



UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e de Engenharia

Curso: Engenharia Informática

Cadeira: Projeto/Estágio

Orientado por: Professor Filipe Quintal
Bernardo Luis

Elaborado por: Inês Jardim

Data: 23/07/2024

Índice

Índice	2
1. Introdução	3
1.1. O problema	3
1.1.1. Introdução do problema	3
1.1.2. O problema	3
2. Solução	4
2.1. Implementação em servidor local	4
2.2. Implementação do Goggle Wallet	4
2.3. Implementação em Android Studio: Kotlin	5
3. Revisão de Literatura	6
3.1. Soluções semelhantes	6
3.1.1. <i>Starbucks</i>	6
3.1.2. <i>McDonald's</i>	6
3.2. Ferramentas	7
3.2.1. Android Studio	7
3.2.2. Kotlin	8
3.2.3. Google wallet cards	8
3.2.4. Postman	9
3.2.5. Retrofit	9
3.3. Sumário	10
4. Implementação	11
4.1. Pesquisa e implemetação no Android Studio:	11
4.2. Pesquisa sobre Google Wallet Services:	11
4.3. Implementação do servidor local	13
4.4. Implementação do Loyalty Card	14
4.5. Design da Aplicação	14
5. Avaliação	16
6. Discussão	17
7. Referências	18

1. Introdução

Neste capítulo introdutório, o âmbito e objetivo do projeto e o projeto proposto, serão introduzidos.

1.1. O problema

1.1.1. Introdução do problema

Os smartphones tornaram-se uma parte indispensável da vida diária, facilitando diversas atividades por meio de aplicações móveis. Com o advento das carteiras digitais (Digital Wallets), é possível substituir muitos documentos físicos, incluindo cartões de identificação universitária e cartões de fidelidade de restaurantes, por versões digitais que oferecem maior conveniência e funcionalidades aprimoradas.

Os métodos tradicionais de programas de fidelidade de perda ou danos aos cartões, além de dificuldades na gestão manual de pontos e recompensas. Além disso, esses métodos carecem de integração com tecnologias modernas que poderiam melhorar significativamente a experiência do usuário. Essas limitações resultam em menor envolvimento dos clientes e desafios administrativos para os restaurantes, comprometendo a eficácia dos programas de fidelidade tradicionais.

1.1.2. O problema

O objetivo será desenvolver uma aplicação móvel que permita aos restaurantes criar e gerenciar um sistema de fidelidade digital. Esta aplicação será especialmente útil para a acumulação de pontos através do escaneamento de QR code e resgatar recompensas ao atingir um determinado número de pontos.

O público-alvo principal são os restaurantes que desejam a fidelização de clientes que procuram uma maneira conveniente de acumular e resgatar pontos de refeição. Este sistema digital de fidelidade não só simplificará a prática e moderna para os clientes, alinhando-se à crescente adoção de carteiras digitais em suas rotinas diárias.

2. Solução

Como mencionado na introdução, a solução que desenvolvemos passará por 3 fases principais:

2.1. Implementação em servidor local

A implementação do projeto envolveu tecnologias modernas e amplamente utilizadas para garantir eficiência, flexibilidade e fácil manutenção. Utilizamos um formato leve de troca de dados, ideal para a comunicação entre o frontend e o backend, permitindo uma transmissão de informações clara e estruturada.

Foi utilizado JavaScript, para frontend, que é uma linguagem de programação versátil que permitiu a manipulação dinâmica da interface, tratamento de eventos do usuário e envio de requisições assíncronas ao servidor. Isso garante uma experiência de usuário interativa e responsiva.

Foi também utilizado Node e Express, para backend, pois é um ambiente de execução robusto que permite a criação de servidores eficientes e escaláveis. Com o auxílio de um framework popular, conseguimos configurar rotas, gerenciar requisições HTTP e lidar com middleware de forma eficaz. A arquitetura escolhida foi ideal para aplicações que requerem alta escalabilidade e desempenho.

Esta combinação de tecnologias proporcionou uma comunicação eficiente entre as partes do sistema, uma arquitetura flexível e um desenvolvimento ágil.

