aplicativos é voltada apenas para o ambiente de trabalho, como ele poderá ser usado para todo tipo de evento, de consultas a festas.
- Como ele é um aplicativo Android ele faz uso das funcionalidades de Smartphones, como notificações para notificar os usuários de eventos ou atualizações, e câmeras para customizar grupos e eventos.

1.5. Escopo do Produto

Como especificado antes, o objetivo do “BoraMarcar” é otimizar e facilitar a organização de eventos. O sistema será um aplicativo mobile para Android, o que facilita a notificação do usuário, a utilização frequente e a facilidade de acesso dos usuários. O propósito deste aplicativo é facilitar o controle do horário de trabalho e proporcionar aos usuários um meio de ter um equilíbrio melhor entre trabalho e lazer.

2. Descrição Geral

2.1. Perspectiva do Produto

No momento existem vários aplicativos de agendamento; como Boomerang Calendar, Calendly, dentre outros, mas o diferencial do “BoraMarcar” em relação a outros é o fato dele poder determinar o melhor horário para o evento e ele terá uma proposta mais casual, para mostrar que ele pode ser usado para diferentes tipos de evento.

Além disso, por ser um aplicativo ele sempre mantém o usuário atualizado sobre os eventos que ele faz parte, de uma forma mais eficaz do que outros programas.

2.2. Classes e Definições dos Usuários

Os usuários do “BoraMarcar” devem ter acesso a eventos e grupos que criaram e que estão convidados ou são membros. Todo usuário precisa ser autenticado para utilizar o aplicativo. Os usuários podem alterar as suas informações salvas; a sua foto de perfil, nome, aniversário e horários de trabalho e lazer.

Um usuário padrão pode;

- Criar um novo evento.
- Criar um novo grupo.

- Aceitar convites para eventos ou grupos.
- Alterar os seus horários.

Usuários administradores são os criadores dos eventos e grupos, então, além das funções de usuários-padrão, eles podem;

- Editar as informações dos eventos e grupos.
- Convidar ou desconvidar outros usuários para um evento ou grupo.
- Deletar eventos ou grupos.

2.3. Ambiente Operacional

O aplicativo foi feito e otimizado para ser utilizado em *smartphones* com Android.

Uma versão para iOS foi planejada na primeira proposta deste aplicativo, mas por questões de recursos e tempo o escopo foi reduzido para apenas uma versão Android. Ainda assim, uma versão tanto para iOS quanto Web são relativamente simples de serem implementadas dado o que já foi implementado para o aplicativo, mas exigiria mais tempo do que o projeto dispõe.

Para que o aplicativo possa operar corretamente, o dispositivo Android deve apresentar uma versão igual ou superior ao Android 6.0 (api de nível 23). O aplicativo tem tamanho 152MB, e armazena dados do usuário e um cache, então é recomendado para o uso do aplicativo um espaço livre mínimo de 210 MB.

2.4. Cenário de Uso

Faremos um exemplo ilustrado do uso do “BoraMarcar” onde dois usuários, chamaremos de Usuário A e Usuário B, irão marcar uma reunião.

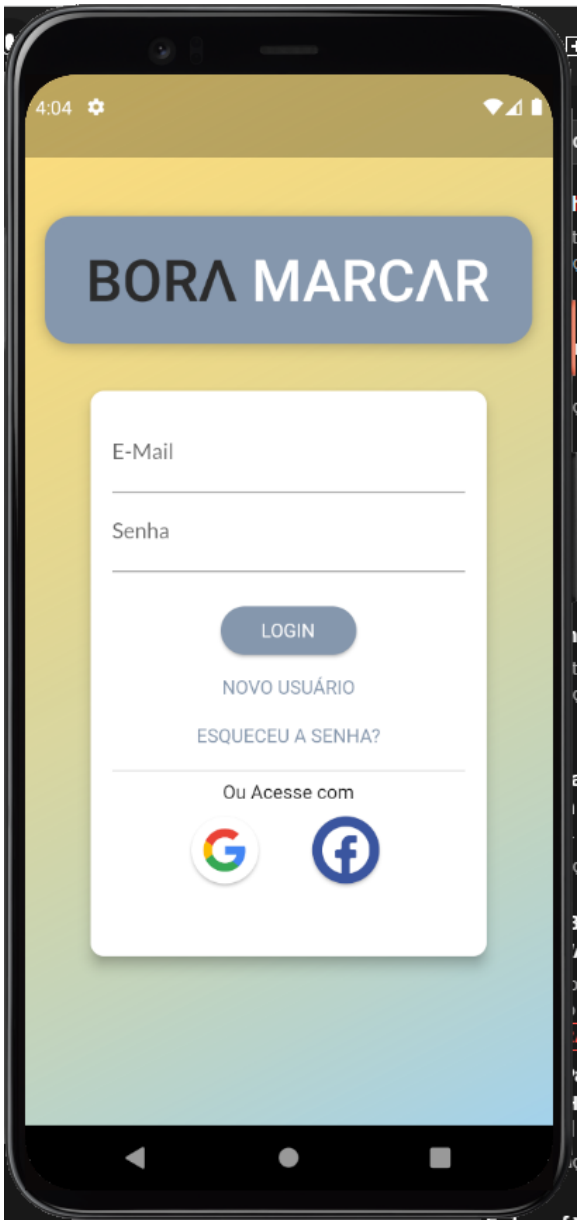


Figura 1: Tela de autenticação do “BoraMarcar”

Primeiramente os dois usuários devem estar autenticados.

Usuário A se autentica usando a sua conta Google.

Com o acesso aceito, ele é direcionado para a página inicial.

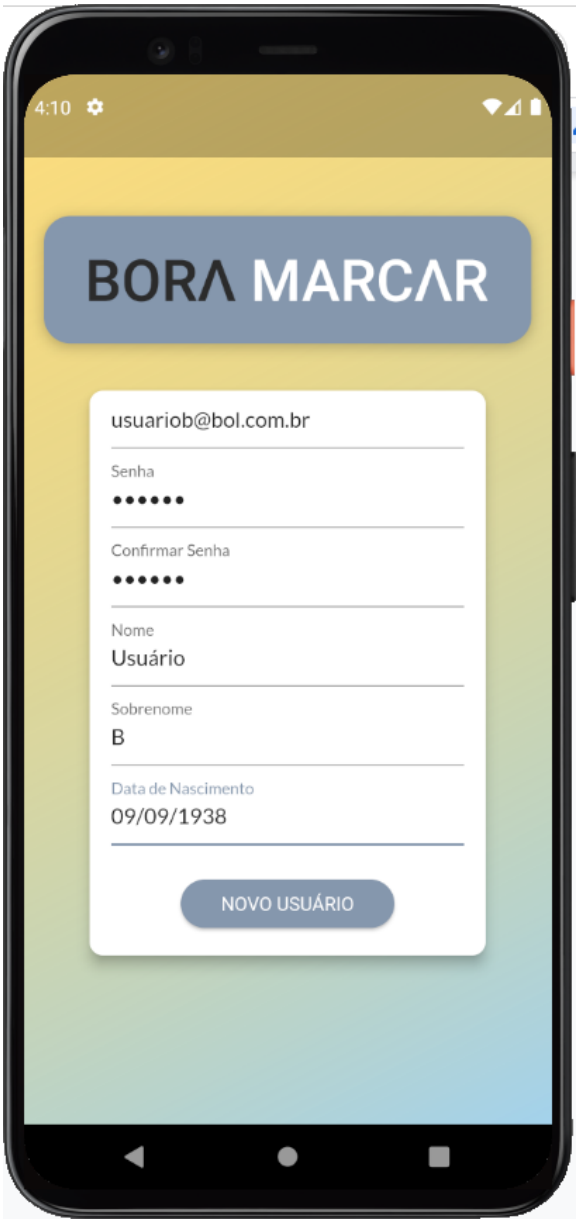


Figura 2: Usuário B se cadastrando pelo formulário de cadastro

O Usuário B, não querendo compartilhar sua conta, prefere se cadastrar diretamente no aplicativo.

O Usuário B insere a suas informações e confirma e é direcionado para a tela inicial.

Figura 3: Tela inicial do “BoraMarcar”

Ao entrar no aplicativo o Usuário A, antes de criar o novo evento, lê o lembrete sobre o seu horário, e decide verificar o seu horário clicando na barra de navegação, a segunda opção à direita.

O Usuário B também é recebido pela mesma imagem, e decide verificar o seu horário.

