



U N I V E R S I D A D E
LUSÓFONA

Relatório de Engenharia de Software

Projeto II

João Fernandes a22003504

Ricardo Machado a22002860

Índice

1. Introdução	4
1.1 Contexto	4
1.2 Justificação	4
1.3 Desenvolvimento	4
1.4 Objetivos	4
1.5 Stakeholders	5
1.6 Análise SWOT	5
1.7 Gestão de Projeto	6
1.7.1 Cronologia Projeto	6
1.7.2 Tarefas e seu desenvolvedor	6
2. Metodologia em Cascata	6
2.1 Fases da Metodologia	7
2.1.1 Análise de Requisitos	7
2.1.2 Design	7
2.1.3 Implementação	7
2.1.4 Testes	7
2.1.5 Manutenção	8
3. Requisitos e análise de software	8
3.1 Atores e Casos de uso	8
3.1.1 História de utilizador	9
3.2 Diagramas de UML	10
3.2.1 Diagrama de Classe	10
3.2.2 Diagrama de Componentes	11
3.3 Requisito funcional	11
3.3.1 Sistema de Autenticação	11
3.3.2 Visualização de produtos da loja	11
3.3.3 Adicionar/Remover produtos ao carrinho de compra	12
3.3.4 Marcação de serviço	12
3.3.5 Visualizar catálogo de cortes de cabelo	12
3.3.6 Adicionar e visualizar avaliações	13
3.4 Requisito não funcional	13
4. Arquitetura da aplicação	14
4.1 Plataforma e ferramentas utilizadas	14
4.2 Arquitetura em camadas	14
4.3 Realtime Database do Firebase	14
4.4 Autenticação do Firebase	15
4.5 Firestore Database do Firebase	15
4.6 Modelo entidade associação	16
5. Implementação	17
5.1 Tecnologias utilizadas	17
5.2 Interface	18

5.2.1 Login	18
5.2.2 Registo	18
5.2.3 Página Inicial	19
5.2.4 Loja de produtos	20
5.2.5 Catálogo de cortes de cabelo	20
5.2.6 Marcação do serviço	21
5.2.7 Avaliação	22
5.3 Funcionalidades da aplicação	22
5.3.1 Autenticação do utilizador	22
5.3.2 Página Inicial	22
5.3.3 Loja de produtos	23
5.3.4 Catálogo de cortes de cabelo	23
5.3.5 Marcação de serviço	23
5.3.6 Avaliação	23
6. Construção de software e Testes	23
6.1 Codificação	23
6.1.1 Implementação do Firestore Database do Firebase	23
6.1.2 Carrinho de produtos	24
6.1.3 Avaliação	26
6.1.4 Marcação de Serviço	28
6.2 Testes	29
6.2.1 Teste de Software	30
6.2.1.1 Teste unidade	30
6.2.1.1.1 Resultado	31
6.2.1.2 Teste integração	31
6.2.1.2.1 Resultado	31
6.2.2 Análise de Feedback da Aplicação: Resultados do Questionário	31
6.2.2.1 Introdução	31
6.2.2.2 Resultados	31
6.2.2.3 Conclusão	32
6.2.3 Entrevista	33
6.2.3.1 Barbearia Mota	33
6.2.3.1.1 Perguntas e respostas	33
6.2.3.1.2 Conclusões	34
7. Conclusão	35
8. Referências	36

1. Introdução

1.1 Contexto

A aplicação foi desenvolvida no contexto em que a maioria das pessoas utiliza dispositivos móveis para realizar tarefas diárias, e os serviços prestados por barbearias são bastante requisitados. Dessa forma, a "Barbearia Lusófona" pretende oferecer aos seus clientes a conveniência de marcar cortes de cabelo e adquirir produtos da barbearia diretamente através da aplicação, além de disponibilizar um catálogo de cortes para ajudar na escolha de um novo visual.

1.2 Justificação

A "Barbearia Lusófona" pretende oferecer aos seus clientes a conveniência de marcar cortes de cabelo e adquirir produtos da barbearia diretamente através da aplicação. Além disso, a aplicação também disponibiliza um catálogo de cortes para ajudar na escolha de um novo visual. Essa conveniência oferecida pela aplicação pode ajudar a "Barbearia Lusófona" a se destacar em relação a outras barbearias que ainda não oferecem serviços móveis.

1.3 Desenvolvimento

O desenvolvimento da aplicação foi desafiador, envolvendo a análise de diversos requisitos funcionais e não funcionais, a escolha das tecnologias mais adequadas para o desenvolvimento e implementação e a realização de testes para garantir a qualidade e usabilidade da aplicação.

1.4 Objetivos

O relatório abordará os requisitos funcionais e não funcionais, a metodologia utilizada no desenvolvimento da aplicação, a arquitetura e implementação da solução, os testes realizados e os resultados obtidos, bem como as conclusões e lições aprendidas ao longo do projeto. O relatório também tem como objetivo ser útil para outros desenvolvedores que estejam a trabalhar em projetos semelhantes.

1.5 Stakeholders

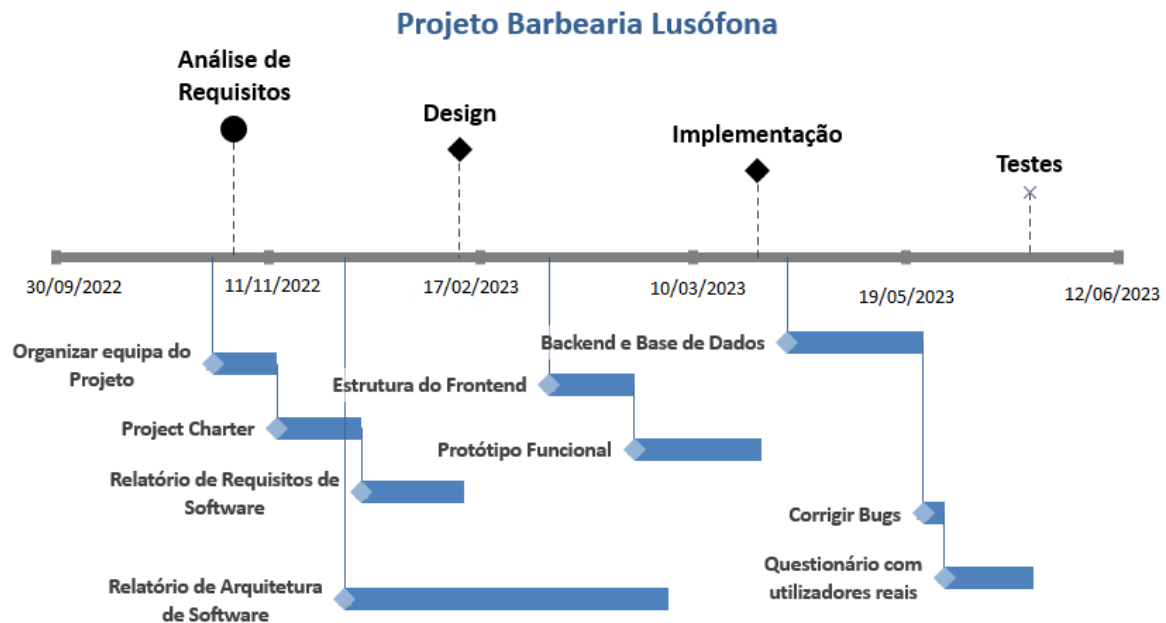
Os stakeholders incluem a equipa de desenvolvimento, os utilizadores e a barbearia parceira que testará a aplicação e fornecerá feedback sobre a sua experiência. O objetivo dessas parcerias é aprimorar a aplicação com base nos comentários e sugestões dos utilizadores finais e, assim, melhorar sua usabilidade e satisfação.

1.6 Análise SWOT

Força	Fraqueza	Oportunidade	Ameaça
Facilita o agendamento de horários	Dependência da tecnologia e da conectividade à 'internet'	Aumento da satisfação do cliente e fidelidade à barbearia	Concorrência de outras aplicações ou sistemas de gestão de barbearias
Melhora a eficiência e eficácia da barbearia		Atração de novos clientes através da promoção da marca e dos serviços da barbearia	Riscos de segurança cibernética e proteção de dados
Promove a marca e os serviços da barbearia			Dependência de terceiros para manter e atualizar a aplicação

1.7 Gestão de Projeto

1.7.1 Cronologia Projeto



1.7.2 Tarefas e seu desenvolvedor

Início	Fim	Nome da Tarefa	Desenvolvedor
30/09/2022	21/10/2022	Project Charter	João Fernandes
22/10/2022	11/11/2022	Relatório de Requisitos de Software	João Fernandes e Ricardo Machado
12/11/2022	16/12/2022	Relatório de Arquitetura de Software	João Fernandes e Ricardo Machado
20/02/2023	27/02/2023	Estrutura do Frontend	Ricardo Machado
28/02/2023	15/03/2023	Protótipo Funcional	João Fernandes
16/03/2023	20/05/2023	Backend e Base de Dados	João Fernandes e Ricardo Machado
21/05/2023	25/05/2023	Questionário com utilizadores	João Fernandes
27/05/2023	04/06/2023	Testes	João Fernandes e Ricardo Machado
05/06/2023	12/06/2023	Relatório de Engenharia de Software	João Fernandes e Ricardo Machado

2. Metodologia em Cascata

Escolhemos a metodologia em cascata devido a algumas razões específicas. A abordagem sequencial da cascata proporcionou uma estrutura lógica e clara para o projeto, facilitando o acompanhamento das etapas. Além disso, esta metodologia alinhou-se com a natureza teórica do trabalho, permitindo uma análise aprofundada de cada fase. A

disponibilidade de tempo também influenciou a escolha, uma vez que a Cascata ajudou a gerir prazos de forma mais previsível. Além disso, a Cascata favoreceu a produção de documentação completa e detalhada, essencial para a avaliação académica. Em resumo, a metodologia em cascata foi escolhida devido à sua estrutura sequencial, adequação à natureza teórica do projeto, gestão de prazos e importância da documentação.