2.2. Implementação do Google Wallet

Implementamos o Google Wallet no nosso projeto para oferecer uma solução eficiente e segura para a gestão de cartões de fidelidade. Utilizando a funcionalidade de cartões de fidelidade do Google Wallet, integramos uma plataforma robusta que permite aos usuários armazenar e acessar facilmente os cartões digitais. A configuração envolveu a criação de um projeto no Google Cloud Console, instalação do SDK do Google Wallet, definição de modelos de cartões de fidelidade e implementação de endpoints no backend para gerenciamento dinâmico dos cartões. Esta integração aprimorou a experiência do usuário, aumentou o engajamento e garantiu a segurança dos dados, proporcionando uma solução moderna e confiável para a fidelidade dos clientes.

2.3. Implementação em Android Studio: Kotlin

A implementação do google wallet no android studio utilizando Kotlin, sugerida pelo orientador do projeto, foi essencial para garantir uma integração eficiente e robusta. Usando Kotlin, uma linguagem moderna e expressiva, desenvolvemos uma aplicação android que facilita a gestão de cartões de fidelidade.

Começamos criando um novo projeto no android studio e configurando as dependências necessárias. Implementamos a autenticação de usuários com a api do google sign-in para garantir a segurança. Utilizamos o SDK do google wallet para criar e gerenciar cartões de fidelidade, definindo sua estrutura com classes de dados Kotlin. Desenvolvemos uma interface atraente e intuitiva, aproveitando as ferramentas de design do android studio, e implementamos comunicações com o backend utilizando Retrofit e Gson para sincronizar os dados em Json.

Integrar o Google Wallet no Android Studio com Kotlin resultou em uma solução moderna e eficiente para a gestão de cartões de fidelidade em dispositivos móveis, melhorando a experiência do usuário e aumentando o engajamento.

3. Revisão de Literatura

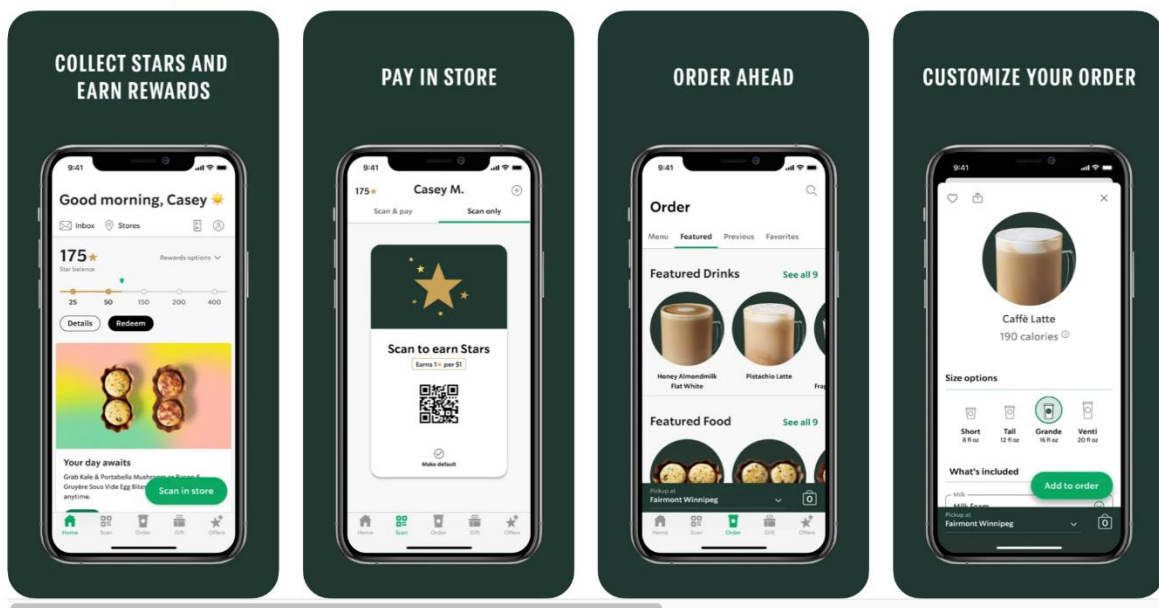
3.1. Soluções semelhantes

Nesta secção serão apresentadas aplicações parecidas à solução que foi desenvolvida e como a solução referida acima pode oferecer algo que estas soluções não oferecem ao utilizador.