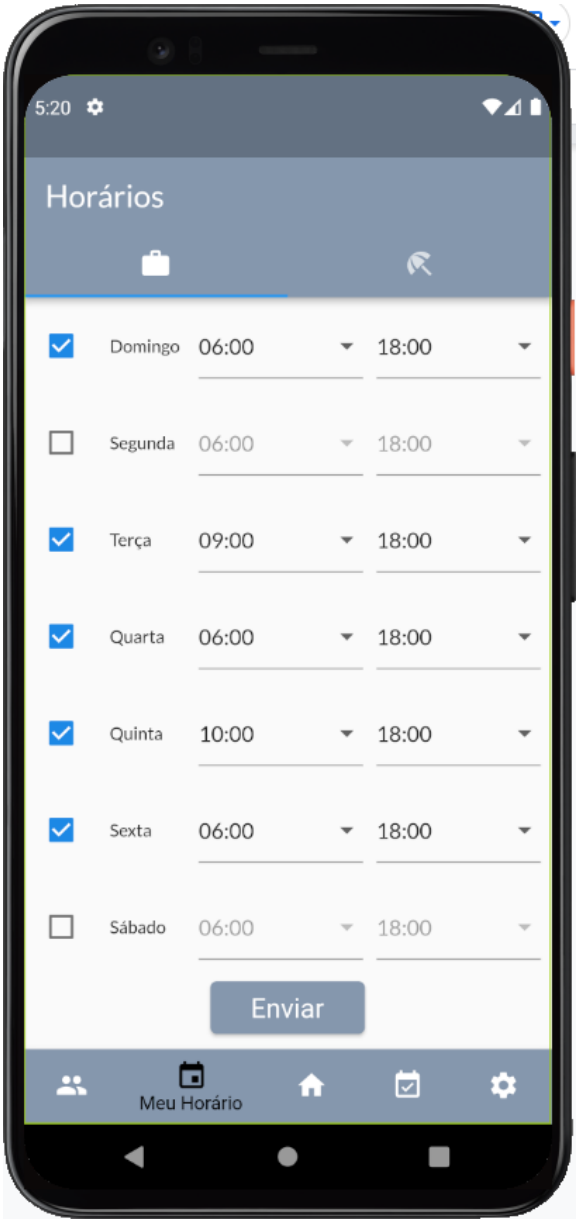
<table>
<tr>
<td> 

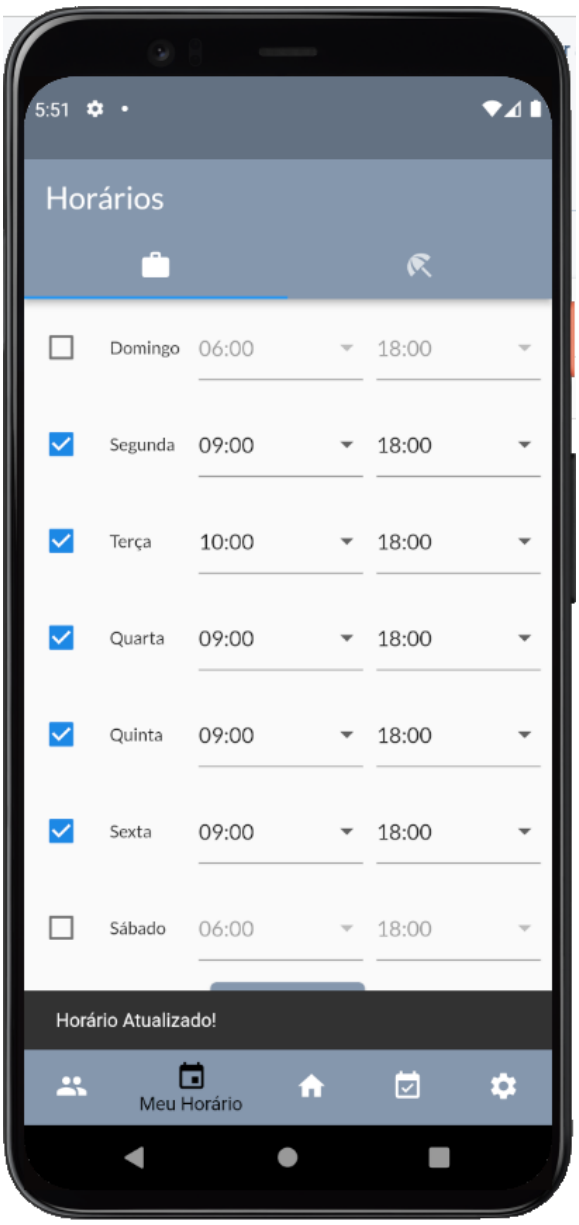
Figura 4: Tela de Cadastro do horário do usuário A.</td>
<td> 

Figura 5: Tela de Cadastro do horário do usuário B.</td>
</tr>
<tr>
<td colspan="2">Os dois usuários cadastram os seus horários;</td>
</tr>
<tr>
<td>Usuário A:
Domingo: 06:00 - 18:00
Segunda: Bloqueado
Terça: 09:00 - 18:00
Quarta: 06:00 - 18:00
Quinta: 10:00 - 18:00
Sexta: 06:00 - 18:00
Sábado: Bloqueado</td>
<td>Usuário B:
Domingo: Bloqueado
Segunda: 09:00 - 18:00
Terça: 10:00 - 18:00
Quarta: 09:00 - 18:00
Quinta: 09:00 - 18:00
Sexta: 09:00 - 18:00
Sábado: Bloqueado</td>
</tr>
</table>

Figura 6: Janela de Convite de Usuários.

Figura 7: Formulário de Novo Evento preenchido pelo Usuário A.

O Usuário A precisa se reunir com o Usuário B na semana seguinte, então ele coloca o intervalo da data entre segunda e sexta da semana seguinte.

Ao clicar em “Adicionar Convidados” ele é instruído a inserir o email do usuário que ele deseja adicionar. Ao inserir o email do Usuário B, ele encontra o Usuário B e ao clicar no seu nome adiciona ele ao evento. Um chip indicando o email do usuário aparece no formulário.

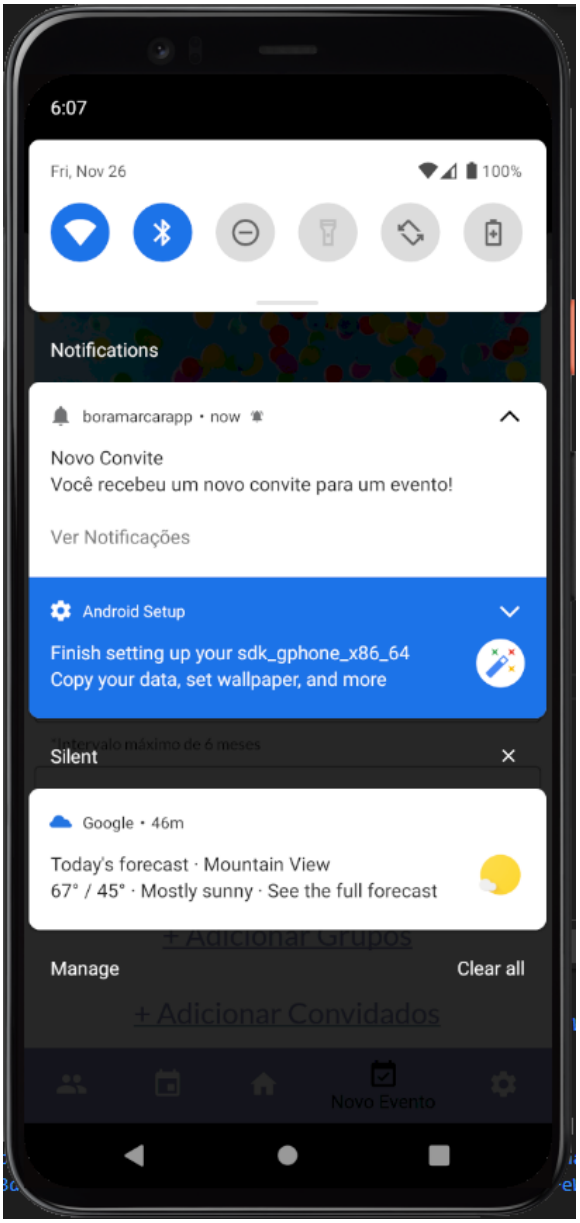


Figura 8: Notificações no Android do Usuário B.

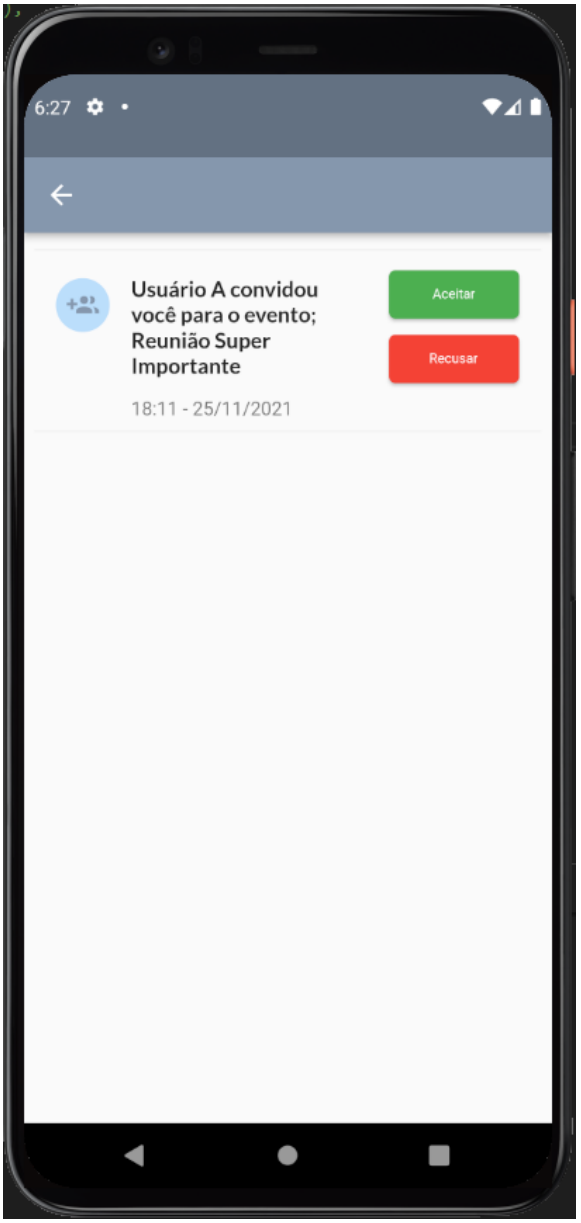


Figura 9: Página de Notificações do Usuário B.

Quando o Usuário A criou o evento, o Usuário B recebeu uma notificação Push avisando do convite. Ao clicar ele foi direcionado para as suas notificações onde ele viu o convite. Ao clicar em aceitar, ele foi direcionado para a página do evento.

Figura 10: Página do Evento Reunião Super Importante.

