Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

IGOR VAZ DA SILVA

UMA FORMA INTELIGENTE DE SER SAUDÁVEL

Assis/SP
2020

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

IGOR VAZ DA SILVA

UMA FORMA INTELIGENTE DE SER SAUDÁVEL

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Instituto Municipal do Ensino Superior de Assis – IMESA e Fundação Educacional do Município de Assis – FEMA, como requisito para a obtenção do Certificado de Conclusão.

Orientando: Igor Vaz da Silva
Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Assis/SP
2020

FICHA CATALOGRÁFICA

S586u SILVA, Igor Vaz.
Uma forma inteligente de ser saudável / Igor Vaz da Silva. – Assis, 2020.

35p.

Trabalho de conclusão do curso (Ciência da Computação). – Fundação Educacional do Município de Assis-FEMA

Orientador: Dr. Alex Sandro Romeo de Souza Poletto

1. Programa-academia 2. Dart 3. Flutter

CDD005.131

UMA FORMA INTELIGENTE DE SER SAUDÁVEL

IGOR VAZ DA SILVA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Prof. Dr. Alex Sandro Romeo de Souza Poletto

Examinador: _____
Prof. Me. Fábio Eder Cardoso

Assis/SP
2020

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me proporcionar saúde, sabedoria, determinação e disciplina.

Agradeço a minha família que sempre esteve ao meu lado, me motivando e apoiando no possível.

Ao Prof. Dr. Alex Sandro Romeo de Souza Poletto pela ótima orientação.

E agradeço a todos os meus amigos e colegas que sempre estiveram juntos de mim.

RESUMO

Como todos sabem, nos dias atuais o celular inteligente sempre esteve no cotidiano das pessoas, principalmente por ser um instrumento de fácil utilização e por ser compacto, podendo ser carregado no bolso para vários lugares. Através deste instrumento, pode-se trabalhar, se divertir, fazer compras, entre outros. Um fato muito importante que se deve levar em consideração é que pessoas sedentárias, em sua maioria utilizam o celular. Pensando nisso, surgiu a ideia de desenvolver um aplicativo que seja capaz de mesclar o uso do celular inteligente com a prática de atividade física, utilizando-se ferramentas nativas *android/ios* que contenham velocidade e flexibilidade, em qualquer mobile. O foco principal é estudar as ferramentas de desenvolvimento, plataformas ágeis e produzir um aplicativo para fins de utilização de praticantes de atividade física, com foco na musculação.
Palavras-chave: Academia, Treino, *Dart*, *Flutter*, Sedentarismo

ABSTRACT

As you all know, SmartPhone has always been in people's daily lives these days, mainly because it's easy to use and compact, and can be carried in your pocket to various places. Through this instrument we can work, have fun, shop, among others. A very important fact to take into account is that sedentary people mostly use the mobile phone. Thinking about it I had the idea to develop something that would be able to mix the use of SmartPhone with the practice of physical activity, using native android/ios tools that contain speed and flexibility through an application installed on any mobile. The main focus is to study the development tools, agile platforms and produce an application for the use of physical activity practitioners, with a focus on weight training.

Keywords: Academy, Training, Dart, Flutter, Sedentarism

LISTA DE ILUSTRAÇÕES

Figura 3-1 Exemplo Hot Reload (Flutter, 2020) 12

Figura 3-2 : Arquitetura Dart Flutter (Flutter, 2019) 14

Figura 3-3: Aplicação com ponte (Flutter, 2019)..... 14

Figura 3-4: Aplicação sem ponte (Flutter, 2019)..... 15

Figura 5-1: Diagrama de Casos de Uso Geral (Feita pelo autor) 21

Figura 6-1: Tela de Login (Feita pelo autor)..... 23

Figura 6-2: Tela de Cadastro (Feita pelo Autor) 24

Figura 6-3: Tela após login (Feita pelo autor) 25

Figura 6-4: Tela Tipo de Corpo (Feita pelo autor) 26

Figura 6-5: Tela Especificar Meta (Feita pelo autor) 27

Figura 6-6: Tela Condicionamento (Feita pelo autor) 28

Figura 6-7: Tela StateLess (Feita pelo autor) 29

Figura 6-8: Tela Requisição Metas (Feita pelo autor) 30

Figura 6-9: Tela Requisição Tipo Corpo (Feita pelo autor) 30

Figura 6-10: Tela Requisição Meta Corpo (Feita pelo autor) 31

Figura 6-11: Tela Requisição Condicionamento (Feita pelo autor) 31

SUMÁRIO

1. INTRODUÇÃO 9

1.1. OBJETIVOS 10

1.2. JUSTIFICATIVA 10

2. DART 11

2.1. FRAMEWORKS E BIBLIOTECAS DART 11

3. FLUTTER 12

3.1. ARQUITETURA FLUTTER 13

4. DESCRIÇÃO DO AMBIENTE DE DESENVOLVIMENTO 16

4.1. LINGUAGEM UTILIZADA 16

4.2. SDK UTILIZADO 16

4.3. FERRAMENTAS PARA O DESENVOLVIMENTO 16

4.3.1. ANDROID STUDIO VERSÃO 3.6.2 WINDOWS 16

4.3.2. Firebase 17

5. FERRAMENTAS DE ANÁLISE 18

5.1. DRAW.IO 18

5.2. DB DESIGNER 18

5.3. RECURSOS NECESSÁRIOS 19

5.3.1. ESPECIFICAÇÃO DE CUSTOS 19

5.4. LEVANTAMENTO DOS REQUISITOS 20

5.5. DIAGRAMAS 21

5.5.1. DIAGRAMAS DE CASO DE USO 21

6. DESENVOLVIMENTO DO APLICATIVO 23

6.1. Protótipos de tela com design 23

6.1.1. Codificação 29

7. CONCLUSÃO 32

8. REFERÊNCIAS 33

1. INTRODUÇÃO

Ao longo dos anos os Smartphones vêm ganhando mais espaço na vida das pessoas, tanto servindo como instrumento de trabalho, como em ocasiões de diversão. Com isso, se transformou em um objeto quase indispensável na vida de todas as pessoas. Anteriormente os celulares eram usados apenas como meio de comunicação, seja ele através de ligações ou mensagens, nos dias atuais esses detalhes mudaram muito com a evolução das tecnologias. Várias plataformas de desenvolvimento foram sendo criadas e otimizadas para uma melhor flexibilidade e desempenho de uma aplicação para mobiles. Um ferramenta de desenvolvimento de software (SDK) que vem se destacando bastante é o Flutter, que possibilita criar aplicativos bonitos, modernos e de alta performance. O aplicativo será desenvolvido com o objetivo de incentivar as pessoas sedentárias ou não para a prática de atividade física, ele salvará todos os dados de cada usuário e de acordo com as informações obtidas no cadastro, resultará em uma grade de treinos e exercícios específicos.

A obesidade tem se tornado uma epidemia nos últimos 25 anos, tanto em países desenvolvidos, como em países subdesenvolvidos, decorrente de maus hábitos alimentares e sedentarismo. (Dantas, 2020).