2.1 Fases da Metodologia

2.1.1 Análise de Requisitos

Na primeira fase da metodologia, a análise de requisitos foi realizada. Foram identificadas as necessidades e expectativas do cliente em relação à aplicação e foram levantados requisitos funcionais e não funcionais. Esses requisitos foram documentados em um documento de especificação de requisitos.

2.1.2 Design

Em seguida, foi realizada a fase de design, onde a arquitetura da aplicação foi definida e as funcionalidades foram detalhadas. Foram criados diagramas de classes, diagramas de sequência, diagramas de casos de uso e outros artefatos de design para representar a estrutura e o comportamento da aplicação.

2.1.3 Implementação

Com o design completo, a equipa de desenvolvimento iniciou a fase de implementação, que consistiu na codificação das funcionalidades definidas na fase anterior. Foram utilizadas tecnologias e ferramentas específicas para o desenvolvimento de aplicações Android, como o Android Studio e o Java. Durante a implementação, foram adotadas práticas de programação orientada a objetos e boas práticas de programação.

2.1.4 Testes

Após a implementação, foi realizada a fase de testes, onde foram identificados e corrigidos bugs e erros de funcionamento da aplicação. Foram realizados testes de unidade, testes de integração e testes de aceitação para validar o funcionamento da aplicação em diferentes cenários de uso.

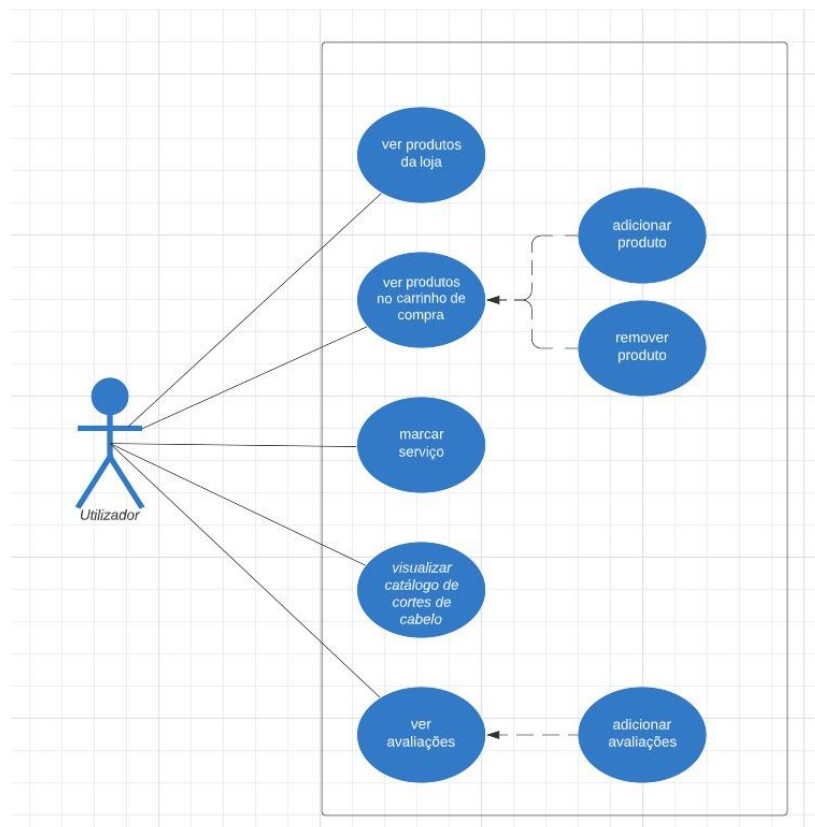
2.1.5 Manutenção

Por fim, a aplicação será entregue a um cliente interessado no projeto e será realizada a fase de manutenção, onde correções de erros e melhorias serão realizadas ao longo do tempo.

3. Requisitos e análise de software

3.1 Atores e Casos de uso

Os casos de utilização são uma técnica de modelação de requisitos de software que descreve a forma como um sistema é utilizado pelos utilizadores. Representam uma descrição detalhada de um ou mais cenários que mostram como um utilizador ou ator interage com um sistema para realizar uma tarefa específica. O ator presente é o utilizador da aplicação. Tem permissões para aceder à loja virtual e adicionar produtos no carrinho de compras, pode ver os produtos no carrinho de compras e tem a possibilidade de remover produtos do mesmo, consegue marcar o serviço de corte com a escolha de um dos profissionais que estiverem disponíveis no horário, consegue aceder ao catálogo de cortes de cabelo e por fim, consegue visualizar as avaliações e posteriormente, se utilizar o serviço, também pode deixar a sua avaliação.



3.1.1 História de utilizador

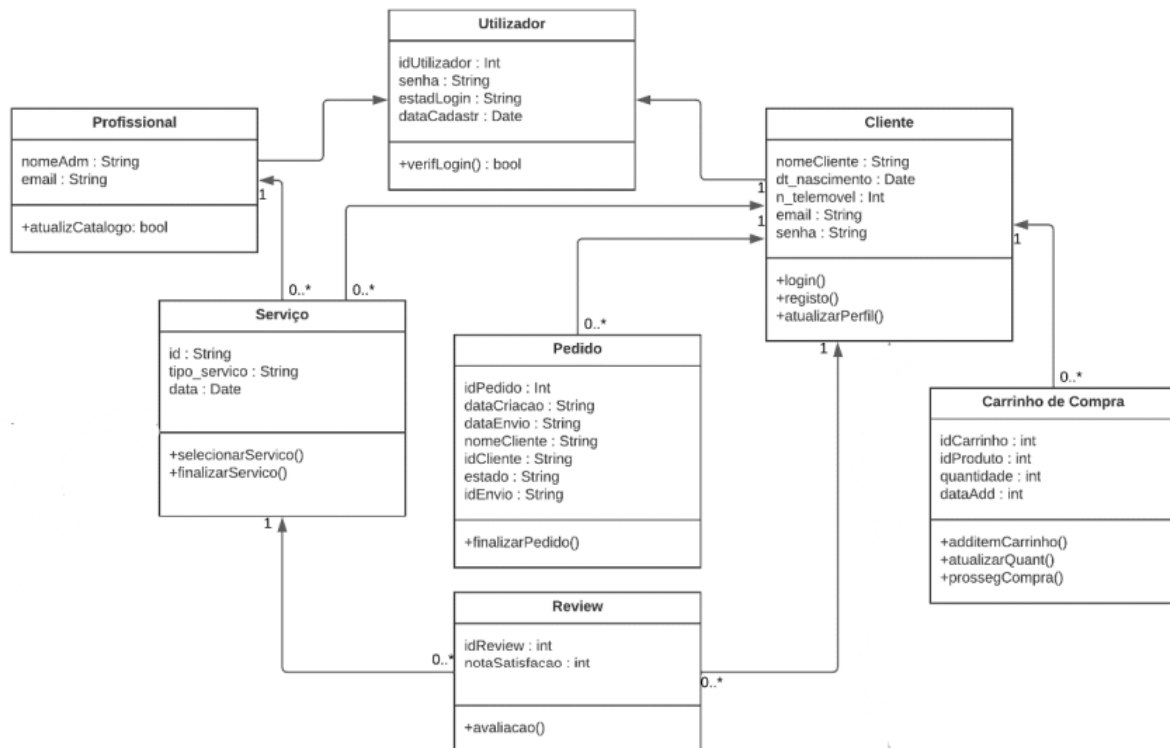
“Alberto Gomes, um homem casado de 42 anos, sentia que estava na altura de ter um novo visual e decidiu procurar por uma barbearia diferente na sua área na internet. Foi então que descobriu a Barbearia Lusófona e percebeu que tinha uma aplicação disponível para download. Ao abrir a aplicação, como novo utilizador, teve de criar uma conta, introduzindo o seu nome de utilizador, endereço de email, palavra-passe e número de telefone. Depois de confirmar a sua conta, fez login na aplicação e deparou-se com um menu inicial muito intuitivo, fácil de perceber e dividido em quatro grandes botões: "Loja", "Catalogo de Cortes", "Marcação de Serviço" e "Avaliação". Sentindo curiosidade em relação aos produtos disponíveis na barbearia e aos preços praticados, Alberto decidiu dar uma vista de olhos na loja, e em seguida explorou o catálogo de cortes, algo que nunca tinha visto noutra aplicação de barbearia. Achou criativo e prático ter acesso a uma variedade de cortes de cabelo disponíveis e, depois de gostar de um ou dois cortes, foi à página de avaliações, um recurso que também não é tão comum em aplicações do género, e considerou interessante ler as opiniões de outros utilizadores sobre o serviço e os cortes de cabelo da barbearia.