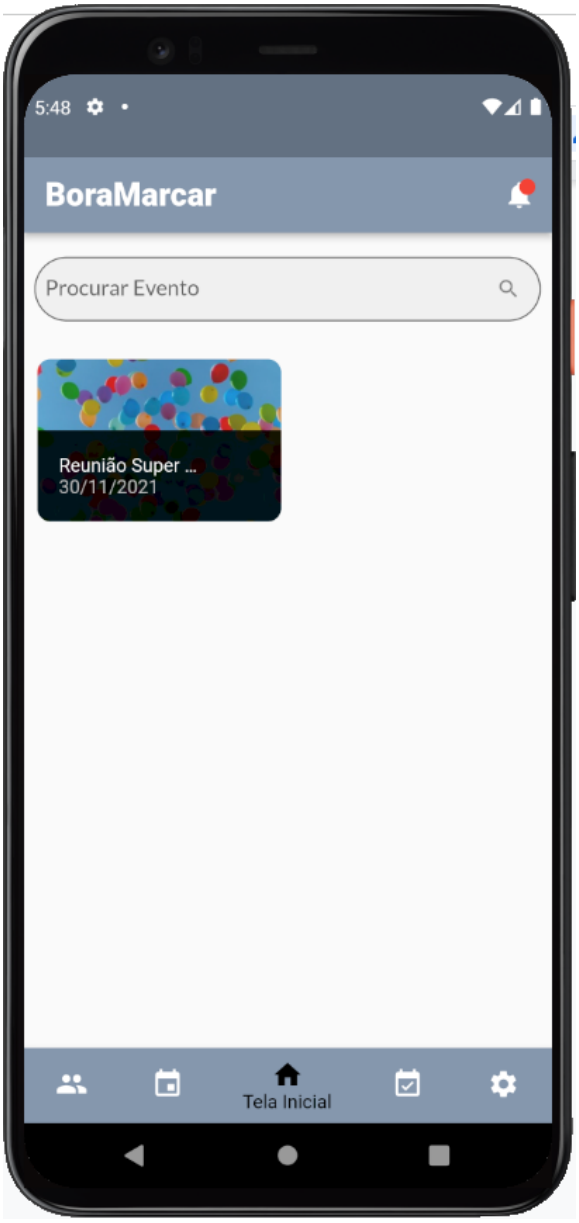


Figura 11: Tela inicial, agora com o novo evento.

Quando o Usuário B aceitou o convite o evento “Reunião Super Importante” atualizou o seu horário para acomodar os horários de todos os usuários.

Ao retornar para a página inicial, agora os dois usuários verão o evento da reunião.

3. Desenvolvimento

3.1. Ferramentas

Nesta seção do relatório, especificamos quais ferramentas foram utilizadas para desenvolver o "Bora Marcar", o seu uso e o motivo da escolha.

Flutter (Versão 2.5.3)

Flutter é um framework para o desenvolvimento de aplicativos mobile para Android e iOS. Desenvolvido pelo Google e lançado em maio de 2017, com uma versão estável em dezembro de 2019, ele possibilita a criação de aplicativos compilados nativamente em múltiplas plataformas, estas sendo (até o momento) Android, iOS, Windows, Mac, Linux, Google Fuchsia (um sistema operacional atualmente sendo desenvolvido pelo Google) e Web [1].

A engine do Flutter, escrito principalmente em C++, fornece suporte de renderização de baixo nível usando a biblioteca de gráficos Skia do Google. Além disso, ele faz interface com SDKs específicos da plataforma, como os fornecidos pelo Android e iOS.

Ao criar um aplicativo com o Flutter, seu código é compilado para a linguagem base do dispositivo, ou seja, as aplicações são realmente nativas e por isso conseguem acessar recursos do dispositivo sem a “ajuda” de terceiros e com um desempenho maior. A figura 12 a seguir demonstra a comparação do flutter com React, um outro framework para desenvolvimento mobile;

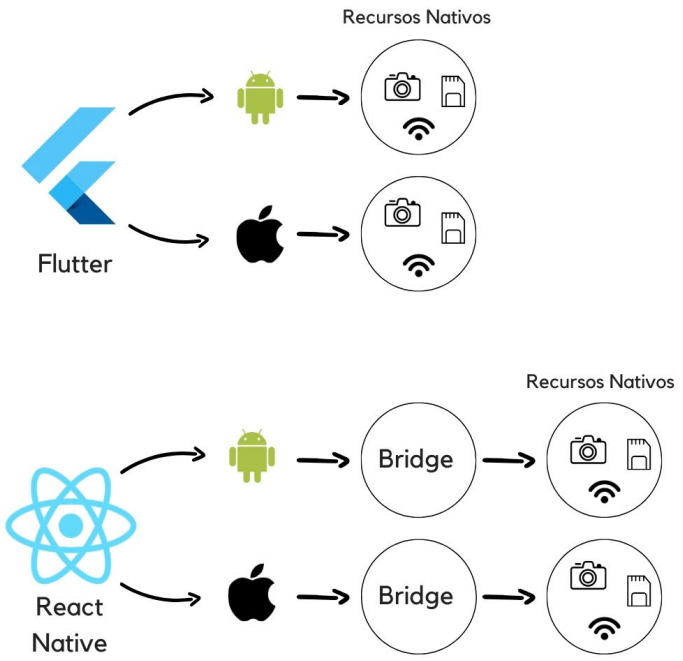
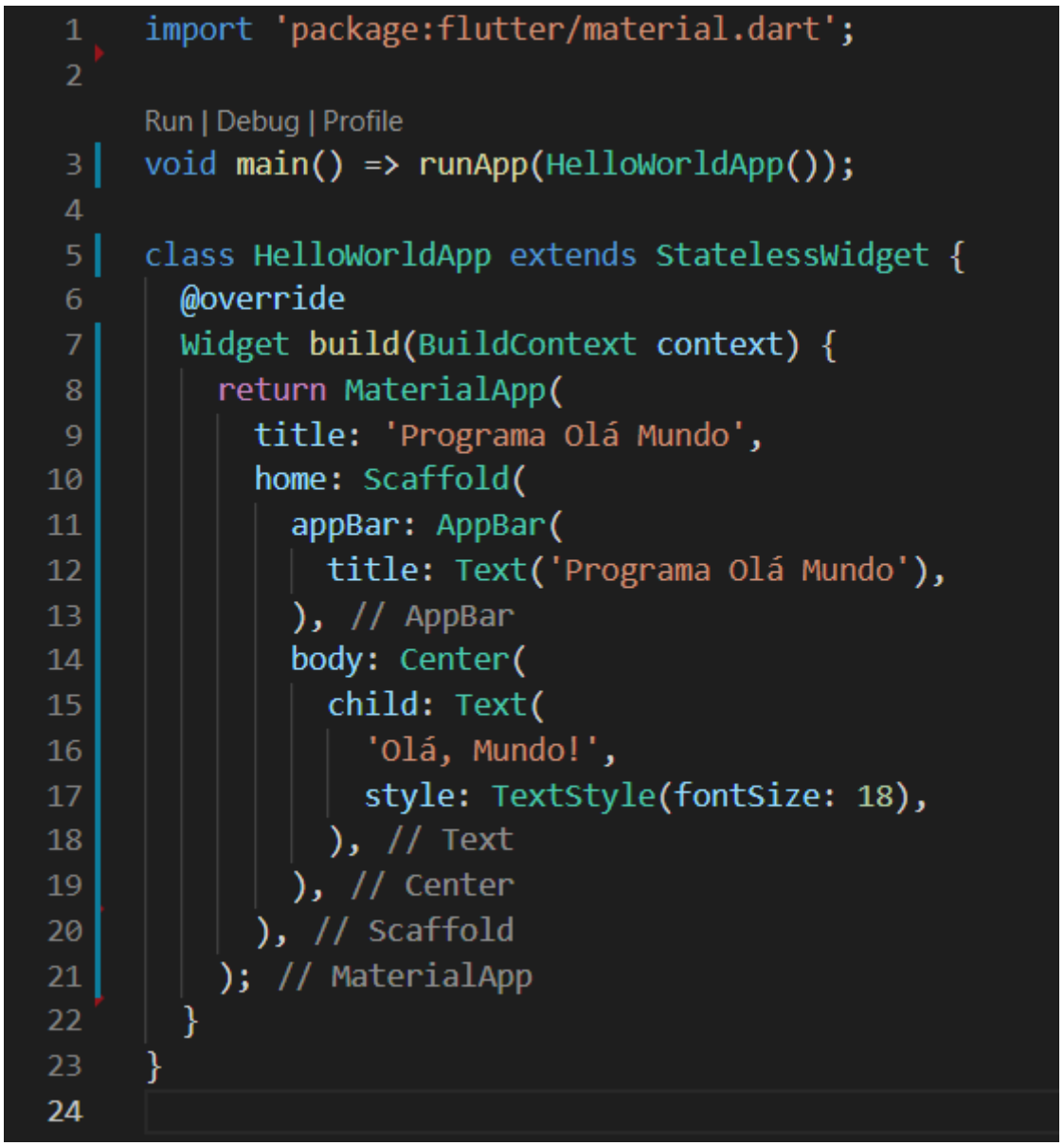


Figura 12: Ilustração da diferença de acessos entre o React Native e o Flutter

O código gerado por Flutter ainda não é tão eficiente quanto código nativo desenvolvido especificamente para cada dispositivo, mas a versatilidade multiplataforma e o acesso direto aos recursos nativos do sistema tornam o Flutter uma boa escolha de framework para desenvolver o “BoraMarcar”.

Segue um exemplo da sintaxe Flutter usando o exemplo clássico do programa *Hello World*;



```dart
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() => runApp(HelloWorldApp());

class HelloWorldApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Programa Olá Mundo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Programa Olá Mundo'),
        ), // AppBar
        body: Center(
          child: Text(
            'Olá, Mundo!',
            style: TextStyle(fontSize: 18),
          ), // Text
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}

```

Figura 13: Exemplo *Hello World* do Flutter

O Flutter foi escolhido como base para o aplicativo porque, apesar dele ser relativamente recente, eu possuía um pouco de experiência com o framework, tendo estudado sobre ele por curiosidade e fiz um aplicativo como projeto em uma matéria da PUC. O Flutter também tem tido aumento de interesse no mercado recentemente, com empresas como NuBank[2] e iFood[3] usando Flutter nos seus novos produtos.