3.1.1. Starbucks

A aplicação Starbucks é um exemplo de aplicação como a ser desenvolvida no projeto. É uma plataforma abrangente que permite aos clientes acumularem pontos em casa compra. Estes pontos podem ser trocados por itens gratuitos.

Como conseguimos ver na imagem abaixo, podemos observar um exemplo de cartão de fidelidade com o QR code, mas ao testar a aplicação, foi observado que o uso desse cartão com o google wallet não era possível.



3.1.2. McDonald's

A aplicação McDonald's é outro exemplo de aplicação na mesma área que o projeto, esta aplicação oferece "Rewards", permitindo que os clientes acumulem em suas compras, que podem ser trocados por produtos gratuitos ou com desconto.

Como referido no exemplo acima, também conseguimos observar o cartão de fidelidade da aplicação e também a falta de acessibilidade ao google wallet.



Com estes exemplos conseguimos concluir a falta de integração das carteiras digitais com os seus cartões de fidelidade. Algo que era fundamental na implementação neste projeto.

3.2. Ferramentas

O processo de desenvolvimento utilizado para desenvolver a solução foi um processo ágil, onde a colaboração entre o aluno e os orientadores foi contínua ao longo do desenvolvimento.

Nas secções abaixo detalha-se as tecnologias usadas e as suas características.

3.2.1. Android Studio

O android studio é um ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicações android. [1]

O android studio oferece um conjunto completo de ferramentas para a criação de aplicações android:

a. Editor de Código:

Com suporte a recursos avançados como autocompletar, refatoração de código e navegação rápida, o editor de código do android studio facilita a escrita e manutenção do código.

b. Designer de Interface:

O android studio inclui um designer de interface intuitivo que permite arrastar e soltar elementos para criar layouts visuais, bem como visualizar e editar o código XML correspondente.

c. Emulador Android:

O emulador integrado permite testar e depurar aplicativos em um ambiente virtual que simula uma variedade de dispositivos android, tamanhos de tela e versoes do sistema operacional.

d. Ferramentas de Depuração:

Com capacidades de depuração abrangentes, incluindo inspeção de variaveis, pontos de interrupção e perfil de desempenho, os desenvolvedores podem identificar e corrigir problemas rapidamente.

e. Sistema de Compilação:

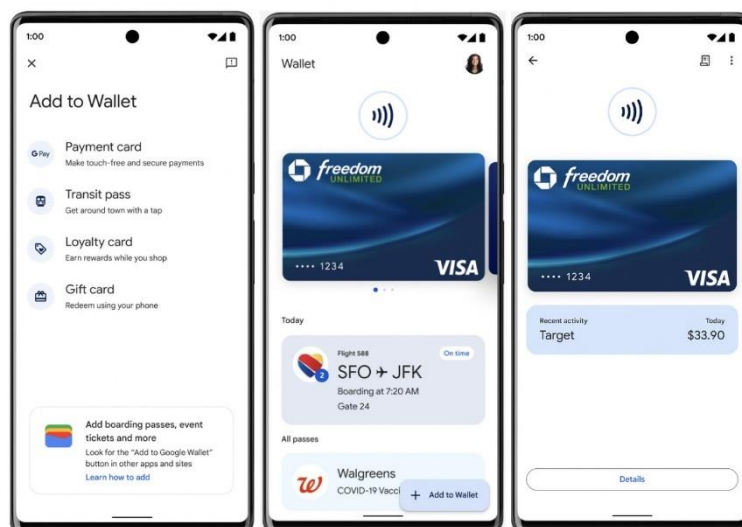
Baseado no Gradle, o sistema de compilação flexivel do android studio permite configurar e gerenciar dependencias, criar multiplas variantes de build e automatizar tarefas.