Um estudo realizado por Varady *et al*. (2013, p. 146) confrontou o efeito do jejum alternado em indivíduos obesos pelo tempo de 12 semanas, tanto na relação de composição corporal, como nos traços do perfil lipídico, sendo assim, 15 pessoas realizaram o processo de jejum em dias alternados diminuindo em até 75% do total as calorias diárias recomendadas, nos outros dias fora do jejum ela comiam a vontade, enquanto o grupo monitorado comia a vontade todos os dias, as respostas foram uma considerável perda de peso e nenhuma alteração nos valores (Triglicerides, LDL e HDL), confirmando assim que o jejum em dias alternados podem ser eficientes em pessoas obesas.

Com esse estudo de Varadyet, o aplicativo poderá levar em consideração o jejum alternativo para pessoas com IMC acima de 30, que é considerado obesidade.

1.1. OBJETIVOS

O objetivo principal foi desenvolver um aplicativo Mobile capaz de gerenciar rotinas de treino (musculação), repetições, séries, entre outros, para todos os tipos de pessoas. Se enquadrando também na perspectiva de o praticante poder se relacionar tanto com o instrutor físico, como o instrutor virtual que será a base fundamental de quem usará a aplicação

1.2. JUSTIFICATIVA

O tema “Uma forma inteligente de ser saudável” identifica bem o projeto a ser desenvolvido, pois através do smartphone, o indivíduo poderá seguir sua rotina de musculação, acompanhar o desenvolvimento pessoal e obter resultados rápidos. Outro ponto importante é mostrar que uma aplicação criada a partir de Dart + Flutter pode ser muito mais eficiente e proativa do que instrumentos de desenvolvimento que não são nativas.

2. DART

O Dart é uma linguagem de programação orientada a objetos, nela existem muito paradigmas criados pelo Google. A linguagem é muito versátil e pode ser usada para o desenvolvimento de aplicativos mobile, desktops, criação de scripts, e também no backend. Para que tudo isso ocorra se utilizam muitas plataformas, onde cada uma é específica para o ambiente de desenvolvimento utilizado.

A primeira aparição do Dart foi em uma conferência GOTO na Dinamarca em 2011, que posteriormente em fevereiro de 2012 teve uma nova versão lançada chamada de Dart 2. Para desenvolver com DART focada em web services ou script de comando deve ser utilizado a SDK(Software Developent KIT). Se o objetivo for desenvolvimento mobile, o SDK não será necessário, porém é preciso a instalação do framework Flutter que também foi criado pelo Google.

A Linguagem Dart é fortemente tipada, porém pode ser explicita. Os tipos mais utilizados são: int, double, string, boolean, list, map(chave-valor). (Dart, 2011).

2.1. FRAMEWORKS E BIBLIOTECAS DART

Os frameworks e bibliotecas mais utilizados pelo Dart são: AngularDart, Material Design Lite, OverReact, VueDart. Para o foco no desenvolvimento backend a biblioteca Aqueduct (Framework http utilizada na construção de RestAPIs na plataforma Dart Virtual Machine, uma opção de máquina virtual que fornece o ambiente de execução para uma linguagem de programação de alto nível) é uma ótima opção. Angel e Jaguar também se enquadram no mesmo estilo de ambiente para development. No momento da codificação é possível utilizar as mais conhecidas IDES (Integrated Development Environment ou Ambiente de Desenvolvimento), como o IntelliJ IDEA, VS Code, Sublime, Atom, Vim, entre outros. Os melhores exemplos de utilização que já estão em funcionamento atualmente, são o Google AdSense, e o Google Adwords. (Silva, 2020).

3. FLUTTER

O Flutter é um SDK (Software Development KIT) de código aberto criado pela Google, com o foco principal em aumentar a produtividade de aplicativos belos, compilados nativamente para mobiles, web e desktop, tudo isso através de único código base.

Hot Reload é uma ferramenta do Flutter capaz de ajudar o desenvolvedor a experimentar de maneira fácil e rápida todos os novos edites que fizer, podendo assim adicionar recursos e corrigir bugs na hora. O Hot Reload funciona injetando arquivos de código-fonte atualizações no Dart Virtual Machine (VM) que está em execução. Assim que a VM atualiza as classes do projeto, as alterações feitas nos campos e funções, a estrutura do Flutter reconstrói automaticamente todas as árvores de widgets, fazendo com que o desenvolvedor observe rapidamente todos os efeitos que foi alterado instantaneamente. (Flutter, 2020).

Para utilizar esta ferramenta, é necessário instalá-la dentro do editor Flutter, ou até mesmo na própria janela do terminal, funciona tanto em dispositivos físicos, como virtuais. Importante identificar que o Hot Reload só funciona quando os aplicativos Flutter estão em modo de depuração. Na imagem abaixo será possível ver o ícone verde “atualizar”:

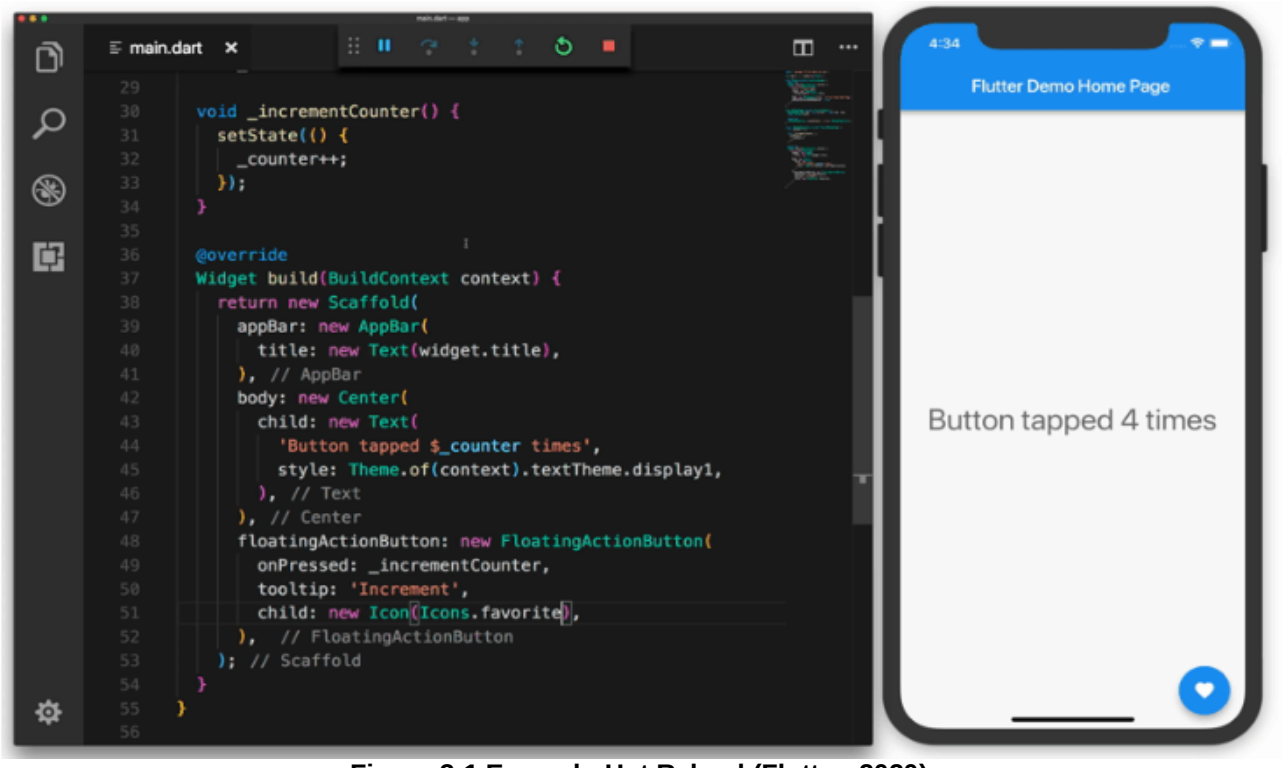

Figura 3-1 Exemplo Hot Reload (Flutter, 2020)

Caso o desenvolvedor esteja em modo depuração, poderá realizar todos os ajustes necessários no código e quando quiser ver na prática as alterações, poderá clicar no ícone “atualizar” e em seguida o Hot Reload faz o código-fonte Dart das bibliotecas transformar em arquivos do kernel e enviar para a VMDart do dispositivo móvel, fazendo assim todas as mudanças requisitadas e mostrando ao lado direto no simulador de smartphone os resultados obtidos.

Existem algumas limitações e casos que o Hot Reload não funcionará e será necessário reiniciar totalmente o Start da aplicação, um exemplo seria alterar o tipo de uma classe.

Com o desenvolvimento rápido é possível pintar o aplicativo em milissegundos com o Stateful Hot Reload, usando vários widgets repletos de personalizações e criar interfaces totalmente nativas em minutos. Com a UI (Interfaces de usuários) expressiva e flexível podemos enviar recursos rapidamente, focando nas experiências nativas de quem utilizará o aplicativo final, a arquitetura em camadas permite uma completa personalização que resulta em incríveis renderizações rápidas e com design expressivo e flexível. O desempenho nativo é um ponto forte do Flutter, os widgets conseguem ser executados nas mais críticas plataformas, como navegação, rolagem, ícones e fontes. O código do Flutter é compilado no código de máquina ARM nativo usando os compiladores nativos do Dart, fornecendo assim um desempenho nativo completo para IOS e Android. (Flutter, 2020).

3.1. ARQUITETURA FLUTTER

O Flutter é um SDK(Ferramenta de desenvolvimento de software) que permite criar apps bonitos, modernos e de alta performance, podendo desenvolver tanto para android como para ios e futuramente para web e desktop, tudo isso com apenas um código.

É uma ferramenta opensource desenvolvida pela Google e já possui mais de 100 contribuintes na comunidade, está crescendo exponencialmente e pode ser uma potência mundial.

Os designs gostam muito do flutter, pois podem ir além da imaginação e criar ótimas experiências de Android e iOS para seus clientes. Podendo fazer protótipos de telas bem rápidos.

Os desenvolvedores se aproveitam de uma linguagem fácil de usar que seria o Dart em conjunto com o Flutter, existe um rico conjunto de Widgets e um grande suporte a IDEs. (Melo, 2019)

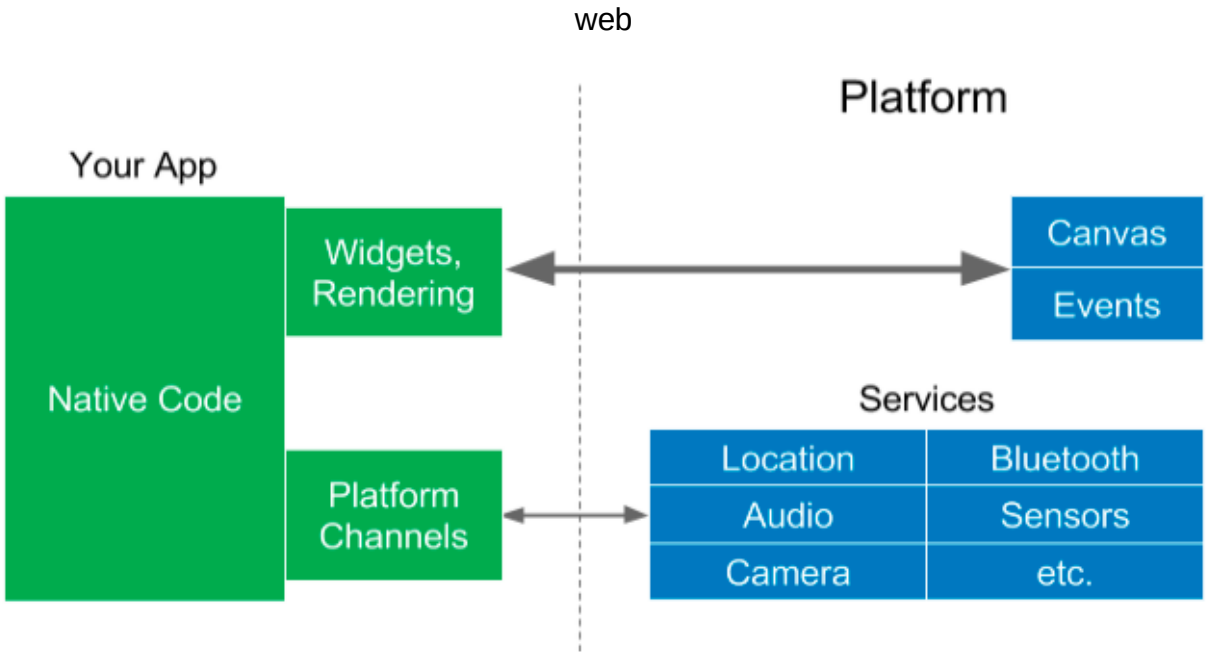

Figura 3-2 : Arquitetura Dart Flutter (Flutter, 2019)

Na imagem acima pode-se ver que não existe uma ponte entre a aplicação e a plataforma, firmando assim que é um desenvolvimento nativo, nas imagens abaixo podemos ver a diferença entre uma aplicação que possui ponte e uma sem ponte.