Dart (Versão 2.14.1)

Os aplicativos Flutter são escritos na linguagem de programação Dart e fazem uso de muitos dos recursos mais avançados da linguagem.

No Windows, macOS e Linux, por meio do projeto semi-oficial Flutter Desktop Embedding, o Flutter é executado na máquina virtual Dart, que possui um mecanismo de compilação que ocorre em tempo de execução. Ao escrever e depurar um aplicativo, o Flutter usa a compilação JIT (*just-in-time*), permitindo o "*hot reload*", através do qual as modificações nos arquivos de origem podem ser injetadas em um aplicativo em execução. O Flutter estende isso com suporte para *hot reload* de widgets *stateful*, onde na maioria dos casos as alterações no código-fonte podem ser refletidas imediatamente no aplicativo em execução, sem a necessidade de uma reinicialização ou perda do *state* [4].

As versões de lançamento dos aplicativos Flutter são compiladas com a compilação antecipada (AOT) no Android e no iOS, possibilitando o alto desempenho do Flutter em dispositivos móveis.

Firebase

O Firebase é uma plataforma BaaS (Backend-as-a-Service) que fornece aos desenvolvedores uma variedade de ferramentas e serviços para ajudá-los a desenvolver aplicativos de qualidade, aumentar sua base de usuários e ser mais lucrativo [5].

Toda sua base é construída na infraestrutura do Google, sendo categorizado como um programa de banco de dados NoSQL, que armazena dados em documentos do tipo JSON.

O Firebase complementa muito bem o Flutter, afinal eles são da mesma empresa. Flutter possui bibliotecas que facilitam o uso do Firebase, além de possuir muitas ferramentas úteis para o desenvolvimento de um aplicativo mobile.

O Firebase conta com um grande conjunto de ferramentas de desenvolvimento. Destas, o Realtime Database e o Cloud Firestore podem armazenar dados estruturados em documentos e sincronizar os aplicativos

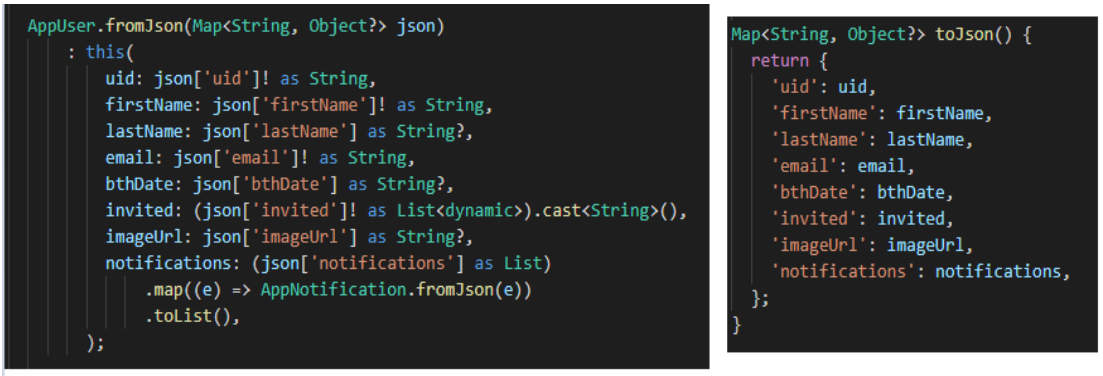
correspondentes em milissegundos sempre que ocorre uma transformação de dados. Destas ferramentas, as seguintes foram usadas no projeto;

A. Cloud Firestore [6]

Banco de dados de nuvem NoSQL. É a principal forma de armazenamento de dados do “BoraMarcar”.

Quando aplicativos são criados com os SDKs para Apple, Android e JavaScript todos os seus clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes.

Por causa da formatação JSON todas as classes foram criadas com um método *fromJson* e *toJson*, que formata e lê os dados recebidos e enviados para o Cloud Firestore.



```
AppUser.fromJson(Map<String, Object?> json)
    : this(
        uid: json['uid']! as String,
        firstName: json['firstName']! as String,
        lastName: json['lastName'] as String?,
        email: json['email']! as String,
        bthDate: json['bthDate'] as String?,
        invited: (json['invited']! as List<dynamic>).cast<String>(),
        imageUrl: json['imageUrl'] as String?,
        notifications: (json['notifications'] as List)
            .map((e) => AppNotification.fromJson(e))
            .toList(),
      );
```

```
Map<String, Object?> toJson() {
  return {
    'uid': uid,
    'firstName': firstName,
    'lastName': lastName,
    'email': email,
    'bthDate': bthDate,
    'invited': invited,
    'imageUrl': imageUrl,
    'notifications': notifications,
  };
}
```

Figura 14: Exemplo do método toJson e fromJson na classe AppUser

B. Cloud Functions [7]

O Cloud Functions para Firebase é um framework sem servidor que permite executar automaticamente o código de back-end em resposta a eventos acionados por recursos do Firebase e solicitações HTTPS. Essa funcionalidade foi usada especificamente para o envio de notificações. Diferente das outras funcionalidades listadas

C. Firebase Authentication [8]

O Firebase Authentication fornece serviços de back-end, SDKs fáceis de usar e bibliotecas de IU prontas para autenticar usuários no aplicativo. Ele oferece

suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook e Twitter, entre outros.

O Firebase Authentication é estreitamente integrado a outros serviços do Firebase e aproveita os padrões do setor, como OAuth 2.0 e OpenID Connect, para que possa ser facilmente integrado ao seu back-end personalizado.

D. Cloud Storage [9]

O Cloud Storage para Firebase é um serviço de armazenamento de objetos avançado, simples e econômico, criado para a escala do Google. Neste projeto ele é usado para armazenar imagens enviadas pelos usuários como fotos de perfil, capas de eventos ou imagens de grupos.

Visual Studio Code (Versão 1.62.3)

O Visual Studio Code foi o editor de texto usado. Ele foi escolhido pela versatilidade e leveza do programa, além da minha própria familiaridade com ele.

Uma vantagem do Code são as Extensões, que permitem que ele seja customizado para muitos tipos de desenvolvimento. As extensões usadas para o “BoraMarcar” foram;

A. Flutter [10]

A extensão adiciona suporte para edição, refatoração, execução e recarga de aplicativos móveis Flutter de maneira eficaz, bem como suporte para a linguagem de programação Dart.

Uma extensão com as necessidades básicas para desenvolver em Flutter no Visual Studio Code.

B. Flutter MVC Generator [11]

Extensão de código de estúdio visual para gerar código de modelo de padrões MVC e MV usando provedores.

Ajudou na aplicação da arquitetura do projeto.

C. Firebase Explorer [12]

   Uma extensão do Visual Studio Code para explorar e gerenciar seus projetos do Firebase.

   Essa extensão agilizou bastante o processo de teste com o Firebase, permitindo verificações rápidas dos dados salvos no Cloud Firestore.

D. YAML Language Support [13]

   Fornece suporte abrangente de linguagem YAML para Visual Studio Code, por meio do yaml-language-server, com suporte de sintaxe Kubernetes integrado.

   Foi incluído no projeto apenas para a leitura do arquivo *pubspec.yaml*, que contém informações relevantes do aplicativo, como a versão, o ambiente e as dependências.

Android Studio (Arctic Fox 2020.3.1 Patch 3)

IDE de escolha no projeto para simular um Android no ambiente de produção. Feito especificamente para o desenvolvimento para Android, ele substituiu Eclipse Android Development Tools (ADT) como a IDE primária do Google de desenvolvimento nativo para Android.

Apesar de possuir outras funcionalidades, ele foi usado majoritariamente como um emulador de Android, especificamente um Google Pixel 4 com Android 11. Além do celular emulado, foi usado um telefone físico para teste, um Motorola One com Android 10.

Github

GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git.

O código do “BoraMarcar” foi salvo no GitHub, mesmo sendo um projeto individual, guardar o projeto em um repositório ajudou no controle de versões e principalmente na segurança de ter uma versão acessível em caso de perda do

projeto, o que foi útil quando houve um problema técnico durante o desenvolvimento e o código no computador foi perdido.

O link para o repositório do “BoraMarcar” se encontra na seção Links Importantes.

Trello

O Trello foi usado como uma ferramenta organizadora, para definir as tarefas a serem feitas durante a semana, sua classificação e prioridade para o projeto. Sua utilidade será expandida na seção Processo de Desenvolvimento.

3.2. Processo de Desenvolvimento

Embora o desenvolvimento do projeto não tenha seguido nenhuma metodologia específica, foi aplicado um processo iterativo e incremental de ciclos curtos, no caso uma semana, em que cada ciclo consistia na concepção, implementação e revisão.

Durante cada semana, tarefas eram selecionadas no quadro Trello do projeto. Cada tarefa era classificada como *Life Style Improvement*, *Features* Secundárias ou *Features* Essenciais, além de classificações da área de atuação da tarefa; *backend* ou *frontend*. As tarefas tarefas eram classificadas com um grau de complexidade de um a cinco, e durante a semana uma soma mínima da complexidade das tarefas escolhidas deveria ser de cinco ou mais. Por exemplo; uma tarefa com complexidade 2 e uma tarefa de complexidade 3 em uma semana. Na semana seguinte, apenas uma tarefa de complexidade 5, e na semana após essa três tarefas de complexidade 3. Caso alguma tarefa não tenha sido concluída, ela ainda contaria para as tarefas da semana seguinte. A figura 15 mostra um exemplo de tarefa no Trello. No final da semana (normalmente na sexta ou no domingo) as tarefas feitas são revisadas e as novas tarefas são definidas.

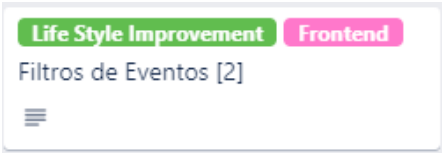


Figura 15: Exemplo de quadro de tarefa do Trello.