Como ficou satisfeito com o que viu na aplicação, Alberto decidiu experimentar a barbearia e dirigiu-se à secção de marcação de serviços, que achou muito fácil e intuitiva. Bastou selecionar o barbeiro, a data e a hora pretendida para o serviço e, em seguida, recebeu uma mensagem de confirmação no seu telemóvel. Agora, está ansioso e entusiasmado com o seu novo visual, mas agradeceu à aplicação por ter sido o catalisador que o encorajou a experimentar a Barbearia Lusófona”.

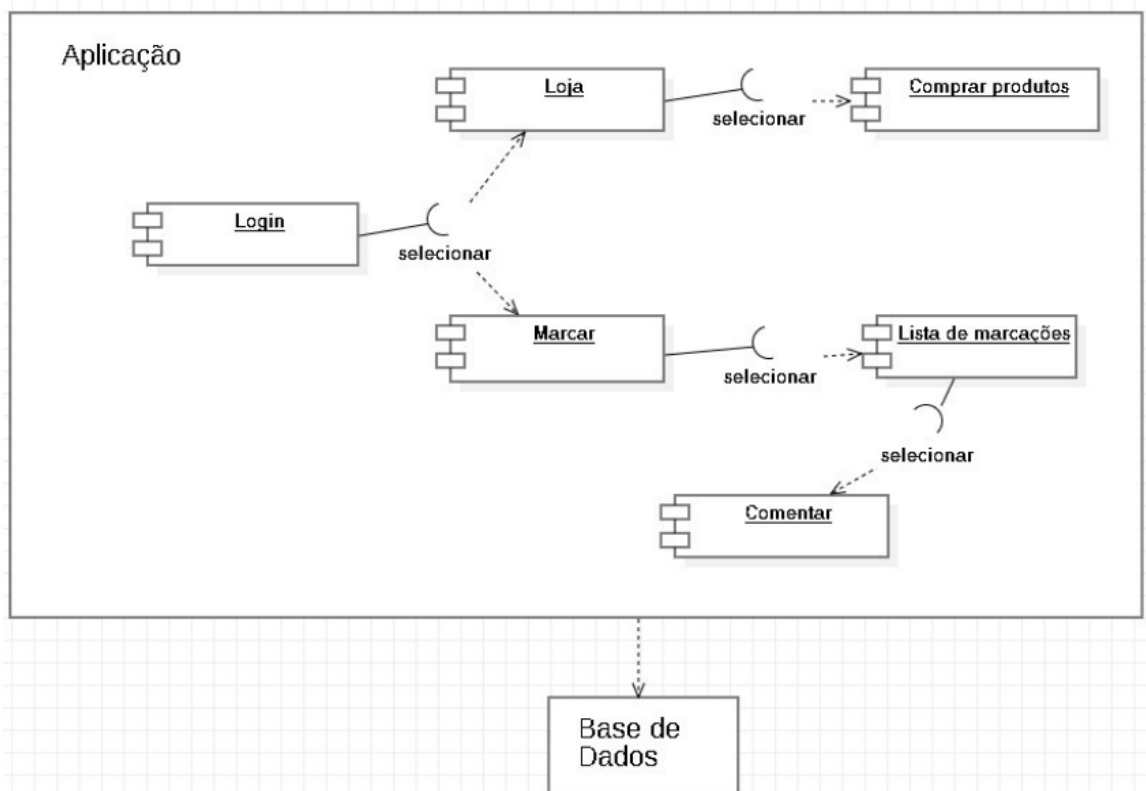
3.2 Diagramas de UML

Os diagramas de UML (Unified Modeling Language) são uma linguagem visual amplamente utilizada na engenharia de software. Um dos diagramas mais comuns é o diagrama de Classes. O diagrama de Classes representa a estrutura estática de um sistema, mostrando as classes, os seus atributos, métodos e relacionamentos. É útil para o design e a modelação de sistemas orientados a objetos. Este diagrama é uma ferramenta essencial na análise, design e documentação de sistemas, permitindo uma comunicação eficaz entre os intervenientes no desenvolvimento de software.

3.2.1 Diagrama de Classe



3.2.2 Diagrama de Componentes



3.3 Requisito funcional

3.3.1 Sistema de Autenticação

Requisito funcional	Sistema de autenticação
Descrição	Descreve a ação de autenticação no software para ser possível interagir com a aplicação
Atores	Utilizador
Pré-requisito	O ator deve ter email e palavra passe

3.3.2 Visualização de produtos da loja

Requisito funcional	Visualizar produtos da loja
Descrição	O software deve permitir que o cliente visualize os produtos disponíveis na loja da barbearia
Atores	Utilizador
Pré-requisito	O cliente deve estar autenticado no sistema

3.3.3 Adicionar/Remover produtos ao carrinho de compra

Requisito funcional	Adicionar/Remover produtos ao carrinho
Descrição	O software deve permitir que o cliente adicione e remova produtos ao carrinho de compra
Atores	Utilizador
Pré-requisito	O cliente deve estar autenticado no sistema e ter acesso aos produtos disponíveis na loja

3.3.4 Marcação de serviço

Requisito funcional	Marcação de serviço
Descrição	O software deve permitir que o cliente agende cortes de cabelo na barbearia
Atores	Utilizador
Pré-requisito	O cliente deve estar autenticado no sistema e ter acesso aos horários disponíveis para agendar

3.3.5 Visualizar catálogo de cortes de cabelo

Requisito funcional	Visualizar catálogo
Descrição	O software deve permitir que o cliente visualize o catálogo de cortes de cabelo disponíveis na barbearia
Atores	Utilizador
Pré-requisito	O cliente deve estar autenticado no sistema

3.3.6 Adicionar e visualizar avaliações

Requisito funcional	Adicionar e visualizar avaliações
Descrição	O software deve permitir que o cliente adicione e visualize avaliações de cortes ou da aplicação
Atores	Utilizador
Pré-requisito	O cliente deve estar autenticado no sistema

3.4 Requisito não funcional

Os requisitos não funcionais descrevem as características não relacionadas diretamente com as funcionalidades do software, mas sim com aspetos como desempenho, segurança e usabilidade. Implementamos autenticação e autorização, pois são fundamentais para garantir que apenas utilizadores autorizados possam aceder à aplicação e seus recursos. Utilizamos boas práticas de programação, como validação de entrada de dados e priorizamos a usabilidade, para garantir que os utilizadores possam utilizar a aplicação de forma fácil e eficiente, ao utilizar interfaces claras e intuitivas, minimizar a quantidade de cliques necessários para realizar uma tarefa e realizamos testes de usabilidade com utilizadores reais para identificar e corrigir problemas de usabilidade na aplicação.

Q

4. Arquitetura da aplicação

4.1 Plataforma e ferramentas utilizadas

A aplicação "Barbearia Lusófona" foi desenvolvida utilizando a plataforma Android e a base de dados Firebase. O Android Studio foi a ferramenta utilizada para o desenvolvimento da aplicação e o Firebase foi utilizado para armazenar e gerir os dados da aplicação.

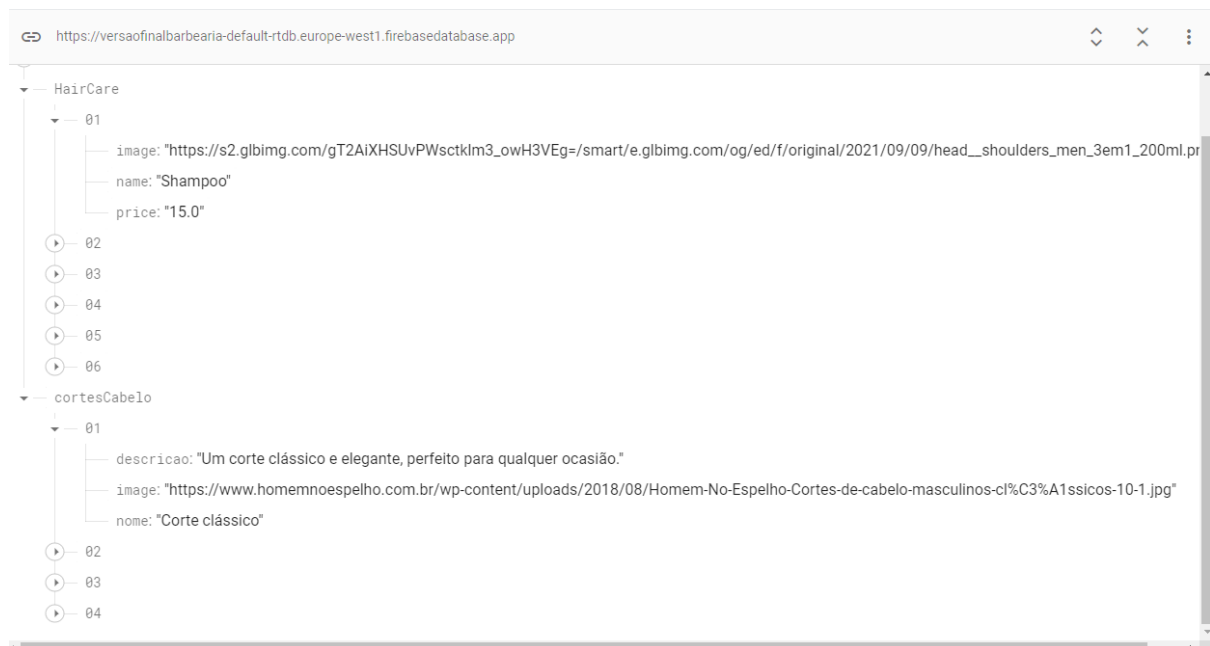
4.2 Arquitetura em camadas

A utilização da arquitetura em camadas permitiu uma organização mais clara e fácil da aplicação, separando a lógica de negócio da interface do utilizador. As camadas foram organizadas em três níveis: a camada de apresentação, que contém as atividades e os layouts em XML; a camada de negócio, que contém as classes de adaptadores, eventos, listeners e utils; e a camada de dados, que contém as classes de acesso a base de dados Firebase.