3.2.2. Kotlin

Kotlin é uma linguagem de programação moderna desenvolvida pela JetBrains, totalmente interoperável com Java. Kotlin é conhecida por sua concisão, segurança e facilidade de uso, oferecendo várias vantagens sobre Java, como exemplo, sintaxe concisa, interoperabilidade, etc. [2]

3.2.3. Google wallet cards

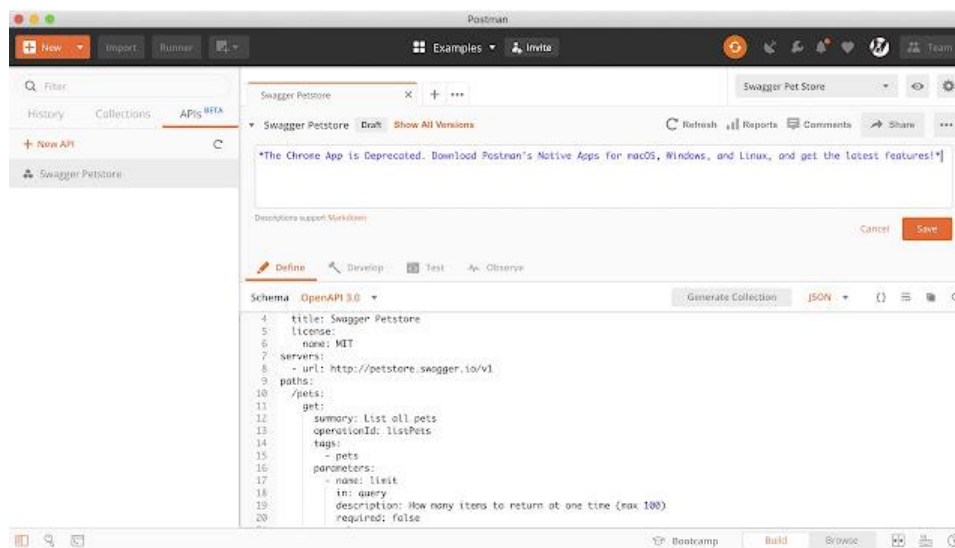
O google wallet cards é uma funcionalidade do google wallet que permite aos usuarios armazenar digitalmente diversos tipos de cartoes diretamente em seus dispositivos moveis. Esta ferramenta proporciona uma maneira pratica e segura de gerenciar cartoes de credito, debito, fidelidade, ingressos e passes.



Esta ferramenta protege os dados dos cartões com tecnologias avançadas, como a tokenização e a autenticação biométrica. Sincroniza automaticamente entre dispositivos, garantindo que as informações dos cartões estejam sempre atualizadas e acessíveis. [3]

3.2.4. Postman

Postman é uma ferramenta usada para desenvolvimento e teste de APIs, amplamente utilizada por desenvolvedores devido à sua intuitiva e funcionalidades abrangentes. Disponível como uma aplicação de desktop para Windows, macOS e Linux, Postman permite criar, enviar e analisar requisições HTTP de forma eficiente.



Esta ferramenta suporta os tipos de requisições HTTP, como GET, POST, PUT, DELETE. Permite também a inclusão de parâmetros, cabeçalhos e corpo da requisição.

Tem uma interface intuitiva, que suporta diversas requisições e personalizações através de scripts e variáveis. [4]

3.2.5. Retrofit

Retrofit é uma biblioteca para Android que simplifica a comunicação com serviços web, facilitando a integração de APIs RESTful em aplicativos móveis. A biblioteca oferece uma forma robusta e eficiente para enviar e receber dados de e para servidores, usando anotações para definir os endpoints das APIs e convertendo automaticamente as respostas em objetos Java. [5]

3.3. Sumário

Neste projeto, utilizamos diversas ferramentas e tecnologias para implementar funcionalidades e garantir a eficiência do desenvolvimento.

Cada uma destas ferramentas desempenham um papel crucial na criação de um asolução coesa e eficiente. O Google Wallet Cards oferece uma maneira prática de gerenciar cartões de fidelidade, enquanto o Postman e o Retrofit garantem a integração e teste eficaz das APIs necessárias. O Android Studio, combinado com Kotlin, proporciona um ambiente robusto e moderno para o desenvolvimento do aplicativo. A combinação dessas ferramentas garante uma implementação bem-sucedida e uma experiência aprimorada para os usuários finais.