3.3. Atividades Realizadas e Lições Aprendidas

O processo de desenvolvimento começou com a fase de planejamento em abril de 2021 e segue até novembro de 2021.

Durante esse período, várias alterações foram feitas no projeto. Nesta seção vamos destacar algumas das mais importantes e aprendizados e dificuldades durante o desenvolvimento.

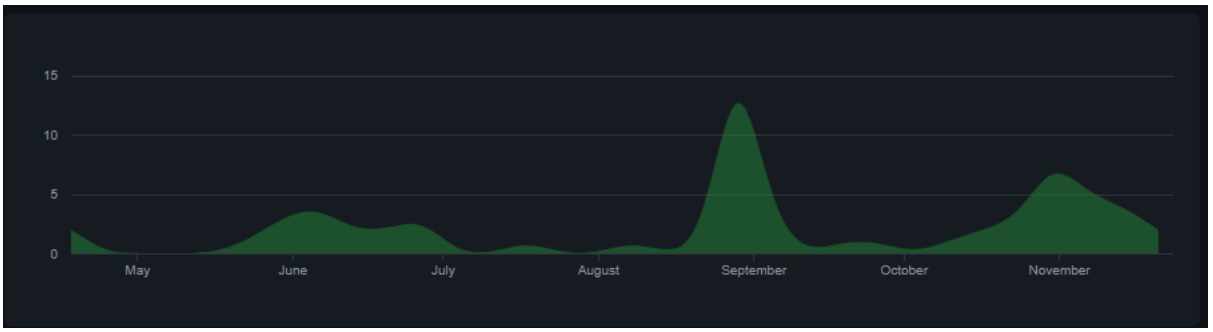


Figura 16: O gráfico indica o número de commits feitos durante o ano, entre maio e novembro.

Abril

O período do começo do projeto foi voltado para o planejamento do projeto, definindo as propostas do projeto, estimando quais ferramentas seriam usadas, o escopo do aplicativo, dentre outros fatores primordiais.

As escolhas de ferramentas, assim como as suas justificativas, estão na seção Ferramentas.

O repositório foi criado dia 18 de abril com os componentes básicos de um projeto Flutter.

Maio

Em maio começaram a ser implementadas as primeiras classes do projeto; Schedule, Event e AppUser (inicialmente chamada apenas de User, mas foi mudada para que não houvesse conflito com a classe User do Firebase Auth), e uma versão preliminar do método que irá determinar a melhor data para o evento.

Junho

Em junho foi implementado toda parte de autenticação. Primeiro foi estabelecido uma conexão com o Firebase[14], foi criado o AuthController, para mediar a interação entre o aplicativo e o Firebase Authentication e com o Cloud Firebase para armazenamento dos dados do usuário. A tela de autenticação também foi implementada.

Além da autenticação foram implementados os assets de imagem e fontes.

Em março de 2021, a Google anunciou a versão 2.0 do Flutter, que, entre outras mudanças, implementou o sistema de *null safety*. Durante a fase de planejamento, eu decidi que iria continuar com uma versão mais antiga, pois era a que eu já estava familiarizado e que eu já tinha alguns componentes que eu poderia usar. Isso se provou um erro. Muitos dos plugins importantes para o projeto precisavam da versão 2.0 do Flutter (ou superior), então algumas semanas de junho foram dedicadas somente a adequar o que já havia sido feito para a versão mais recente do Flutter.

Julho

A maior parte de julho não faz parte do período letivo, então não houve muito progresso durante esse período, apenas alguns estudos e refatoração do código.

Agosto

Agosto começou com uma refatoração do código, para me re-familiar com o projeto depois de um tempo sem mexer muito nele.

Foi implementado o componente das configurações, onde o usuário pode editar as suas informações. Nesse período foram implementados os métodos para usar o Firebase Storage como forma de armazenar fotos, tanto para usuários (que não usam contas do Facebook ou Google) quanto para eventos, e eventualmente grupos.

Também foi implementada a autenticação pelo Facebook e pelo Google, que estava pendente desde junho.

Setembro

O começo de setembro teve um foco no componente Schedule do aplicativo, refatorando o ScheduleController e a página dos horários do usuário, e criando uma nova versão do método para determinar a data ideal do evento.

O componente dos grupos foi introduzido no sistema, com todos os componentes MVC necessários (tela, controller, modelo).

Além da implementação desses componentes, muitas telas foram ajustadas para melhorar a UX.

Outubro

Uma das implementações de outubro foram as notificações push, uma parte importante do projeto. O maior problema em relação a implementar notificações é a quantidade de informações datadas. Em artigos e tutoriais de notificações de menos de um ano atrás já estavam irrelevantes porque eles foram escritos com plugins que na época estavam na versão 2 ou três, mas que agora já estão na versão 11, e muitos métodos instruídos não funcionam ou nessas versões, e usar versões antigos poderia causar conflitos com as outras dependências. Essa é uma consequência de trabalhar com um programa que ainda é muito novo, ele está sempre em evolução, então é preciso ficar atento para acompanhá-lo.

Um outro problema técnico ocorreu em outubro. Na terceira semana o computador que eu estava usando para desenvolver o projeto pifou, e eu estava a uma semana sem enviar as alterações para o repositório. Além disso, o conserto do computador demorou mais uma semana. Efetivamente, foram duas semanas de trabalho perdido em outubro.

Novembro

Em novembro houve um esforço maior para tentar compensar o tempo perdido em outubro, o número de tarefas semanais foi aumentado para tentar recuperar o tempo perdido. Foram também implementadas funcionalidades que já estavam pendentes, como poder adicionar convidados por grupo e convidar múltiplos usuários ao dividir os seus e-mail com ponto e vírgula na área de “adicionar convidados”.

Também houveram alterações da UI para implementar o novo tema e melhorar a UX do usuário.

Depois do feedback de alguns usuários sobre o aplicativo, algumas mudanças estéticas foram feitas e um novo tema para o aplicativo foi implementado.

Porém, a maior parte do mês foi dedicada a refatorações, limpeza de código e a escrita do relatório final.

3.4. Testes

Por causa de complicações no durante o desenvolvimento do “BoraMarcar”, o período de teste não foi tão extenso quanto poderia ter sido. Durante cada ciclo semanal de desenvolvimento (especificado na seção Processo de Desenvolvimento) as alterações no sistema, sejam ajustes ou implementações, eram submetidas a testes unitários antes que a tarefa fosse considerada concluída.

Como o “BoraMarcar” é um aplicativo que depende bastante da interação com usuários, foram feitos testes unitários com dois grupos distintos; amigos e familiares. Os amigos escolhidos para testar o aplicativo tinham mais conhecimento técnico no uso de aplicativos, com alguns até com experiência de desenvolvimento na área. O feedback deles foi mais voltado para interface e design do aplicativo, com algumas sugestões técnicas.

Em um completo oposto, os familiares que testaram o aplicativo eram pessoas mais velhas e sem muito conhecimento tecnológico, que tiveram dificuldade de usar o aplicativo sem algum auxílio. Isso inspirou a necessidade de um processo de *on-boarding* para explicar o aplicativo para os mais leigos.

As experiências dos grupos com o aplicativo foram registradas e ajudaram a moldar o aplicativo a forma atual.

3.5. Limitações e Trabalhos Futuros

Algumas das funcionalidades propostas ou cogitadas durante o processo de idealização deste aplicativo não puderam ser compridas por falta de tempo ou de recursos;

- Versão iOS e Web

  Um dos grandes pontos fortes do Flutter é a versatilidade de plataformas que ele pode abranger com poucas complicações. Infelizmente, para criar uma versão para iOS seria necessário dispositivos com o sistema operacional iOS, recurso este que eu não dispunha. A versão Web não era uma proposta original, durante o período do planejamento do projeto, o Flutter Web ainda não tinha uma versão estável confiável, porém ela foi incluída como um *stretch-goal* do projeto. Adaptar o código atual para acomodar essas versões não seria muito complicado, com as devidas ferramentas e tempo disponíveis.

- Acesso por Link Externo

  Como a forma de convidar usuários para eventos é feita pelo próprio aplicativo, esse recurso se tornou não muito relevante.

- *On-Boarding*

  On-Boarding é o processo de instruir novos usuários nas funcionalidades do seu aplicativo, normalmente com uma sequência de telas ou imagens indicando o que cada componente do aplicativo deve fazer. Como esse não era um componente fundamental ele não foi implementado.

- Preferências do Aplicativo

  Funcionalidades como modo noturno, linguagem, tamanho da fonte, componentes que melhoram a UX não foram implementados por falta de tempo e de prioridade.

- Questões de Privacidade

  Como o aplicativo lida com informações do usuário, no caso o seu horário e eventos, o responsável pelo evento poderia escolher o nível de acesso a informações que os convidados têm a informações do evento e de outros convidados, como saber quem são os outros convidados.

- Acesso ao Evento ao ser Convidado

  No momento apenas usuários convidados têm acesso a página do evento, mas um usuário que recebeu o convite poderia ter acesso ao evento, para saber do que se trata antes de aceitar. Atualmente o usuário só tem acesso ao nome do evento e ao nome do responsável que o convidou.

- Integração com o Google Calendar

  Apesar de ser um componente mais importante para a proposta do aplicativo, ele ainda não era essencial para a proposta do aplicativo, e foi deixado de lado para que o foco se mantivesse em outras funcionalidades.

4. Arquitetura do Flutter

Quando inicializamos qualquer aplicativo Flutter, a ferramenta cria um modelo de aplicativo com arquivos e pastas padrão, conforme mostrado na figura a seguir.