4.3 Realtime Database do Firebase

Para armazenar os produtos da loja e os cortes de cabelo do catálogo, a aplicação utiliza o Realtime Database do Firebase, que é uma base de dados em tempo real e em nuvem que permite armazenar e sincronizar dados em tempo real. Essa abordagem permite

que a aplicação tenha um acesso rápido e eficiente aos dados e que os dados estejam sempre atualizados para todos os dados.



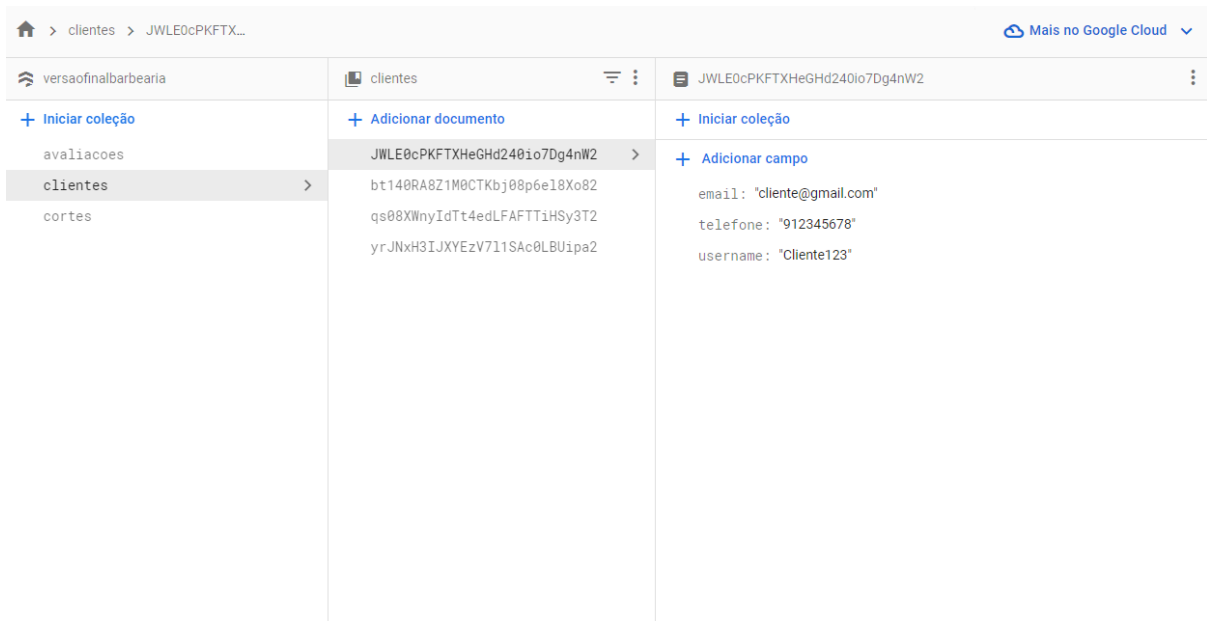
4.4 Autenticação do Firebase

Além disso, a aplicação utiliza a Autenticação do Firebase para permitir que os clientes façam o registo e login na aplicação. As regras de autenticação foram respeitadas para proteger os dados pessoais dos clientes e garantir a segurança da aplicação.

Pesquise por endereço de e-mail, número de telefone ou UID do usuário					Adicionar usuário		
Identificador	Provedores	Data de criação	Último login	UID do usuário			
ricardo@gmail.com		26 de abr. d...	26 de abr. d...	m3o8wFHNPCVCUMMrlkpehlALU...			
jpfernandes@gmail.com		25 de abr. d...	26 de abr. d...	xzclbCINm4gTG7GPF8EzvdCA7nq1			
cliente@gmail.com		25 de abr. d...	27 de abr. d...	bt140RA8Z1M0CTKbj08p6el8Xo82			

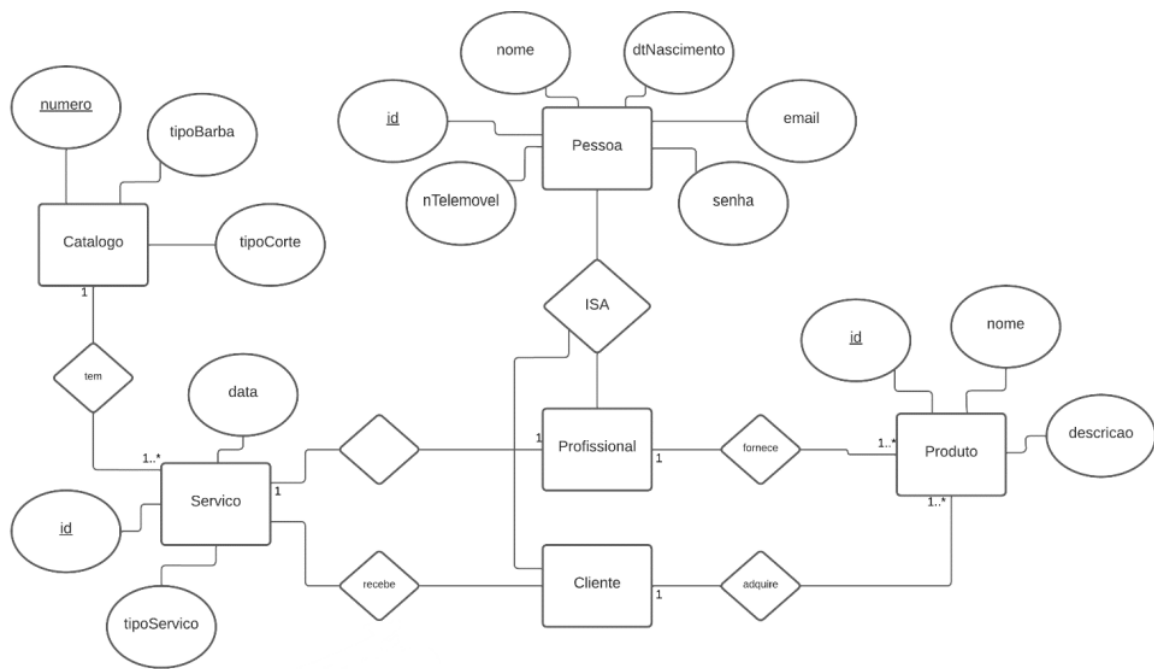
4.5 Firestore Database do Firebase

As avaliações, os dados dos clientes e os cortes de cabelo marcados no agendamento são armazenados no Firestore Database do Firebase, que é uma base de dados em nuvem que permite armazenar, sincronizar e consultar dados em tempo real.



4.6 Modelo entidade associação

Um modelo de entidade-relacionamento (também conhecido como modelo de entidade-associação) é uma representação visual de uma base de dados que mostra as entidades (tabelas) e as relações entre elas. As entidades são representadas por retângulos com seus nomes no topo, e as relações entre elas são representadas por linhas. As linhas podem mostrar relações 1:1, 1:muitos ou muitos:muitos entre as entidades. Um modelo de entidade-relacionamento também pode mostrar atributos das entidades, que são os campos de cada tabela. O objetivo de um modelo de entidade-relacionamento é fornecer uma visão geral da estrutura da base de dados e como as entidades se relacionam entre si. É uma ferramenta útil para visualizar e entender como uma base de dados é estruturada e como os dados são armazenados e consultados.



5. Implementação

5.1 Tecnologias utilizadas

No desenvolvimento da aplicação "Barbearia Lusófona", foram utilizadas diversas tecnologias. O Android Studio foi a ferramenta principal utilizada para desenvolver a aplicação, permitindo criar atividades em Java e layouts em XML de forma eficiente e intuitiva. Além disso, foi utilizado o Firebase como base de dados para armazenar e gerir os dados da aplicação. O Firebase possui diversas funcionalidades úteis, tais como o Realtime Database para armazenamento de dados em tempo real e o Firestore para armazenamento e consulta de dados em nuvem. Por fim, o Github foi utilizado para partilhar o código fonte, permitindo que em conjunto trabalhássemos e realizássemos as alterações feitas na aplicação.

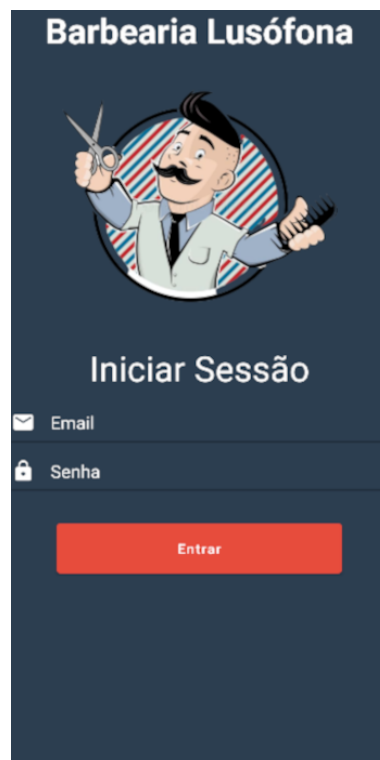


5.2 Interface

Na nossa aplicação, criamos várias interfaces para diferentes funcionalidades para a satisfação e fidelização do utilizado. A seguir, descrevemos as diferentes interfaces em detalhes:

5.2.1 Login


Foi criado um layout com campos para o utilizador inserir o seu email e palavra passe, bem como o botão para iniciar a sessão.