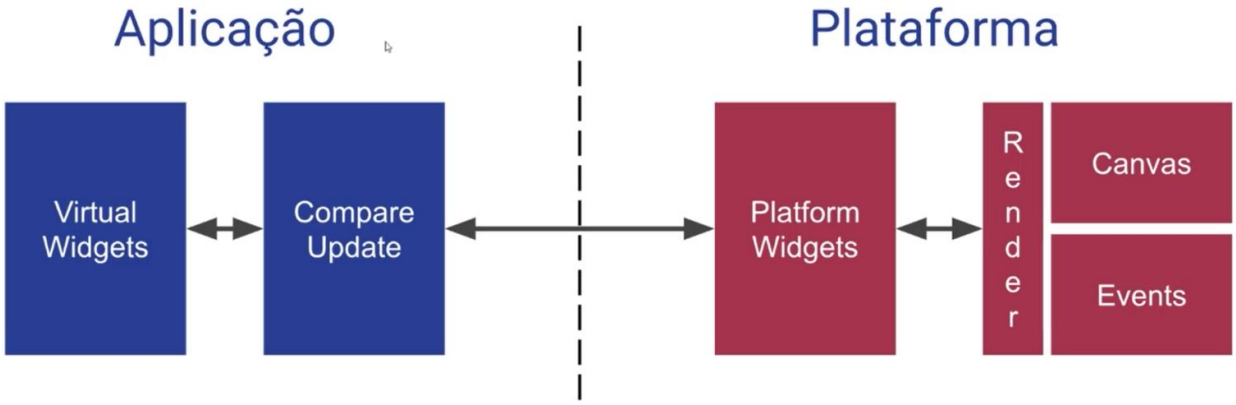

Figura 3-3: Aplicação com ponte (Flutter, 2019)

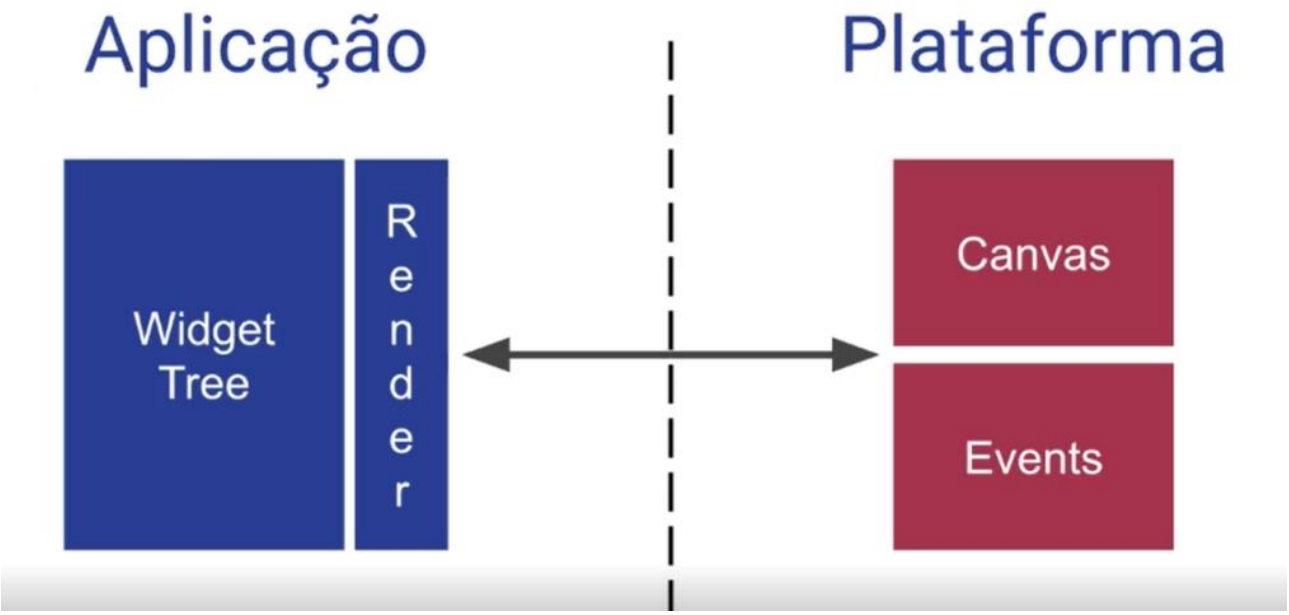

Figura 3-4: Aplicação sem ponte (Flutter, 2019)

Assim tem-se uma ligação direta, já que na aplicação sem ponte tem-se menos camadas em relação a aplicação com ponte. Em flutter é exatamente está imagem 3.4.

4. DESCRIÇÃO DO AMBIENTE DE DESENVOLVIMENTO

O aplicativo será desenvolvido utilizando a linguagem de programação DART junto com o SDK Flutter v1.12.13.

4.1. LINGUAGEM UTILIZADA

A linguagem escolhida para desenvolvimento deste aplicativo é a DART.

4.2. SDK UTILIZADO

O SDK escolhido para desenvolvimento deste aplicativo é o FLUTTER, que por sua vez está com alto crescimento de utilização no Brasil. (Ossada, 2019)

4.3. FERRAMENTAS PARA O DESENVOLVIMENTO

Aplicações para gerenciamento, desenvolvimento, manutenção e update do aplicativo.

4.3.1. ANDROID STUDIO VERSÃO 3.6.2 WINDOWS

Instale e execute seus aplicativos mais rapidamente do que com um dispositivo físico e simule diferentes configurações e recursos, incluindo o ARCore, a plataforma do Google para criar experiências de realidade aumentada. Escreva um código melhor, trabalhe mais rápido e seja mais produtivo com um editor de código inteligente que fornece a conclusão de código para as linguagens Kotlin, Java e C / C ++. As ferramentas de criação de perfil

internas fornecem estatísticas em tempo real da CPU, memória e atividade de rede do seu aplicativo. Identifique gargalos de desempenho registrando rastreamentos de métodos, inspecionando a pilha e alocações e veja as cargas úteis da rede de entrada e saída. (Harada, 2019).

4.3.2. Firebase

Com as funcionalidades do Firebase, como análises, bancos de dados, mensagens e relatórios de erros, você tem mais agilidade e pode se concentrar nos seus usuários. O Firebase foi criado sobre a infraestrutura do Google e faz o escalonamento automático, até mesmo para os maiores apps. Os produtos do Firebase funcionam muito bem sozinhos, mas têm um resultado ainda melhor quando compartilham dados e insights entre si. (Mendes, 2019).

5. FERRAMENTAS DE ANÁLISE

Foi utilizado duas ferramentas gratuitas para modelagem de como será o aplicativo, dentre eles o draw.io e DB Designer para orientar na modelagem do banco de dados.

5.1. DRAW.IO

Totalmente gratuito, o Draw.io é outra ferramenta competente e muito utilizada para a criação de diagramas UML (e de outros tipos). O site apresenta componentes que se adequam à criação de vários tipos de diagramas, como o de sequência, de atividades, de estados e o tradicional caso de uso.

É verdade que a variedade de componentes e recursos pode não ser tão variado assim. Porém, o acesso gratuito compensa a limitação, especialmente para quem não trabalha com diagramas UML profissionalmente e precisa acessar esse tipo de ferramenta apenas esporadicamente. (Furtado, 2013).

5.2. DB DESIGNER

O db designer tem interface do usuário que pode parecer simples, mas é completa e poderosa. Rápido e muito fácil de usar com todos os recursos que possa precisar. Reduz erros e economiza bastante tempo, importe um banco de dados existente ou comece do zero, ao final pode importar o script SQL para onde deseja utilizar. Possui vários designs, modelagem, modos de exibição.