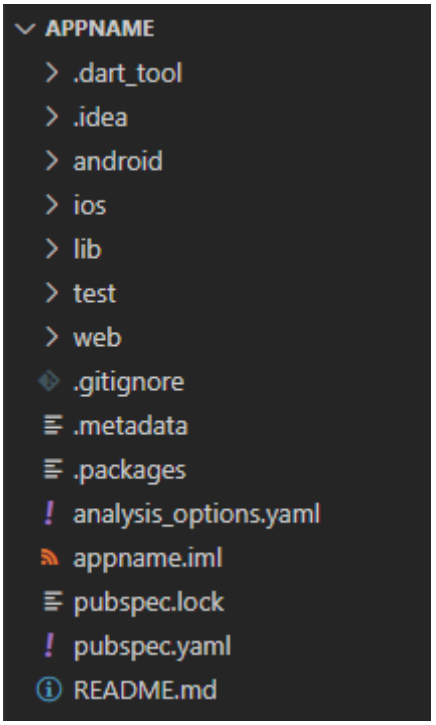


Figura 17: Um projeto Flutter recém criado

- **android**

  A pasta Android contém arquivos e pastas necessários para executar o aplicativo em um sistema operacional Android. Esses arquivos e pastas são gerados automaticamente durante a criação do projeto de flutter. É recomendável que essas pastas e arquivos sejam deixados como estão.

  As subpastas principais da pasta android são a pasta res e o arquivo *AndroidManifest.xml*. A pasta *res* contém recursos não programáveis necessários para o aplicativo, como ícones, imagens e fontes, enquanto o arquivo *AndroidManifest.xml* contém informações necessárias para o SDK do aplicativo.

- **ios**

  Como a pasta android, esta pasta contém os arquivos exigidos pelo aplicativo para executar o código dart nas plataformas iOS. Os principais arquivos na pasta ios são a pasta *Assets.xcassests*, que como a pasta *res* do android contém recursos não programáveis necessários para o aplicativo, e o arquivo *info.plist*, similar ao *AndroidManifest.xml*.

- **.idea**

  A pasta .idea conterá configurações relacionadas ao editor de código que você está usando para construir o aplicativo. Essas configurações são específicas para o projeto atual.

- **lib**

  A pasta mais importante do projeto, onde fica armazenado o código dart do projeto. Por padrão, a pasta lib contém o arquivo main.dart, que é o ponto de entrada do aplicativo.

- .**gitignore**

  Um arquivo *gitignore* específica arquivos intencionalmente não rastreados que o Git deve ignorar.

- **.dart_tool**

  O diretório .dart_tool, que é uma adição da versão 2 do Dart, é usado pelo pub e outras ferramentas. Ele substitui o diretório .pub a partir do lançamento do SDK 2.0.0-dev.32.0.

- .**metadata**

  Ele contém metadados exigidos pelo Flutter para rastrear o projeto de vibração.

- **.packages**

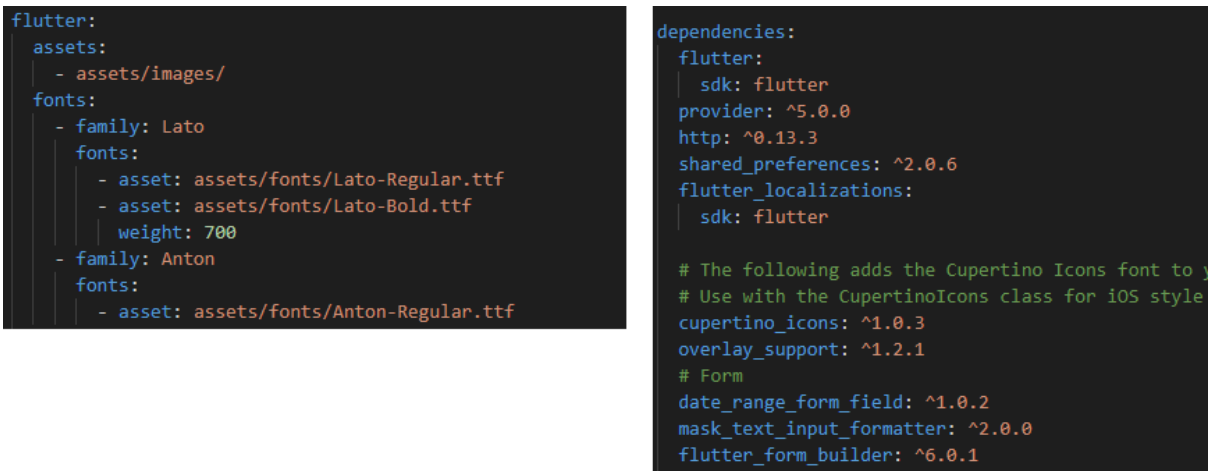
  Como Flutter é um framework, ele vem com vários pacotes e bibliotecas. O arquivo *.packages* contém o caminho para cada pacote e biblioteca usados no projeto. Este arquivo não deve ser alterado pelo desenvolvedor, ele será alterado com as mudanças das dependências.

- **pubspec.lock**

  Este arquivo contém a versão de cada dependência e pacotes usados no aplicativo flutter.

- **pubspec.yaml**

  Este é o arquivo que usamos para adicionar metadados e configurações específicas para nosso aplicativo. Com a ajuda deste arquivo, podemos configurar dependências como ativos de imagem, fontes e versões de aplicativos.



```
flutter:
  assets:
    - assets/images/
  fonts:
    - family: Lato
      fonts:
        - asset: assets/fonts/Lato-Regular.ttf
        - asset: assets/fonts/Lato-Bold.ttf
          weight: 700
    - family: Anton
      fonts:
        - asset: assets/fonts/Anton-Regular.ttf
```

```
dependencies:
  flutter:
    sdk: flutter
  provider: ^5.0.0
  http: ^0.13.3
  shared_preferences: ^2.0.6
  flutter_localizations:
    sdk: flutter

  # The following adds the Cupertino Icons font to y
  # Use with the CupertinoIcons class for iOS style
  cupertino_icons: ^1.0.3
  overlay_support: ^1.2.1
  # Form
  date_range_form_field: ^1.0.2
  mask_text_input_formatter: ^2.0.0
  flutter_form_builder: ^6.0.1
```

Figura 18: Exemplo de configuração de assets (a esquerda) e dependências (a direita) no arquivo pub spec.yaml do “BoraMarcar”

- **README**

  Este arquivo markdown é usado para descrever seu aplicativo no repositório GitHub.

5. Arquitetura do Projeto

5.1. Escolha da Arquitetura

Durante a etapa de planejamento, duas arquiteturas se destacaram como boas opções de padrão de projeto; **MVC** (Model-View-Controller) e **MVP** (Model-View-Presenter).

MVC é um padrão de arquitetura que já foi a única maneira de desenvolver um aplicativo da web. Ele é um padrão de apresentação da interface do usuário que se concentra em separar a UI (View) de sua camada de negócios (Model). Para isso, o MVC define três componentes:

- View: apresenta os elementos da UI.
- Controller: responde às ações da UI.
- Model: comportamentos de negócio e gerenciamento de estado.

Isso reduz a quantidade de tempo e esforço necessários para estender, testar e manter o aplicativo. Entre a UI, a lógica de vinculação de dados e as atividades de negócios, o processo otimiza o tamanho da classe View.

Na maioria das implementações todos os três componentes podem interagir diretamente uns com os outros e em algumas implementações o controlador é responsável por determinar qual visualização vai ser exibida. A figura 19 demonstra a estrutura de um projeto MVC.

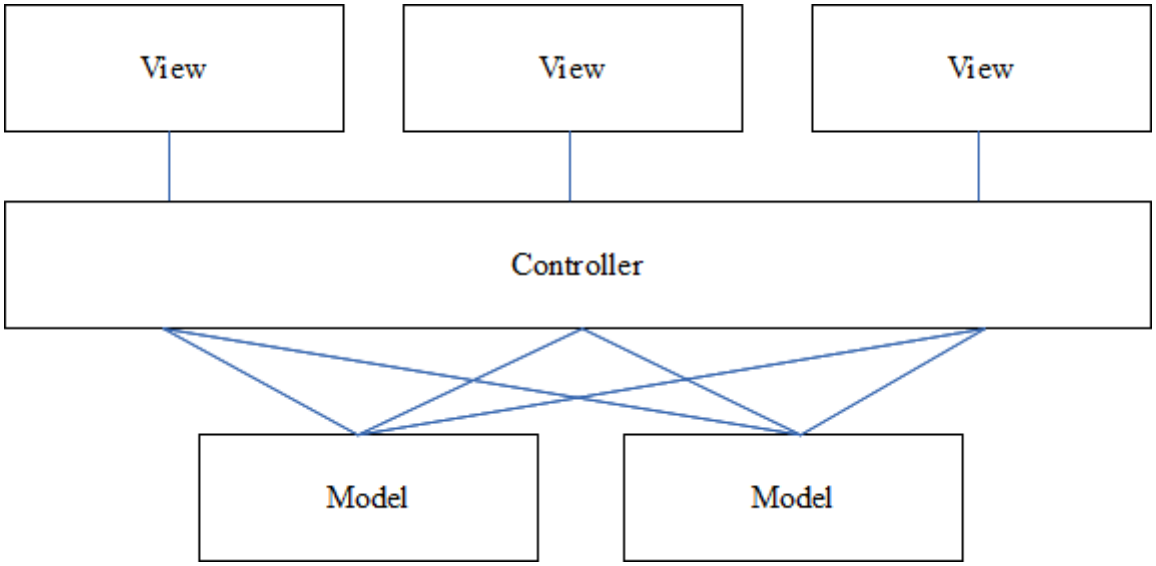


Figura 19: Diagrama do Model View Controller.

O MVP também é um padrão de apresentação da interface do usuário e é considerado por muitos uma evolução dos conceitos do MVC. Ele separa as responsabilidades em quatro componentes:

- View: apresenta os elementos da UI.
- Presenter (Interface): interface que define os eventos que o Presenter poderá responder. Serve unicamente para desacoplar a View do Presenter.
- Presenter: responde às ações da UI controlando a interação entre a View e o Model.
- Model: comportamentos de negócio e gerenciamento de estado.