5.2.2 Registo


Foi criado um layout com campos para o utilizador inserir o username, e-mail, palavra-passe e número de telemóvel, bem como um botão para concluir o registo e redirecionar o utilizador para a página inicial.

Criar conta

 Email

 Senha

 Username

 Numero Telefone

☐ Aceita os termos e condições

Registo

error_view

5.2.3 Página Inicial

A página principal apresenta um menu com as várias funcionalidades da aplicação, como a loja, o catálogo com os diferentes cortes de cabelo, a marcação do serviço e as avaliações dos utilizadores. Este layout foi projetado para ser simples e fácil de usar.



Barbearia Lusófona

Bem vindo,

LOJA

CATÁLOGO DE CORTES

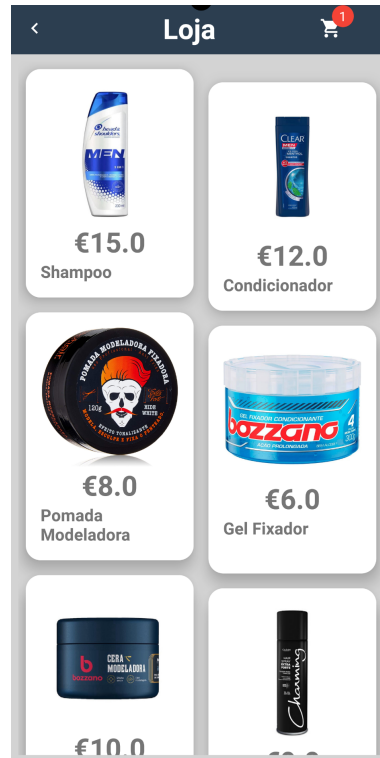
MARCAÇÃO DE SERVIÇO

AVALIAÇÃO

TERMINAR SESSÃO

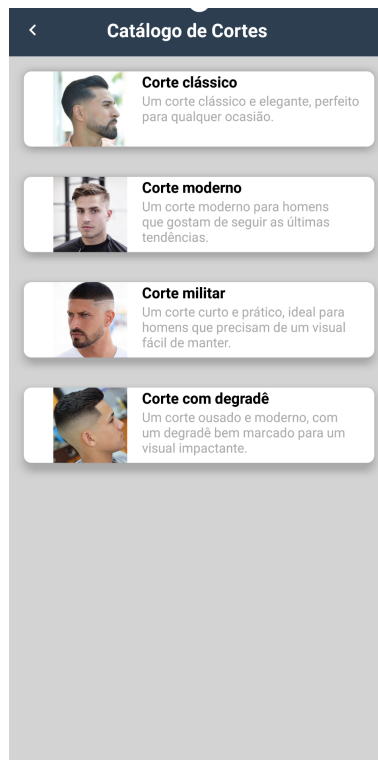
5.2.4 Loja de produtos

Estes layouts apresentam uma lista de produtos, incluindo imagem, nome e preço, e o utilizador pode seleccionar quais e quantidade de produtos quer enviar para o carrinho que faz o somatório do preço dos produtos seleccionados.



5.2.5 Catálogo de cortes de cabelo

Estes layouts apresentam uma lista de cortes de cabelo, incluindo imagem, nome e descrição, e o utilizador pode se inspirar na próxima marcação do corte, num corte do catálogo.



5.2.6 Marcação do serviço

Este layout permite ao utilizador escolher um serviço e um barbeiro para marcar um corte de cabelo. O utilizador pode seleccionar a data e hora desejada e visualizar um resumo da sua marcação.

Marcação de serviço

Escolha o barbeiro: Luís

May 11

Escolha a data: Jun 12 2023

Jul 13 2024

3 29 AM

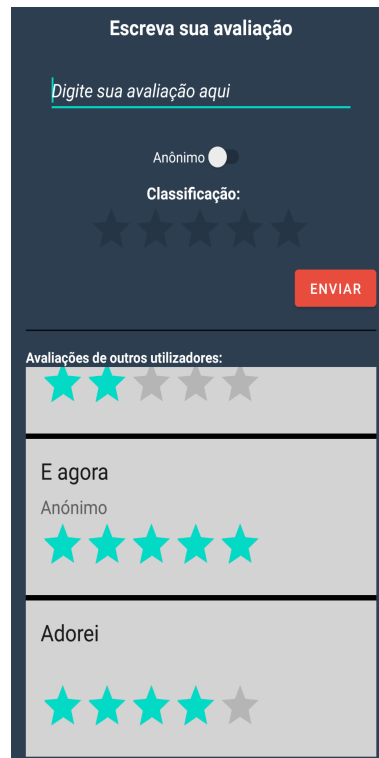
Escolha o horário: 4 : 30 PM

5 31

Confirmar Marcação

5.2.7 Avaliação

O layout de avaliação permite aos utilizadores escrever e ler avaliações de outras pessoas. O utilizador pode inserir a sua própria avaliação e visualizar todas as avaliações feitas por outros utilizadores.



The screenshot shows a mobile application interface for writing and viewing reviews. At the top, the title 'Escreva sua avaliação' is displayed. Below it is a text input field with the placeholder 'Digite sua avaliação aqui'. A toggle switch for 'Anônimo' is set to 'off'. Below the toggle is a star rating section labeled 'Classificação:' with five stars. A red 'ENVIAR' button is positioned to the right of the stars. Below the input field, there is a section titled 'Avaliações de outros utilizadores:'. This section contains three rows of star ratings. The first row shows a rating of 3 stars (3 teal stars, 2 grey stars). The second row is labeled 'E agora' and 'Anônimo', showing a rating of 5 stars (5 teal stars). The third row is labeled 'Adorei', showing a rating of 4 stars (4 teal stars, 1 grey star).

5.3 Funcionalidades da aplicação

5.3.1 Autenticação do utilizador

O utilizador tem a opção de criar uma nova conta de utilizador ou fazer login na sua conta existente. Se o utilizador escolher criar uma nova conta, ele será direcionado para a página de registo, onde terá que preencher informações como nome de utilizador, e-mail, palavra-passe e número de telemóvel. Após o registo, o utilizador será redirecionado para a página de login. Se o utilizador já tiver uma conta, pode fazer login na sua conta existente usando o seu e-mail e palavra-passe.

5.3.2 Página Inicial

Uma vez autenticado, o utilizador é redirecionado para a página inicial. Nesta página, o utilizador tem acesso a diferentes funcionalidades, incluindo a loja, catálogo de cortes de cabelo, marcação de serviços e avaliações.

5.3.3 Loja de produtos

Ao clicar na opção "Loja" na página inicial, o utilizador é direcionado para a página da loja, onde pode visualizar e seleccionar os produtos disponíveis para compra. O utilizador pode seleccionar o número de unidades que deseja de cada produto e ver o total no carrinho de compras. O utilizador também pode remover produtos do carrinho de compras ou limpar completamente o carrinho.

5.3.4 Catálogo de cortes de cabelo

Ao clicar na opção "Catálogo de cortes de cabelo" na página inicial, o utilizador é direcionado para a página do catálogo de cortes de cabelo. O utilizador pode visualizar todos os cortes de cabelo disponíveis. Poderia ser adicionado um navegador de pesquisa para facilitar a busca do corte de cabelo específico que o utilizador procura.

5.3.5 Marcação de serviço

Ao clicar na opção "Marcação de serviços" na página inicial, o utilizador é direcionado para a página de marcação de serviços. O utilizador pode seleccionar uma data disponível no calendário, escolher um horário disponível e seleccionar um barbeiro. Depois de seleccionar os dados do serviço, o utilizador recebe uma mensagem de texto a confirmar a marcação e os dados do serviço.

5.3.6 Avaliação

Ao clicar na opção "Avaliações" na página inicial, o utilizador é direcionado para a página de avaliações, onde pode dar a sua opinião sobre a aplicação ou sobre o serviço da barbearia. O utilizador pode avaliar diferentes aspetos da aplicação ou do serviço e deixar um comentário.