5.3. RECURSOS NECESSÁRIOS

Recursos de Software

Windows 10 Home Single Language;

Android Studio 3.6.2;

Firebase;

Dart Flutter;

Draw io;

DB Designer.

Recursos Humanos

Um programador/estudante

Recursos de Hardware

Um notebook Acer F5-573G

5.3.1. ESPECIFICAÇÃO DE CUSTOS

Recursos Computacionais

Windows 10 Home	Valor = licenciado FEMA
Android Studio	Valor = Free
Draw io	Valor = Free
Firebase	Valor = Free
DB Designer	Valor =Free

Recursos Humanos

Um programador/estudante	Valor Diário = 200,00

Recursos Físicos

Um Notebook Acer	Valor = 3000,00

5.4. LEVANTAMENTO DOS REQUISITOS

Os requisitos para o desenvolvimento do aplicativo serão listados a seguir:

Usuário

Manter Usuário

Manter Alimentação

Login

Altura/Kilogramas

IMC (índice de massa corporal)

Treinos

Aeróbicos

5.5. DIAGRAMAS

O diagrama é uma linguagem de modelagem que se utiliza para criar uma documentação e analisar todos os requisitos de um software, podendo ser em plataforma web e desktop. Possui muitos itens para o desenvolvimento e implantação de um sistema básico e até mesmo avançados.

5.5.1. DIAGRAMAS DE CASO DE USO

Através do diagrama de caso de uso conseguimos observar de um modo geral todas as funcionalidades que o usuário poderá ter ao utilizar o aplicativo, sendo elas apresentada na imagem abaixo:

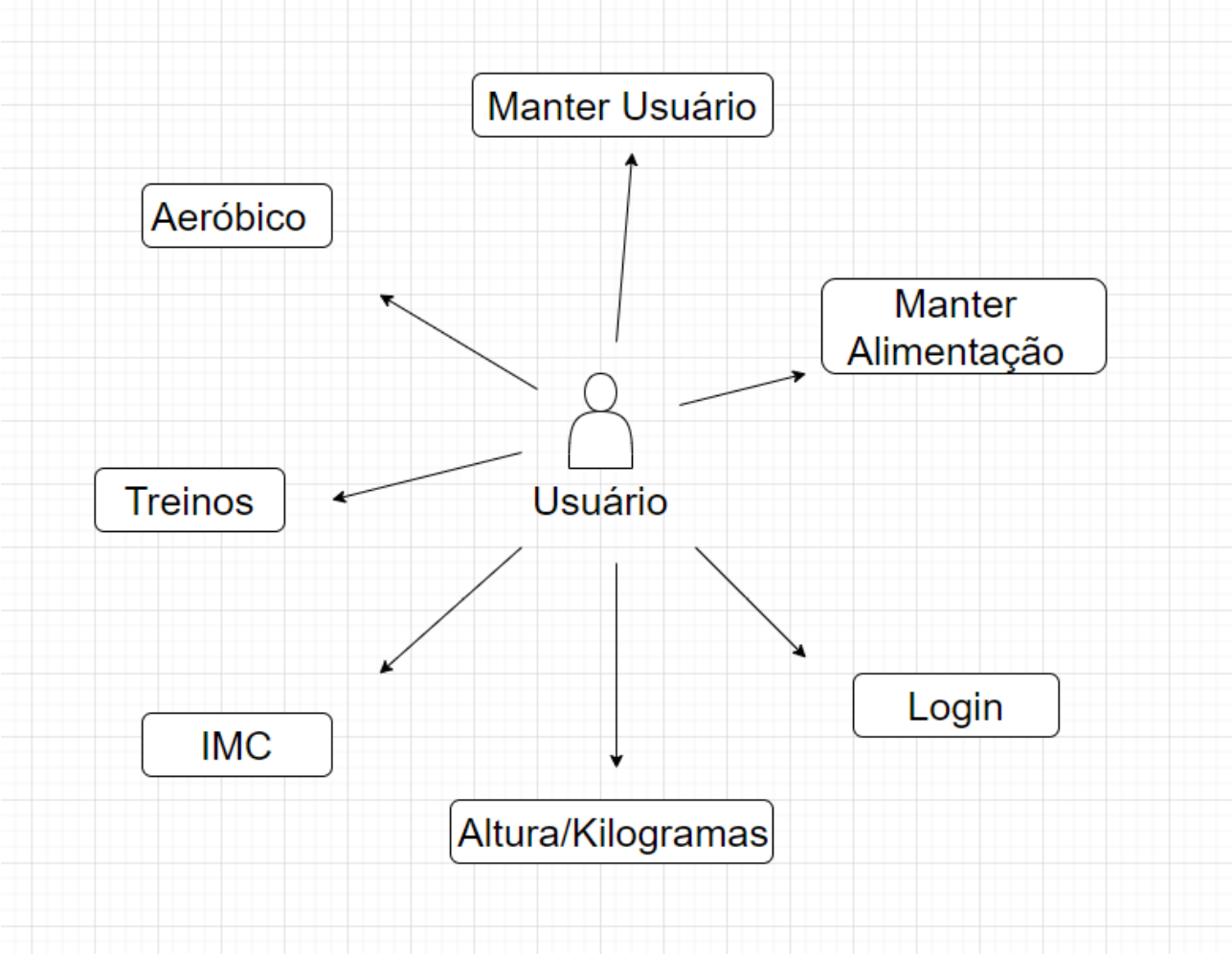

Figura 5-1: Diagrama de Casos de Uso Geral (Feita pelo autor)

A parte de manter usuário é focada em cadastro e gerenciamento dos dados pessoas de quem utilizará o aplicativo. Manter alimentação é para administrar os alimentos da dieta escolhida. O treino mostrará todos os exercícios que o usuário deverá fazer diariamente. O IMC é demonstrado através de um cálculo científico que é capaz de aponta se o ser humano está acima, abaixo, ou no peso ideal. O *login* é o e-mail e senha do praticante. A altura e quilogramas também guardará os dados do praticante.