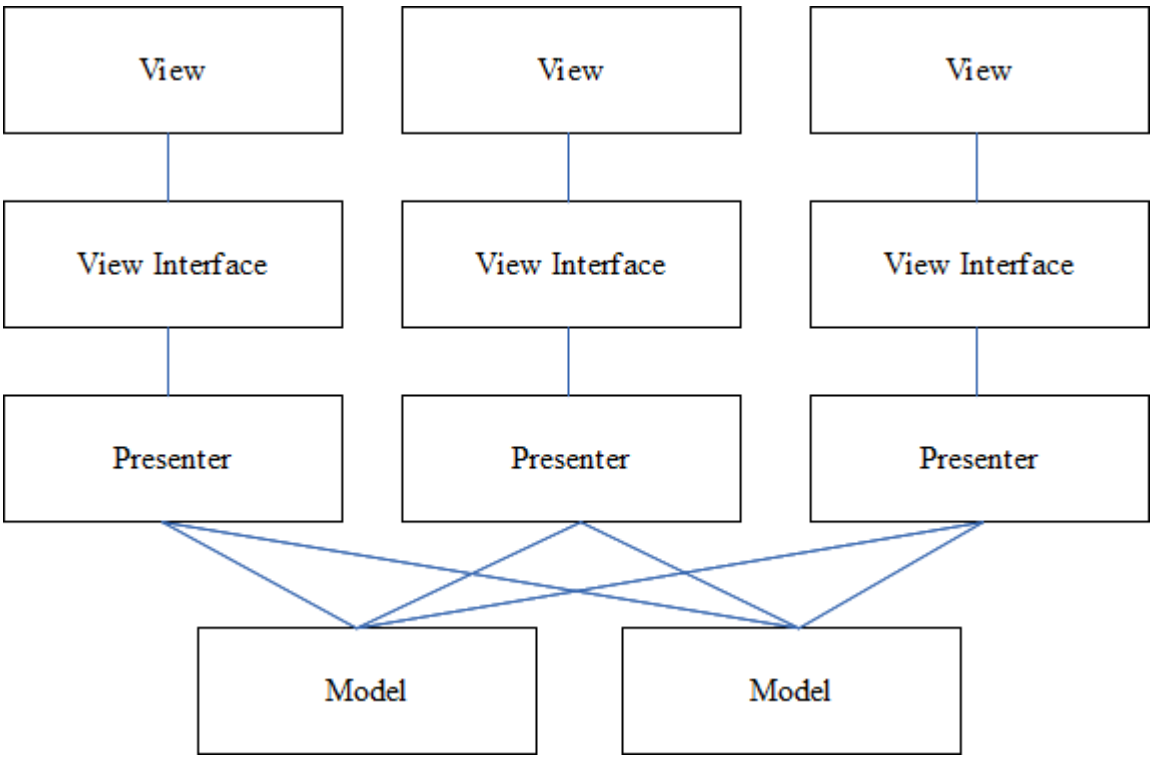


Figura 20: Diagrama do Model View Presenter.

Na prática, não há muitas diferenças entre as duas propostas de apresentação. Ambas se concentram em separar as responsabilidades entre as camadas e incentivar o desacoplamento da UI (View) com a camada de negócios (Model). Porém no MVC o Controller é baseado em comportamentos e podem ser compartilhados em múltiplas Views, enquanto o Presenter do MVP possui um relacionamento 1 para 1 com o View, ou seja, para cada View existe um Presenter. Isso facilita o processo de teste do aplicativo [15].

Com essas informações é possível determinar que o MVP é mais difícil de implementar do que o MVC, mas é mais indicado para projetos maiores, pelo seu desacoplamento entre a View e o Presenter, pela utilização de uma interface que ajuda na criação de testes e permite uma evolução estrutural e de apresentação do aplicativo maiores complicações.

Já o MVC se mostrou melhor para projetos menores onde é possível trabalhar com uma burocracia de comunicação menor, reaproveitar rapidamente os comportamentos dos Controllers entre múltiplas Views e ainda ter uma boa separação para a manutenção. Logo o MVC atende melhor às necessidades desse projeto.

É válido mencionar outros padrões de arquitetura que poderiam ser implementados, como o MVVM (Model-View-ViewModel)[16], um padrão projetado para remover virtualmente todo o código GUI ("code-behind") da camada de visualização, mas que pode ser um padrão exagerado para interfaces mais simples[17] e o padrão BLoC (Business Logic of Component)[18] que é um padrão destinado ao Flutter, mas que usa o recurso Stream que não é utilizado neste projeto.

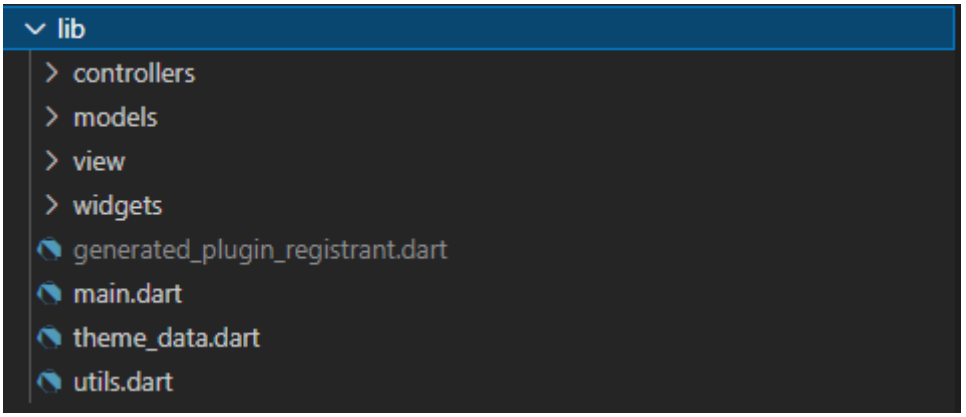


Figura 21: Estrutura da pasta lib que contém os códigos em dart.

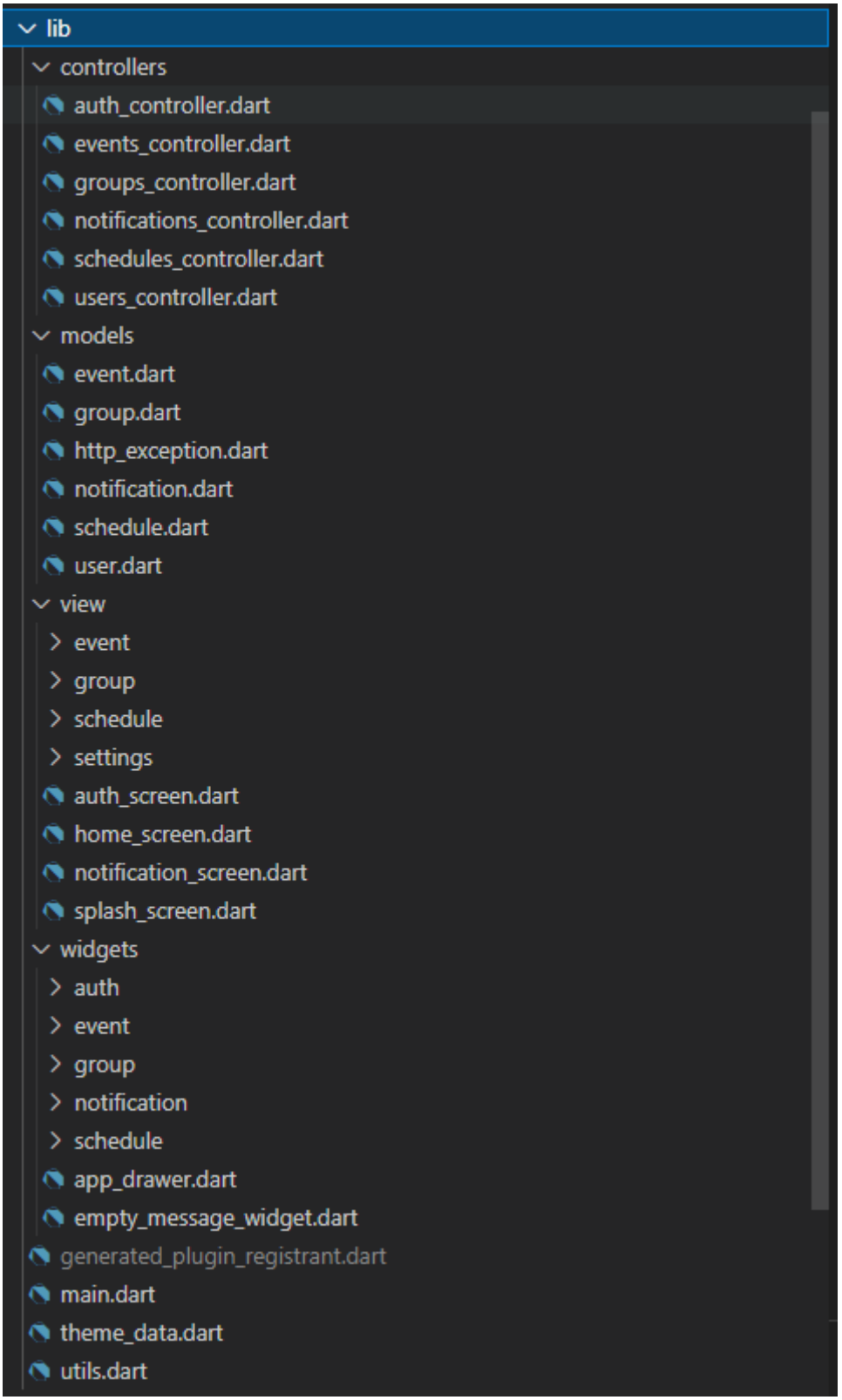


Figura 22: Estrutura da pasta lib expandida.

5.2. Estrutura do Projeto

O “BoraMarcar” seguiu os padrões MVC determinados acima, resultando nas seguintes camadas.

A camada dos Controllers realiza toda interação do aplicativo com o Firebase e determina o comportamento da View.

A camada View armazena as telas com que o usuário irá interagir.