6. Construção de software e Testes

6.1 Codificação

6.1.1 Implementação do Firestore Database do Firebase

A base de dados do Firebase foi utilizada para armazenar os dados dos cortes de cabelo. O código estabelece a conexão com a base de dados, recupera os dados dos

cortes de cabelo e os exibe em um RecyclerView por meio do adaptador “MyCorteAdapter”. Isso permite que os utilizadores vejam os cortes de cabelo disponíveis na aplicação.

```
recyclerView = findViewById(R.id.cortelist);
database = FirebaseDatabase.getInstance( url: "https://versaofinalbarbearia-default-rtdb.europe-west1.firebaseio.com/").getReference( path: "cortesCabelo");
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));

btnBack = findViewById(R.id.btnBack);

btnBack.setOnClickListener((v -> {
    Intent review = new Intent( packageContext: cortelist.this,MainActivity.class);
    startActivity(review);
}));

list = new ArrayList<>();
myAdapter = new MyCorteAdapter( context: this,list);
recyclerView.setAdapter(myAdapter);

database.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for (DataSnapshot dataSnapshot : snapshot.getChildren()){
            Corte corte = dataSnapshot.getValue(Corte.class);
            list.add(corte);
        }
        myAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});
```

6.1.2 Carrinho de produtos

A classe CartActivity é responsável por exibir os itens do carrinho de compras e implementa a interface ICartLoadListener. Ela registra-se como assinante do EventBus no método onStart() para receber eventos de atualização do carrinho. No método onStop(), a assinatura do EventBus é removida quando a atividade é interrompida. O método onUpdateCart() é acionado pelo EventBus quando ocorre uma atualização no carrinho e chama o método loadCartFromFirebase() para atualizar os dados. No método onCreate(), os componentes da interface são inicializados. O método loadCartFromFirebase() recupera os dados do carrinho do Firebase e exibe-os na atividade. O método init() realiza a configuração inicial da atividade. O método onCartLoadSuccess() é chamado quando os dados são carregados com sucesso, calculando o total do carrinho e exibindo-o, além de associar um adaptador ao RecyclerView. O método onCartLoadFailed() exibe uma mensagem de erro caso haja falha no carregamento dos dados do carrinho. Essas funcionalidades fornecem uma experiência completa na exibição e gerenciamento dos itens do carrinho de compras.

```

@Override
protected void onStart() {
    super.onStart();
    EventBus.getDefault().register( subscriber: this);
}

@Override
protected void onStop() {
    if (EventBus.getDefault().hasSubscriberForEvent(MyUpdateCartEvent.class));
    EventBus.getDefault().removeStickyEvent(MyUpdateCartEvent.class);
    EventBus.getDefault().unregister( subscriber: this);
    super.onStop();
}

@Subscribe(threadMode = ThreadMode.MAIN,sticky = true)
public void onUpdateCart(MyUpdateCartEvent event){

    loadCartFromFirebase();
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cart);

    init();
    loadCartFromFirebase();
}

2 usages
private void loadCartFromFirebase() {
    List<CartModel> cartModels = new ArrayList<>();
    FirebaseDatabase.getInstance( url: "https://versaofinalbarbearia-default-rtdb.europe-west1.firebaseio.com/") FirebaseDatabase
        .getReference( path: "Cart") DatabaseReference
        .child( pathString: "UNIQUE_USER_ID")
        .addListenerForSingleValueEvent(new ValueEventListener() {
            2 usages
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

```

```

private void init(){
    ButterKnife.bind( target: this);

    cartLoadListener = this;
    LinearLayoutManager layoutManager = new LinearLayoutManager( context: this);
    recyclerCart.setLayoutManager(layoutManager);
    recyclerCart.addItemDecoration(new DividerItemDecoration( context: this,layoutManager.getOrientation()));

    btnBack.setOnClickListener(v -> finish());
}

2 usages
@Override
public void onCartLoadSuccess(List<CartModel> cartModellist) {
    double sum = 0;
    for (CartModel cartModel : cartModellist){
        sum+=cartModel.getTotalPrice();
    }
    txtTotal.setText(new StringBuilder("€").append(sum) );
    MyCartAdapter adapter = new MyCartAdapter( context: this, cartModellist);
    recyclerCart.setAdapter(adapter);
}

8 usages
@Override
public void onCartLoadFailed(String message) {
    Snackbar.make(mainLayout,message,Snackbar.LENGTH_LONG).show();
}

```


6.1.3 Avaliação

A atividade possui uma `ListView` para exibir avaliações dos usuários e um `TextView` vazio (`emptyView`) quando a lista está vazia. A referência `"avaliacaoRef"` conecta-se à coleção `"avaliacoes"` no `Firestore`, o banco de dados utilizado. A atividade possui uma lista de objetos `Avaliacao` e um adaptador (`avaliacaoAdapter`) para exibir os dados na lista. No método `onCreate`, são inicializados os elementos da interface e configurado um botão para enviar uma avaliação. É feita a conexão com o `Firestore` usando `FirebaseFirestore`. `addSnapshotListener` é usado para receber atualizações em tempo real dos dados da coleção `"avaliacoes"`. Quando houver alterações, a lista de avaliações é atualizada e o adaptador é notificado. O método `enviarAvaliacao` é acionado pelo clique do botão de envio. Os elementos da interface (`EditText`, `RatingBar` e `Switch`) são usados para obter os valores inseridos. É verificado se a avaliação não está vazia e exibida uma mensagem se estiver. É verificado se o `switch` `"anonimo"` está ligado ou desligado para decidir se exibir o nome do usuário na avaliação. Os dados são armazenados em um objeto `Map` chamado `novaAvaliacao`. Uma caixa de diálogo é exibida para confirmar o envio. Após o envio, uma mensagem de sucesso é exibida e os campos são limpos.

```

FirebaseFirestore db = FirebaseFirestore.getInstance();
avaliacaoRef = db.collection( collectionPath: "avaliacoes");

avaliacoesList = new ArrayList<>();
avaliacaoAdapter = new AvaliacaoAdapter( context: this, avaliacoesList);
listView.setAdapter(avaliacaoAdapter);

enviarButton.setOnClickListener(v -> enviarAvaliacao());

avaliacaoRef.addSnapshotListener((snapshot, e) -> {
    if (e != null) {
        Toast.makeText( context: ReviewActivity.this, text: "Erro ao carregar avaliações", Toast.LENGTH_SHORT).show();
        return;
    }
    avaliacoesList.clear();
    assert snapshot != null;
    for (QueryDocumentSnapshot document : snapshot) {
        Avaliacao avaliacao = document.toObject(Avaliacao.class);
        avaliacoesList.add(avaliacao);
    }
    if (avaliacoesList.isEmpty()) {
        emptyView.setVisibility(View.VISIBLE);
        listView.setVisibility(View.GONE);
    } else {
        emptyView.setVisibility(View.GONE);
        listView.setVisibility(View.VISIBLE);
        avaliacaoAdapter.notifyDataSetChanged();
    }
});
}
}
1 usage
public void enviarAvaliacao() {
    username = findViewById(R.id.autorTextView);

    EditText avaliacaoEditText = findViewById(R.id.avaliacaoEditText);
    String avaliacao = avaliacaoEditText.getText().toString().trim();

    if (TextUtils.isEmpty(avaliacao)) {
        Toast.makeText( context: this, text: "Por favor, escreva uma avaliação", Toast.LENGTH_SHORT).show();
        return;
    }
}

```

```

RatingBar ratingBar = findViewById(R.id.ratingBar);
float rating = ratingBar.getRating();

Switch switchAnonimo = findViewById(R.id.switch_anonimo);
boolean anonimo = switchAnonimo.isChecked();

if (anonimo) {

} else {

    username.setText(GlobalVar.currentUser.getName());
}

Map<String, Object> novaAvaliacao = new HashMap<>();
novaAvaliacao.put("avaliacao", avaliacao);
novaAvaliacao.put("rating", rating);
novaAvaliacao.put("anonimo", anonimo);

AlertDialog.Builder builder = new AlertDialog.Builder(context: this);
builder.setTitle("Confirmar avaliação");
builder.setMessage("Tem certeza que deseja enviar esta avaliação?");
builder.setPositiveButton(text: "Sim", (dialog, which) -> {
    // Enviar a avaliação para o Firestore
    avaliacaoRef.add(novaAvaliacao)
        .addOnSuccessListener(documentReference -> {
            Toast.makeText(context: this, text: "Avaliação enviada com sucesso", Toast.LENGTH_SHORT).show();

            avaliacaoEditText.setText("");
            ratingBar.setRating(0);
        })
        .addOnFailureListener(e -> Toast.makeText(context: this, text: "Erro ao enviar avaliação", Toast.LENGTH_SHORT).show());
});
builder.setNegativeButton(text: "Cancelar", (dialog, which) -> {

});
builder.show();
}

```

6.1.4 Marcação de Serviço

A classe `ServiceActivity` é responsável pela funcionalidade de agendamento de serviços no Android. Ela possui elementos da interface do usuário, como um `Spinner` para selecionar um barbeiro, um `DatePicker` para escolher uma data e um `TimePicker` para selecionar um horário. Também conta com um botão de confirmação (`btnConfirmar`) para finalizar o agendamento. A classe `FirebaseFirestore` é usada para obter uma instância do Firestore, o banco de dados utilizado. No método `onCreate`, os elementos da interface são configurados. Ao clicar no botão, os valores selecionados pelo usuário são obtidos, um objeto `Map` é criado para armazenar as informações e uma caixa de diálogo é exibida para confirmar o agendamento. Os dados são salvos no Firestore e uma mensagem de sucesso ou falha é exibida. Essa implementação permite aos usuários realizar o agendamento de serviços de forma interativa e simples.