6. DESENVOLVIMENTO DO APLICATIVO

6.1. Protótipos de tela com design

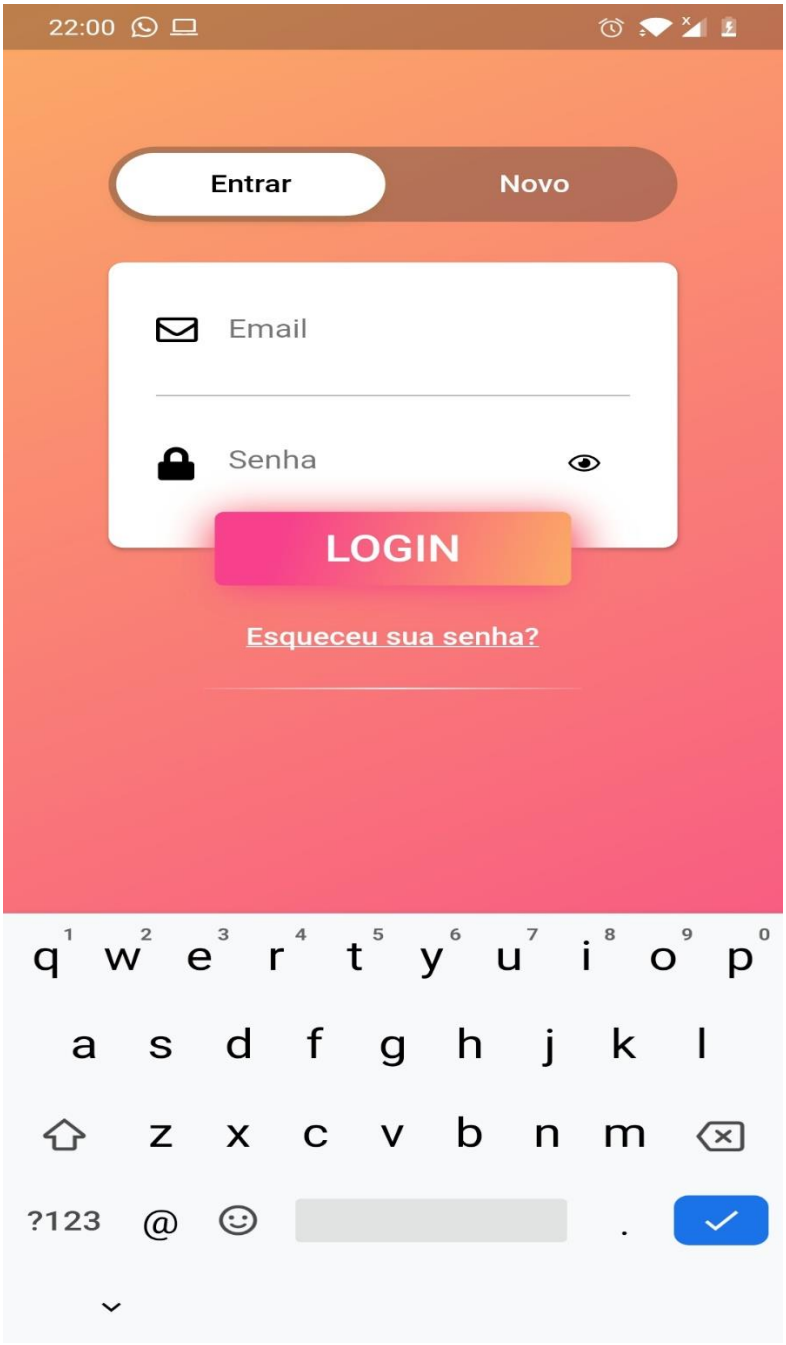

Figura 6-1: Tela de Login (Feita pelo autor)

Na imagem acima tem-se a tela de *login,* onde o usuário que já estiver cadastrado no aplicativo, pode utilizar o seu *email* e senha. A validação dos campos é feita através do

firebase citado no capítulo 4.3.2, o *firebase* verifica se os campos existem e se estão exatamente como foi cadastrado anteriormente.

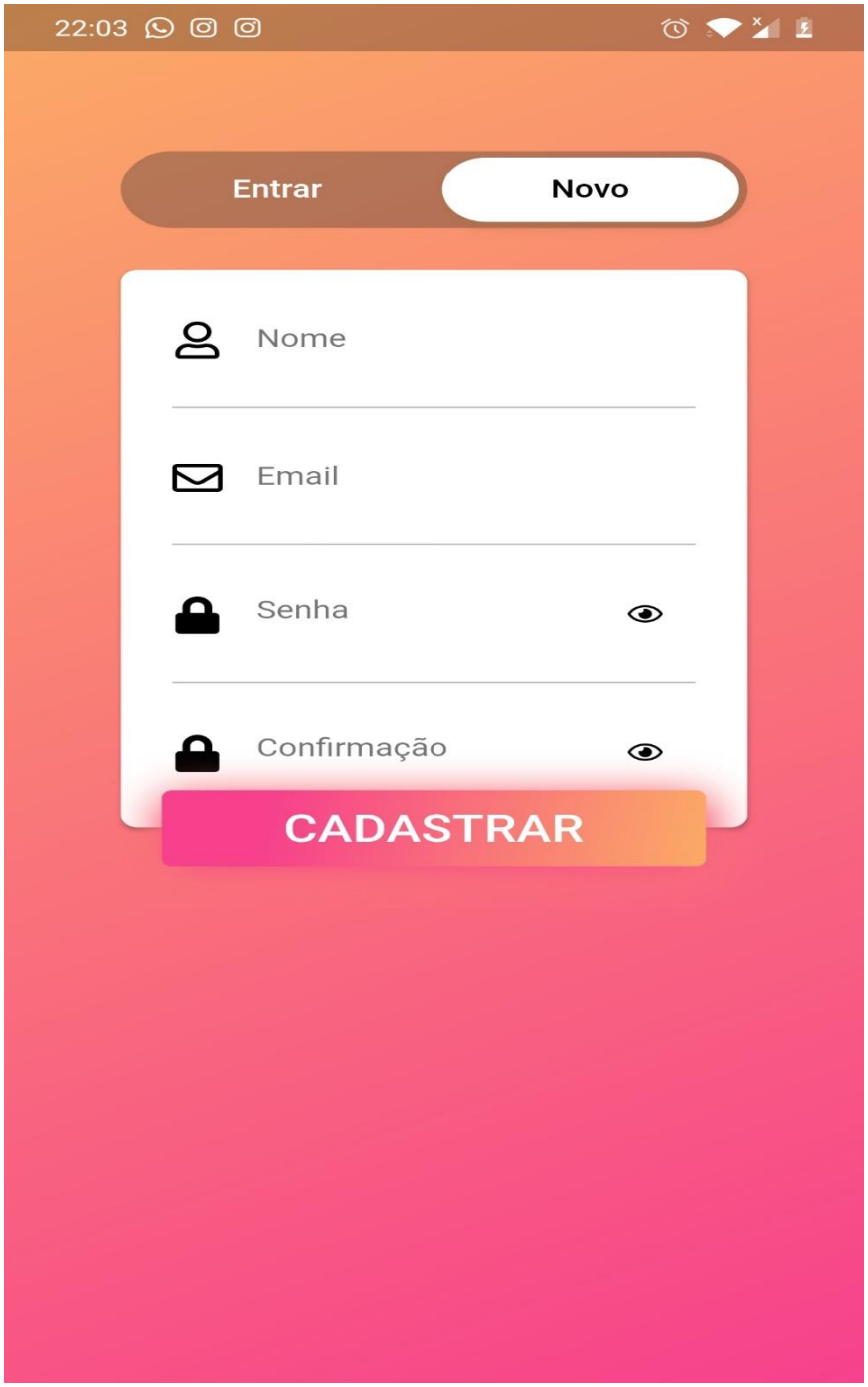

Figura 6-2: Tela de Cadastro (Feita pelo Autor)

Na imagem acima pode-se inserir o nome, *e-mail* e senha do usuário. Para entrar nesta tela basta arrastar a tela para a esquerda, e o título da página muda de ‘entrar’ para ‘novo’,

habilitando assim as configurações de cadastro. As validações de *e-mail* e senha são configuradas pelo próprio *firebase*. A senha deve possuir no mínimo 6 dígitos, podendo ser letras, números e caracteres especiais. Já no *e-mail* é necessário ter o símbolo ‘@’.