A camada Widgets armazena Widgets específicos feitos para o “BoraMarcar”, que são usados em uma ou mais páginas.

A camada dos Models contém as classes criadas para o aplicativo;

- AppUser
- Event
- Group
- Schedule
- Notification

A relação das classes, assim como seus atributos e métodos, são especificados no diagrama de classe abaixo.

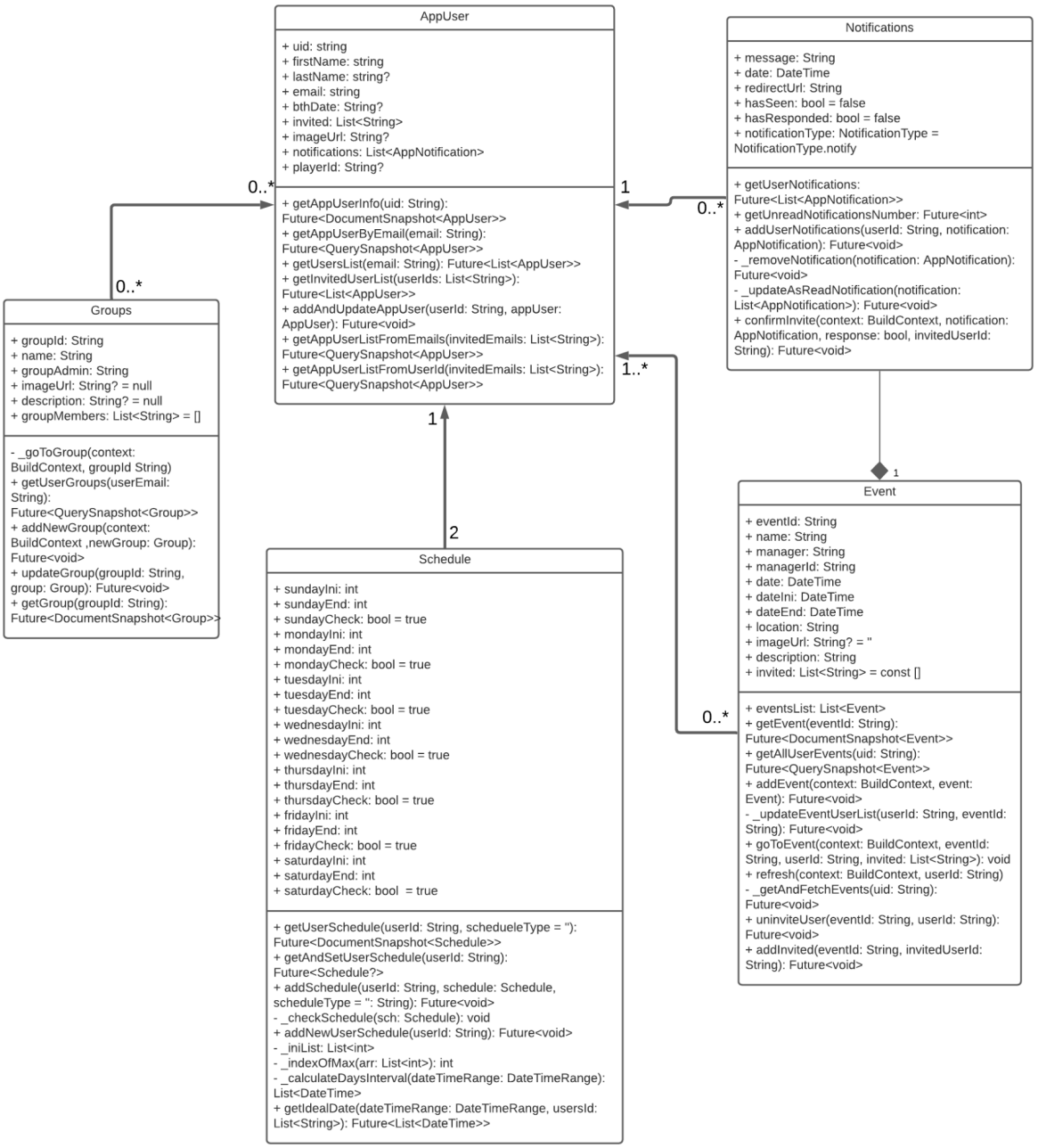


Figura 23: Diagrama de Classe do “BoraMarcar”

6. Considerações Finais

Neste projeto foi desenvolvido um aplicativo Android que tenta melhorar a produtividade e vida social dos usuários. O cerne do projeto foi tentar ser uma ferramenta que ajuda no equilíbrio entre trabalho e lazer, ao tentar separar e organizar esses dois tipos de horários. Em geral eu creio que o aplicativo é capaz de atingir o seu objetivo, com um bom grau de sucesso.

Como um projeto acadêmico, ele foi um bom exemplo da aplicação dos conceitos aprendidos durante o curso de ciência da computação. Vários aprendizados mais técnicos como práticas de programação e uso de banco de dados foram aplicados no projeto, assim como conceitos como arquitetura de software e metodologias ágeis.

Além disso, ele foi uma excelente experiência na área de desenvolvimento mobile, como uma experiência como desenvolvedor full-stack. O Flutter se mostrou uma boa ferramenta para desenvolvimento mobile e esse projeto me ajudou a me familiarizar com ele, assim como as outras ferramentas do projeto.

Neste momento mais retrospectivo do relatório, teriam algumas coisas que eu teria feito diferente no projeto. No período de concepção do projeto, eu fiquei preocupado do projeto ser um pouco anêmico para os requerimentos de um projeto final, então acabei adicionando mais funcionalidades secundárias, que, apesar de terem sido boas fontes de experiência em desenvolvimento, acabaram ofuscando algumas funcionalidades mais importantes. Além disso, por volta do final do projeto, eu tive a ideia de que talvez o aplicativo funcionasse melhor como um complemento a outros aplicativos de agenda, como o Google Calendar, do que como um aplicativo próprio. Mas essas realizações só surgem depois de investir tempo em um projeto.

Links Importantes

- Link para o repositório do projeto: <https://github.com/GustBM/boramarcarapp>
- Link visualização Diagrama de Classes: <https://lucid.app/lucidchart/fb230a53-d9e4-476a-9655-30bf6d192b58/view?page=0_0&invitationId=inv_4e46e8b3-0c23-465c-b59a-5656847b461c#>
- Link Docs Flutter: <https://docs.flutter.dev/>
- Link Docs Firebase : <https://firebase.google.com/docs>

Referências

1. DE ANDRADE, Ana Paula. **O que é Flutter?**. TreinaWeb, 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-flutter>. Acesso em 20 de nov. 2021.
2. FREIRE, Alexandre. **Porquê nós achamos que Flutter vai nos ajudar a escalar o desenvolvimento mobile no Nubank.** medium, 2019. Disponível em: <https://medium.com/flutter-comunidade-br/porqu%C3%AA-n%C3%B3s-achamos-que-flutter-vai-nos-ajudar-a-escalar-o-desenvolvimento-mobile-no-nubank-95d07b4554d7>. Acesso em 24 de nov. 2021.
3. MATIAS, Samuel. **Por que optamos por usar Flutter em nosso novo produto no iFood.** movile, 2020. Disponível em: <https://movile.blog/por-que-optamos-por-usar-flutter-em-nosso-novo-produto-no-ifood/>. Acesso em 24 de nov. 2021.
4. LELEL, Wm. **Why Flutter Uses Dart**. Hackernoon, 2017. Disponível em: <https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf>. Acesso em 20 de nov. 2021.

5. ESPLIN, Chris. **What is Firebase?**. howtofirebase, 2016. Disponível em: <https://howtofirebase.com/what-is-firebase-fcb8614ba442>. Acesso em 21 de nov. 2021.

6. Documentação Cloud Firestore. Disponível em: <https://firebase.google.com/docs/firestore>.

7. **Documentação Cloud Functions**. Disponível em: <https://firebase.google.com/docs/functions>.

8. **Documentação Firebase Authentication**. Disponível em: <https://firebase.google.com/docs/auth>.

9. **Documentação Cloud Storage**. Disponível em: <https://firebase.google.com/docs/storage>.

10. **Flutter para Visual Studio Code**. Disponível em : <https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter>. Versão 3.28.0.

11. **Flutter MVC Generator** para Visual Studio Code. Disponível em: <https://marketplace.visualstudio.com/items?itemName=Kurosh.flutter-mvc-generator>. Versão 0.1.0.

12. **Firebase Explorer** para Visual Studio Code. Disponível em: <https://marketplace.visualstudio.com/items?itemName=jsayol.firebase-explorer>. Versão 0.3.3.

13. **YAML** para Visual Studio Code. Disponível em: <https://marketplace.visualstudio.com/items?itemName=redhat.vscode-yaml> . Versão 1.2.0.

14. **Adicionar o Firebase ao app Flutter.** Disponível em: <https://firebase.google.com/docs/flutter/setup?hl=pt-br&platform=ios>. Acesso em 24 de nov. de 2021.

15. GALHARDO, Macello. **Desmistificando o MVC e MVP no Android**. Medium, 2016. Disponível em:

<https://medium.com/android-dev-br/desmistificando-o-mvc-e-mvp-no-android-abe927d01df7>. Acesso em: 21 de nov. 2021.

16. **Differences between MVC MVP and MVVM**. Partechit, 2014. Disponível em: <https://partechit.medium.com/differences-between-mvc-mvp-and-mvvm-52528a8935fb>. Acesso em 21 de nov. de 2021.

17. GOSSMAN, John. **Advantages and disadvantages of M-V-VM**. Microsoft, 2006. Disponível em: <https://docs.microsoft.com/en-gb/archive/blogs/johngossman/advantages-and-disadvantages-of-m-v-vm>. Acesso em 21 de nov. de 2021.

18. **BLoC**. flutterparainiciantes, 2021. Disponível em: <https://www.flutterparainiciantes.com.br/gerenciamento-de-estado/bloc>. Acesso em 21 de nov. de 2021.