A classe `ServiceActivity` é responsável pela funcionalidade de agendamento de serviços no Android. Ela possui elementos da interface do usuário, como um `Spinner` para selecionar um barbeiro, um `DatePicker` para escolher uma data e um `TimePicker` para selecionar um horário. Também conta com um botão de confirmação (`btnConfirmar`) para finalizar o agendamento. A classe `FirebaseFirestore` é

usada para obter uma instância do Firestore, o banco de dados utilizado. No método onCreate, os elementos da interface são configurados. Ao clicar no botão, os valores selecionados pelo usuário são obtidos, um objeto Map é criado para armazenar as informações e uma caixa de diálogo é exibida para confirmar o agendamento. Os dados são salvos no Firestore e uma mensagem de sucesso ou falha é exibida. Essa implementação permite aos usuários realizar o agendamento de serviços de forma interativa e simples.

```
builder = new AlertDialog.Builder( context: this);

btnConfirmar.setOnClickListener(view -> {

    String selectedBarber = spinnerBarbeiro.getSelectedItemAt().toString();

    int year = dateData.getYear();
    int month = dateData.getMonth();
    int dayOfMonth = dateData.getDayOfMonth();
    Calendar selectedDate = Calendar.getInstance();
    selectedDate.set(year, month, dayOfMonth);

    Calendar selectedTime = Calendar.getInstance();

    Map<String, Object> appointment = new HashMap<>();
    appointment.put("barber", selectedBarber);
    appointment.put("date", selectedDate.getTime());
    appointment.put("time", selectedTime.getTime());

    builder.setMessage("Deseja confirmar a marcação?")
        .setCancelable(false)
        .setPositiveButton( text: "Confirmar", (dialog, id) -> db.collection( collectionPath: "marcacao").document().documentReference()
            .set(appointment) Task<Void>
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Toast.makeText( context: ServiceActivity.this, text: "Marcação com sucesso", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText( context: ServiceActivity.this, text: "Sem marcação", Toast.LENGTH_SHORT).show();
            }
        })
        .setNegativeButton( text: "Cancelar", (dialog, id) -> dialog.cancel());

    AlertDialog alert = builder.create();
    alert.show();
});
}
```

6.2 Testes

Neste capítulo, iremos abordar os diferentes tipos de testes de software realizados na nossa aplicação. Além disso, iremos descrever o processo de avaliação da aplicação através de um questionário com utilizadores reais e duas entrevistas com barbearias locais. O objetivo é obter um feedback para melhorar a nossa aplicação e compreender as expectativas e necessidades do mercado.

6.2.1 Teste de Software

6.2.1.1 Teste unidade

Os testes de unidade desempenham um papel crucial no desenvolvimento da nossa aplicação. Permitem-nos verificar o comportamento correto de unidades individuais de código, como métodos e classes específicas. Neste contexto, iremos utilizar uma framework popular, o Mockito, para testar partes isoladas do código Android, sem depender da base de dados Firebase. Estes testes ajudar-nos-ão a garantir a funcionalidade correta de cada componente da aplicação. Iremos testar a atividade “ServiceActivity”.

```
2 usages
@Mock
private AlertDialog mockAlertDialog;

1 usage
@Mock
private DocumentReference mockDocumentReference;

10 usages
public ServiceActivity serviceActivity;

@Before
public void setup() {
    MockitoAnnotations.initMocks( testClass: this);
    serviceActivity = new ServiceActivity();

    serviceActivity.spinnerBarbeiro = mockSpinnerBarbeiro;
    serviceActivity.dateData = mockDateData;
    serviceActivity.timeHorario = mockTimeHorario;
    serviceActivity.btnConfirmar = mockBtnConfirmar;
    serviceActivity.db = mockDb;
    serviceActivity.builder = mockBuilder;

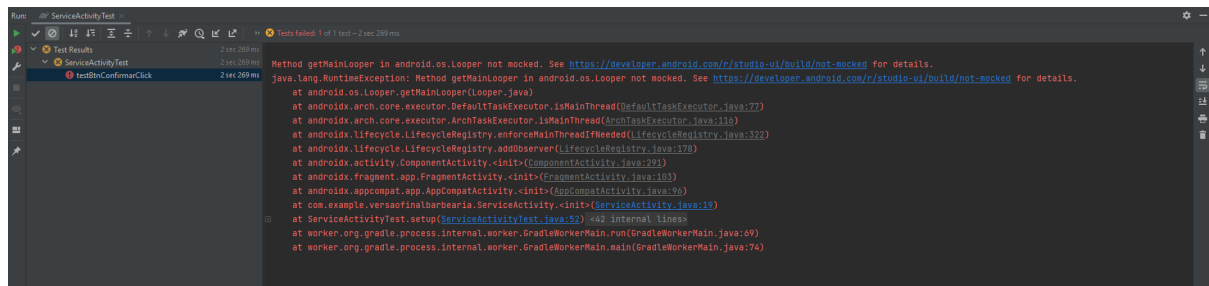
    when(mockBuilder.create()).thenReturn(mockAlertDialog);
    when(mockDb.collection( collectionPath: "marcacao").document()).thenReturn(mockDocumentReference);
}

@Test
public void testBtnConfirmarClick() {
    when(mockSpinnerBarbeiro.getSelectedItem()).thenReturn( value: "Barbeiro 1");

    serviceActivity.btnConfirmar.performClick();

    verify(mockBuilder).setMessage("Deseja confirmar a marcação?");
    verify(mockBuilder).setPositiveButton( text: "Confirmar", (DialogInterface.OnClickListener) serviceActivity);
    verify(mockBuilder).setNegativeButton( text: "Cancelar", (DialogInterface.OnClickListener) serviceActivity);
    verify(mockAlertDialog).show();
}
```

6.2.1.1.1 Resultado



6.2.1.2 Teste integração

Os testes de integração são essenciais para verificar a interação correta entre diferentes componentes da nossa aplicação, incluindo a integração com a base de dados Firebase. Desta forma, poderemos assegurar que a comunicação entre a aplicação e o Firebase ocorre de forma adequada, com dados a serem guardados e recuperados corretamente. Estes testes permitir-nos-ão identificar e corrigir problemas de integração antes do lançamento da aplicação.

6.2.1.2.1 Resultado

```
V/FA: Connection attempt already in progress
I/FirebasePerformance: Firebase Performance Monitoring is successfully initialized! In a minute, visit the Firebase console to view your data: https://console.firebase.google.com/project/versaofinalbarbearia/performance/app/android:com.example.versaofinalbarbearia/trends?utm_source=perf-android-sdk&utm_medium=android-ide
D/FirebasePerformance: onResume(): com.example.versaofinalbarbearia.PickActivity: 3121165 microseconds
V/FA: Activity resumed, time: 1983774
```

6.2.2 Análise de Feedback da Aplicação: Resultados do Questionário

6.2.2.1 Introdução

Além dos testes de software, consideramos fundamental obter o feedback dos utilizadores reais da nossa aplicação. Para isso, elaboramos um questionário abrangente e o disponibilizamos a um grupo selecionado de utilizadores. O objetivo foi compreender a experiência dos utilizadores, obter suas opiniões sobre a interface, funcionalidades e possíveis melhorias. O feedback fornecido pelos utilizadores é de extrema importância para orientar futuras atualizações e aprimoramentos na aplicação.

6.2.2.2 Resultados

Durante o questionário, obtivemos respostas de 26 pessoas, incluindo familiares, amigos, alguns desconhecidos e dois barbeiros. Ao perguntarmos sobre a opinião geral da

aplicação, usando uma escala de 1 a 5, recebemos as seguintes avaliações: 4 pessoas avaliaram com 3, 10 pessoas avaliaram com 4 e 12 pessoas avaliaram com 5. Consideramos esses resultados bastante positivos, indicando que a maioria dos utilizadores está satisfeita com a aplicação. Quanto à usabilidade da aplicação, constatamos que a grande maioria dos utilizadores a achou fácil de utilizar. Embora algumas pessoas tenham mencionado terem enfrentado alguns problemas, essas ocorrências foram relativamente raras. A funcionalidade de marcação de serviço foi considerada útil pelos utilizadores, que também elogiaram a facilidade de escolher o horário desejado e selecionar o barbeiro de preferência. A navegação na loja também foi considerada simples e útil, assim como a presença de um catálogo com vários cortes de cabelo. No entanto, aproximadamente 50% dos utilizadores não acreditam que o catálogo tenha ajudado na escolha de um novo corte. No que diz respeito à aparência da aplicação, recebemos opiniões diferenciadas. Ao avaliar de 1 a 5, obtivemos a seguinte distribuição: 1 pessoa avaliou com 1, 5 pessoas avaliaram com 2, 6 pessoas avaliaram com 3, 8 pessoas avaliaram com 4 e 6 pessoas avaliaram com 5. No geral, a maioria dos utilizadores considerou a aplicação fácil de usar e navegar, apesar das opiniões variadas sobre sua aparência.

Ao final do questionário, também demos a oportunidade aos utilizadores de fornecerem sugestões e críticas construtivas. Algumas das principais sugestões foram:

- Melhorar as validações de e-mail e nome de utilizador.
- Incluir várias imagens com perspectivas diferentes para cada corte de cabelo no catálogo.
- Na loja de produtos da aplicação, permitir que os utilizadores visualizem a descrição e o preço do produto antes de adicionar ao carrinho, além de possibilitar o envio de múltiplos itens do mesmo produto para o carrinho.
- Acrescentar a opção de eliminar a conta do utilizador.
- Após marcar o serviço, fornecer uma notificação ou confirmação visível para o utilizador.
- Adicionar animações à aplicação para torná-la mais dinâmica e envolvente.