4. Implementação

Nesta secção será abordado, de uma forma mais aprofundada, a solução ao problema proposto. Serão explicadas as fases do projeto, desde a implementação do projeto desenvolvido, para verificar a facilidade de implementação, até ao desenvolvimento de outro algoritmo para simular a afluência de pessoas ao serviço.

4.1. Pesquisa e implemetação no Android Studio:

Antes de qualquer implementação do projeto fui fornecido um pequeno curso de como utilizar as ferramentas fornecidas pelo Android Studio. Foi também comunicado a ideia inicial esperada pelo projeto.

Após a introdução ao Android Studio utilizando Kotlin como linguagem de implementação foi realizado o primeiro prototipo com pequenas funcionalidas, que depois foram alteradas para o prototipo final.

O primeiro prototipo consistia de uma simples aplicação pensando unicamente da usabilidade do cliente. Onde poderíamos encontrar um formulario para a criação do cartao de fidelidade e a integração do mesmo com o google wallet.

Após futuras observações, foi entao implementado a funcionalidade por parte da entidade impreendedora (restaurante), que permitia a criação de um novo cartao e a leitura do qr-code do mesmo para possiveis alterações de dados.

4.2. Pesquisa sobre Google Wallet Services:

Para a implementação de alguma funcionalidade por parte dos serviços da Google, foi feita uma pesquisa aprofundada sobre a funcionalidade e ferramentas fornecidas pela Google Wallet Services.

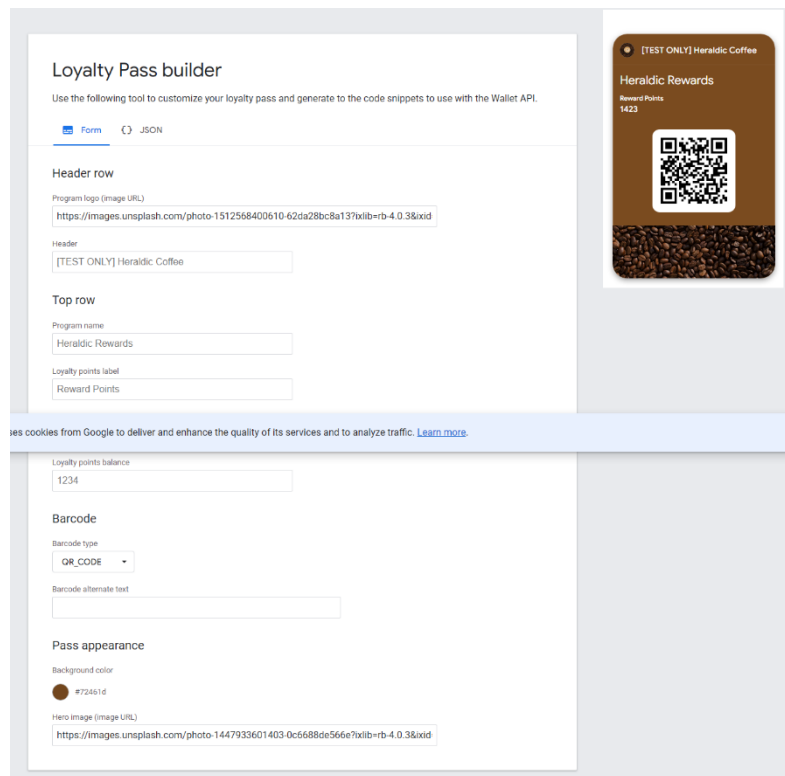
Para o uso das ferramentas precisas da Google Wallet, foi necessario a criação de uma conta na Google Pay & Wallet Console, que nos permite a craição de uma classe e acesso aos serviços da google wallet por um ID de emissor (issuerId). [6]

Foi observado a existencia de varios tipos de cartoes de fidelidade, que inicialmente foi utilizado o “generic pass”, por ser um tipo de cartao de fidelidade mais generico. Apos futura pesquisa foi reavaliado que havia um tipo de cartao de fidelidade mais apropriado ao projeto a ser desenvolvido, sendo esse tipo de cartao o “Loyalty Card”, pois fornecia a funcionalidade e requerimentos ideal para o projeto. [7]

Tambem foi utilizado as ferramentas Codelabs e pass builder, que fornece os passos necessarios para a implementação da funcionalidade de adicionar o cartao de fidelidade a google wallet, e a personalização do mesmo. [8] [9]

Quando nos referimos aos cartoes de fidelidade temos de ter em conta a sua implemetação que consiste na criação de uma classe que por si é a base “fixa” do cartao, esta classe é criada

uma unica vez por entidade, que se refere ao visual e informação presente no cartao. Temos tambem a criaçao de um objecto que por si é a parte variavel do cartao, logo a informação unica para cada cartao/pessoa.