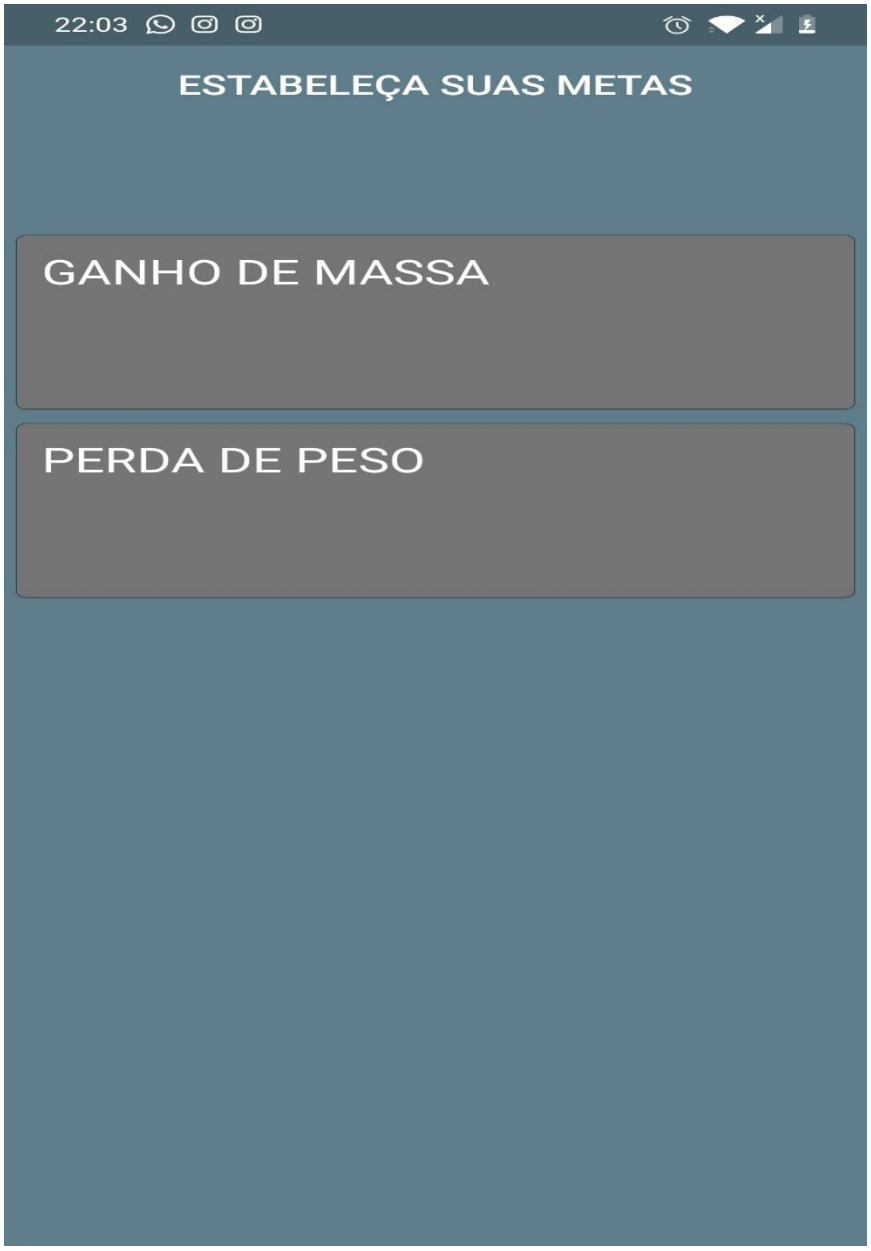

Figura 6-3: Tela após login (Feita pelo autor)

A Figura 6-3 descreve o primeiro ambiente após o usuário inserir os dados de login e a validação ter sido confirmada como verdadeira. Nesta etapa começamos a interagir mais com quem está utilizando o aplicativo, já que inicia a relação com as metas estipuladas que pode ser diferente de pessoa para pessoa.

Figura 6-4: Tela Tipo de Corpo (Feita pelo autor)

A Figura 6-4 o usuário do aplicativo deve selecionar o tipo de corpo que está atualmente, e isso afetará diretamente no tipo de treino, pois determinado corpo é compatível com determinado treino.

Figura 6-5: Tela Especificar Meta (Feita pelo autor)

A Figura 6-5 força o praticamente a especificar ainda mais o que ele realmente deseja, para que tenha melhores resultados calculados pelas regras de negócio, que são baseadas em todas as escolhas feitas. O usuário pode selecionar reduzido e os treinos serão mais leves, o volumoso será um pouco mais intenso e o bem volumoso com intensidade máxima.

Figura 6-6: Tela Condicionamento (Feita pelo autor)

Na Figura 6-6 o praticante escolhe qual é o seu condicionamento físico atual, podendo ser iniciante, intermediário ou avançado. Essa escolha afeta diretamente no treino que será gerado ao usuário, pois é uma informação de extrema importância.

6.1.1. Codificação

Uma determinada aplicação sempre possui um estado, sendo assim, esse estado é quem determina a existência, qualidade e condição do programa. Dois importantes modelos são *stateless* e *stateful.*

Stateless é usado quando uma aplicação tem recursos independentes, onde toda a informação utilizada em uma sessão anterior não será armazenada para a futura sessão, as aplicações desse modelo sempre têm algum serviço ou função que utiliza a rede para entrega de informações que precisam ser processadas e executadas em um curto período de tempo. Um exemplo de *stateless* são pesquisas *online*, onde o cliente necessita de uma resposta rápida. Assim se no caminho de busca acontecer algum erro, é necessário começar novamente para chegar na resposta.

Stateful é usado quando uma aplicação precisa de dados de uma sessão anterior, pode ser serviços de bancos digitais ou até mesmo caixa de mensagens. Neste modelo caso ocorra alguma interrupção ou erro na hora de determinada solicitação, é possível voltar naquele exato momento de onde parou.