6.2.2.3 Conclusão

Os resultados obtidos por meio do questionário destacam a satisfação geral dos utilizadores em relação à nossa aplicação. Eles consideraram a aplicação fácil de utilizar, elogiaram a funcionalidade de marcação de serviço, a facilidade de navegação na loja e a presença de um catálogo de cortes de cabelo. No entanto, algumas sugestões valiosas foram fornecidas, como melhorar as validações, aprimorar as imagens no catálogo, oferecer

mais informações sobre os produtos na loja e adicionar animações para tornar a experiência mais envolvente. Essas sugestões serão cuidadosamente consideradas para futuras atualizações e melhorias na aplicação, para proporcionar aos utilizadores uma experiência ainda melhor.

Agradecemos a todos os participantes do questionário pela sua contribuição e valiosos insights para o aprimoramento da nossa aplicação.

6.2.3 Entrevista

6.2.3.1 Barbearia Mota

6.2.3.1.1 Perguntas e respostas

Pergunta 1: Qual é a sua opinião geral sobre a aplicação? Existe alguma funcionalidade que se destacou e que você acredita que deveria ser aprimorada ou adicionada?

Resposta: Gostei! Para um projeto da faculdade, esta aplicação está bem estruturada e possui quase todos os elementos necessários para uma barbearia. No entanto, o design não é do meu agrado, prefiro que a aplicação tenha tons mais claros. O menu inicial é simples e gostei muito da presença de uma loja para a venda dos produtos. A nossa aplicação também deveria ter essa funcionalidade, mas infelizmente não está a funcionar, pois foi deixada para trás, por outro lado, na vossa aplicação, falta apenas a opção de pagamento. A ideia do catálogo é excelente, pois acredito que pode ajudar a atrair novos clientes, além de permitir que os clientes antigos escolham um novo visual com base nele. Quanto à marcação de serviço, está simples, mas eu prefiro o método que utilizo na minha aplicação, onde primeiro se escolhe o estabelecimento (no nosso caso, temos um na Maia e outro em Ermesinde) e depois o barbeiro, de forma sequencial. Mas ainda assim é possível fazer a marcação, por isso está bem feito. Também gostei da possibilidade dos clientes darem a sua avaliação e avaliarem os cortes, pois isso nos proporciona um feedback diferente. No geral, consideramos que é uma aplicação sem investimentos, está muito bem feita e com um pouco mais de trabalho profissional e orçamento, poderia ser uma ferramenta útil para várias barbearias.

Pergunta 2: Com base na sua experiência com uma aplicação, quais são os benefícios de ter uma aplicação?

Resposta: Em 2018, abri a minha barbearia, onde inicialmente as marcações eram feitas por ordem de chegada. Posteriormente, implementamos o sistema de senhas para evitar grandes filas na porta. Com o objetivo de gerir de forma mais eficiente a ordem das senhas, decidimos desenvolver uma aplicação. Inicialmente, essa aplicação tinha apenas a função de emitir senhas aos clientes, notificando-os quando o seu número estivesse próximo. No entanto, devido à pandemia de COVID-19 e numa conversa com a AppAdvice, a empresa que desenvolveu a nossa aplicação, decidimos expandir as funcionalidades para incluir o agendamento dos serviços. Dessa forma, os clientes poderiam fazer marcações não apenas para o mesmo dia, mas também para outros dias que fossem mais convenientes para eles.

6.2.3.1.2 Conclusões

Durante a entrevista com uma barbearia, foram abordados diversos aspectos relacionados à aplicação desenvolvida, bem como os benefícios de ter uma aplicação para a gestão de serviços. A seguir, apresentamos um resumo das respostas obtidas:

Quanto à opinião geral sobre a aplicação, o entrevistado destacou que, ao considerar que é um projeto acadêmico, a aplicação está bem estruturada e possui a maioria dos elementos necessários para uma barbearia. No entanto, ele expressou preferência por um design com tons mais claros. O menu inicial foi elogiado pela sua simplicidade, e a presença de uma loja virtual para venda de produtos foi bastante apreciada. O entrevistado mencionou que a sua própria aplicação também deveria ter essa funcionalidade, mas, infelizmente, não está a funcionar no momento. Além disso, ele apontou a falta de opção de pagamento na aplicação em análise. Por outro lado, a ideia do catálogo de cortes de cabelo foi considerada excelente, pois pode ajudar a atrair novos clientes e permitir que os clientes existentes escolham um novo visual com base nas opções disponíveis. Quanto à marcação de serviço, o entrevistado mencionou que prefere o método sequencial utilizado na sua própria aplicação, onde primeiro se escolhe o estabelecimento e, em seguida, o barbeiro. No entanto, ele reconheceu que é possível fazer a marcação na aplicação em análise e elogiou a funcionalidade de avaliação dos cortes de cabelo, pois isso proporciona um feedback valioso. Em resumo, o entrevistado considerou que a aplicação, mesmo sendo um projeto acadêmico sem investimentos significativos, está bem desenvolvida e com um pouco mais de trabalho profissional e orçamento, poderia ser uma ferramenta útil para várias barbearias. Em relação aos benefícios de ter uma aplicação, o entrevistado compartilhou a experiência de sua própria barbearia. Inicialmente, as marcações eram feitas por ordem de chegada, mas devido ao desejo de gerir a ordem das senhas de forma mais eficiente, eles decidiram desenvolver uma aplicação. No início, a

aplicação tinha apenas a função de emitir senhas aos clientes, notificando-os quando seu número estava próximo. No entanto, devido à pandemia de COVID-19 e a uma conversa com a empresa responsável pelo desenvolvimento da aplicação, eles decidiram expandir as funcionalidades para incluir o agendamento dos serviços. Isso permitiu que os clientes marcassem os serviços não apenas para o mesmo dia, mas também para outros dias mais convenientes para eles.

Essa entrevista evidencia a importância de uma aplicação para barbearias, pois ela pode melhorar a experiência do cliente, simplificar a gestão das marcações e fornecer uma plataforma para vendas e interação com os clientes. Com base no feedback obtido e nos benefícios destacados, é possível perceber o potencial da aplicação em análise e as oportunidades de aprimoramento e expansão para atender às necessidades das barbearias e dos seus clientes.




7. Conclusão

Ao longo deste projeto, conseguimos desenvolver com sucesso uma aplicação para uma barbearia, destinada a dispositivos Android. Todos os requisitos funcionais foram concluídos com êxito, proporcionando aos utilizadores uma experiência melhorada ao interagir com a barbearia. Embora tenhamos implementado a funcionalidade de marcação de serviços, reconhecemos que há margem para melhorias. Atualmente, os clientes têm a opção de selecionar o barbeiro desejado, mas a escolha da data e hora é feita independentemente da disponibilidade do profissional. Como plano para o futuro, sugerimos aprimorar esse processo, permitindo que os clientes selecionem um horário com base na disponibilidade do barbeiro, oferecendo uma experiência mais conveniente e eficiente. A secção do catálogo foi um ponto positivo, recebendo ótimos feedbacks por parte dos utilizadores que testaram a aplicação. A inclusão de um catálogo com vários cortes de cabelo proporcionou aos clientes uma visão abrangente das opções disponíveis na barbearia, resultando numa maior satisfação e facilitando a escolha do estilo desejado. Além disso, a adição de uma página de avaliações foi muito bem acolhida pelos novos utilizadores da aplicação, fornecendo informações confiáveis e úteis na decisão de utilizar os serviços da barbearia. Como parte dos nossos planos futuros, estamos a considerar a possibilidade de implementar um sistema de pagamento na aplicação, permitindo que os clientes efetuem pagamentos de forma segura e conveniente. Além disso, planeamos incluir a opção de levantamento de produtos na barbearia ou a escolha de locais para recolha, proporcionando aos clientes uma experiência completa de compra e conveniência. Uma

perspetiva interessante para o futuro é estabelecer parcerias com outras barbearias, personalizando a aplicação de acordo com as suas necessidades e identidade visual. Essa colaboração poderia fortalecer a presença da barbearia no mercado, permitindo oferecer uma experiência única e personalizada aos seus clientes. Uma ideia inicial que surgiu durante o projeto foi a implementação de um sistema de acumulação de pontos, permitindo que os clientes acumulem pontos por cada compra de produtos ou marcação de serviço. Esses pontos poderiam ser posteriormente utilizados para obter descontos em futuras compras ou serviços, incentivando a fidelidade dos clientes e promovendo a continuidade da utilização da aplicação.

Em resumo, o desenvolvimento desta aplicação para dispositivos Android, destinada a uma barbearia, foi um sucesso, com todos os requisitos funcionais concluídos com êxito. Os nossos planos futuros incluem melhorar a funcionalidade de marcação de serviços, implementar um sistema de pagamento, oferecer o levantamento de produtos, personalizar a aplicação para outras barbearias e explorar a possibilidade de um sistema de acumulação de pontos. Estas melhorias contribuirão para oferecer aos clientes uma experiência aprimorada, estabelecendo a aplicação como uma solução abrangente para as necessidades da barbearia e dos seus clientes.

8. Referências

1. Singh,A.(2022).
 How to change app NAME and LOGO in Android Studio? || App logo change er...
[Video]
2. EDTM Dev.(2021).  Android Development Tutorial - Build Cart with Firebase part...
[Video]
3. Jovem Programador.(2022).
 AUTENTICAÇÃO DO FIREBASE COM EMAIL E SENHA [Video]
4. Dev Vai Longe. (2022).
 Streamline Communication | Grammarly Makes Writing One Click Simpler [Video]
5. TreinaWeb. (2018).  Testando uma aplicação Android com o Firebase Test Lab
[Video]