A imagem abaixo apresenta a utilização do modelo *StateLess:*

```
Run | Debug
void main() => runApp(new MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Academia Login',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: new LoginPage(),
    ); // MaterialApp
  }
}
```

Figura 6-7: Tela StateLess (Feita pelo autor)

Pode-se observar na figura 6-7 que o a primeira classe do aplicativo se chama *main* e ao iniciar a aplicação a primeira tela requisitada é a chamada de *LoginPage.* Essa tela é a demostrada nas Figuras 6-1 e 6-2.

```
onPressed: () {
  Usuario usuario = new Usuario();
  usuario.email = loginEmailController.text;
  usuario.senha = loginPasswordController.text;
  usuarioController.realizarLoginUsuario(usuario)
    .then((value){
      Navigator.of(context).push(MaterialPageRoute(builder: (_)=> Metas()));
    })
    .catchError((onError){
      Toast.show("Erro ao realizar Login", context, duration: Toast.LENGTH_LONG,gravity: Toast.BOTTOM);
    });
}), // MaterialButton
```

Figura 6-8: Tela Requisição Metas (Feita pelo autor)

A imagem acima demonstra o *backend* da Figura 6-1, e se o *email* e senha estiver correto, é requisitada a próxima tela, demonstrada na Figura 6-3.

```
GestureDetector(
  onTap: (){
    Caracteristica caracteristica = new Caracteristica();
    caracteristica.chave = "meta";
    caracteristica.valor = "GANHOMASSA";
    usuario.adicionarCaracteristica(caracteristica);
    caracteristicaController.salvarCaracteristica(usuario)
     .then((value){
       Navigator.of(context).push(MaterialPageRoute(builder: (_)=> TipoDeCorpo()));
     }).catchError((e){
       print('------------------------');
       print(e);
     });
```

Figura 6-9: Tela Requisição Tipo Corpo (Feita pelo autor)

A imagem acima podemos ver que ao selecionar a meta da Figura 6-3, é requisitada a Tela 6-4, para que o usuário selecione o tipo de corpo.

```
GestureDetector(
    onTap: (){
     Caracteristica caracteristica = new Caracteristica();
     caracteristica.chave = "tipoDeCorpo";
     caracteristica.valor = "MAISPESADO";
     usuario.adicionarCaracteristica(caracteristica);
     caracteristicaController.salvarCaracteristica(usuario)
      .then((value) => Navigator.of(context).push(MaterialPageRoute(builder: (_)=> MetaCorpo())));
    },
```

Figura 6-10: Tela Requisição Meta Corpo (Feita pelo autor)

A imagem acima pode-se observar que quando o usuário escolhe o tipo de corpo, é requisitada a tela da Figura 6-5.

```
GestureDetector(
  onTap: (){
    Caracteristica caracteristica = new Caracteristica();
    caracteristica.chave = "metaCorpo";
    caracteristica.valor = "REDUZIDO";
    usuario.adicionarCaracteristica(caracteristica);
    caracteristicaController.salvarCaracteristica(usuario)
     .then((value) => Navigator.of(context).push(MaterialPageRoute(builder: (_)=> Condicionamento())));
  },
```

Figura 6-11: Tela Requisição Condicionamento (Feita pelo autor)

A Figura 6-11 demonstra que quando o praticante escolhe a meta do corpo da Figura 6-5 o programa chama a tela da Figura 6-6.

7. CONCLUSÃO

Este trabalho teve o intuito de demonstrar como as tecnologias de desenvolvimento estão crescendo no Brasil, com foco na linguagem de programação *Dart* e o *kit* de desenvolvimento de interface chamado *Flutter*. Percebe-se como essa tecnologia ajuda muito no cotidiano do programador, desde a parte que é necessário conectar a um banco de dados em nuvem até o momento do estilo e tela em que o cliente estará totalmente em contato.

O banco de dados *firebase* se mostrou bastante estável durante o projeto, enquanto que os emuladores funcionaram de acordo com a *ex*pectativa. Proporcionou-se a facilidade de utilizar a aplicação de testes tanto nos emuladores *Android Studi*o e *GenyMotion,* e para uma experiência mais realista, conectando apenas o celular pelo cabo USB foi possível ver em tempo real o funcionamento da aplicação.

Os praticantes de atividade física irão ingressar em um novo modelo de treinos, onde a pessoa tem na palma de sua mão tudo o que precisa, perguntando sempre para um profissional capacitado se os movimentos estão corretos. A aplicação não ficou pesada em relação a outras do mesmo ambiente, e por isso a maioria dos celulares serão compatíveis e funcionará com sucesso.

Assim que a primeira versão do aplicativo estiver comtemplada, será adicionada ao *Google Play Store* para que os usuários possam utilizá-lo.

8. REFERÊNCIAS

ANDROID, Android Studio em <https://developer.android.com/studio> Acesso em 06 de março de 2020.

CAVALCANTI, Eric. **Flutter - uma breve introdução**, Disponível em <https://medium.com/> dia 11 de março de 2018. Acesso em 07 de março de 2020.

DANTAS, Lud Lucena, **DIFERENTES ABORDAGENS DIETETICAS PARA PRATICANTES DE ATIVIDADE FISICA VOLTADA PARA O EMAGRECIMENTO,** Disponível em <https://monografias.ufrn.br/> fevereiro de 2017. Acesso em 27 de agosto de 2020.

DART, dart dev em <https://dart.dev/> Acesso em 05 de março de 2020.

DRAW, draw.io em <https://app.diagrams.net/> Acesso em 07 de março de 2020.

FLUTTER, flutte dev em <https://flutter.dev/docs> Acesso em 04 de março de 2020.

FIREBASE, Firebase em <https://firebase.google.com/> Acesso em 07 de março de 2020.

FURTADO, Teresa. **Draw.io é ótimo para criar gráficos e desenhos sem baixar nada.** Disponível em < https://www.techtudo.com.br/tudo-sobre/drawio.html> Acesso em 04 de Junho de 2020.

Google AdSense Disponivel em <https://ads.google.com/intl/pt-BR_br/getstarted> Acesso em 05 de março de 2020.

HARADA, Eduardo. **O que é o Android Studio, ferramenta criada para desenvolver apps mobile.** Disponível em <https://www.tecmundo.com.br/software/146361-o-android-studio-ferramenta-criada-desenvolver-apps-mobile.htm> Acesso em 17 de Julho de 2020.

MELO, Rubens. **Flutter para iniciantes.** Disponível em <https://speakerdeck.com/rubensdemelo/flutter-para-iniciantes> Acesso em 10 de Setembro de 2020.

MENDES, Evelyn. **Firebase.** Disponível em < https://emendes.com/2019/11/02/firebase/> Acesso em 24 de Julho de 2020.

OSSADA, Toshi. **Por que Flutter?** Disponível em <https://medium.com/toshiossada/por-que-flutter-8f17cc2bb02e> Acesso em 11 de Setembro de 2020.

SABA, Fabio , **A importância da atividade física para a sociedade e o surgimento das academias de ginástica** Disponível em <http://periodicos.ufpel.edu.br/ojs2/index.php/RBAFS/article/download/1085/1268> Acesso em 03 de Março de 2020.

SILVA, Carlos L. A. **Frameworks de servidor HTTP para Dart.** Disponível em <https://www.codigofonte.com.br/artigos/frameworks-de-servidor-http-para-dart> Acesso em 07 de Setembro de 2020.

VARADY, Krista A. et al. **Alternate day fasting for weight loss in normal weight and overweight subjects: a randomized controlled trial**. Nutrition journal, v. 12, n. 1, p. 146, 2013.