Podemos observar a ferramenta para personalizar o cartao de fidelidade, onde podemos alterar qualquer informação presente neste tipo de cartao. Que apos as mudanças feitas sao recebidos partes de codigo. [9]

Object definition

```
{
  "id": "ISSUER_ID.OBJECT_ID",
  "classId": "ISSUER_ID.LOYALTY_CLASS_ID",
  "loyaltyPoints": {
    "balance": {
      "int": "1234",
    },
    "localizedLabel": {
      "defaultValue": {
        "language": "en-US",
        "value": "Reward Points",
      },
    },
  },
  "barcode": {
    "type": "QR_CODE",
    "value": "BARCODE_VALUE",
    "alternateText": "",
  },
}
```

Nesta parte de codigo conseguimos ver a parte do objecto que é personalizada a cada pessoa.

```

{
  "id": "ISSUER_ID.LOYALTY_CLASS_ID",
  "programLogo": {
    "sourceUri": {
      "uri": "https://images.unsplash.com/photo-1512568400610-62da28bc8a13?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=660&h=660",
    },
    "contentDescription": {
      "defaultValue": {
        "language": "en-US",
        "value": "LOGO_IMAGE_DESCRIPTION",
      },
    },
  },
  "localizedIssuerName": {
    "defaultValue": {
      "language": "en-US",
      "value": "[TEST ONLY] Heraldic Coffee",
    },
  },
  "localizedProgramName": {
    "defaultValue": {
      "language": "en-US",
      "value": "Heraldic Rewards",
    },
  },
  "hexBackgroundColor": "#72461d",
  "heroImage": {
    "sourceUri": {
      "uri": "https://images.unsplash.com/photo-1447933601403-0c6688de566e?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1032&h=336",
    },
    "contentDescription": {
      "defaultValue": {
        "language": "en-US",
        "value": "HERO_IMAGE_DESCRIPTION",
      },
    },
  },
}

```

Nesta parte de código podemos ver a parte “fixa” da classe.

4.3. Implementação do servidor local

Após a pesquisa sobre a funcionalidade de "Loyalty Cards", avançamos para a implementação do servidor local, utilizando o computador como servidor com Node Express. Após a instalação e configuração do servidor, foram criados vários endpoints para testar a conectividade entre a aplicação e o servidor. Para o protótipo final, foram implementados quatro endpoints distintos: **createPass**, **createObject**, **updatePass**, e **GetPass**. Todos esses endpoints estão relacionados às funcionalidades descritas anteriormente.

```

app.post('/createPass', async (req,res)=>{
    const { issuer_id, class_variable, object_variable} = req.body;
    const class_suffix = `${class_variable}${card.constructor.name}`;
    const object_suffix = `${object_variable}${card.constructor.name}`;
    console.log('Create a Pass for every client');
    try {
        const token = await card.createObject(issuer_id, class_suffix, object_suffix);
        if (token === null)
            res.status(500).send(`Error, pass already created: ${error.message}`);
        else
            res.status(200).send(token);
    } catch (error) {
        console.error(`Error creating class: ${error.message}`, error); // Detailed logging
        res.status(500).send(`Error creating pass: ${error.message}`);
    }
});

```

Estes são exemplos de endpoints, todos com a mesma base, e são chamados usando o Retrofit. A principal função desses endpoints é receber um token, que é utilizado pelos serviços do Google Wallet. Esse token é crucial para a criação e alterações dos cartões de fidelidade.

4.4. Implementação do Loyalty Card

Quanto ao cartão de fidelidade, foi necessário criar várias funções para suportar as diferentes funcionalidades relacionadas aos cartões de fidelidade.

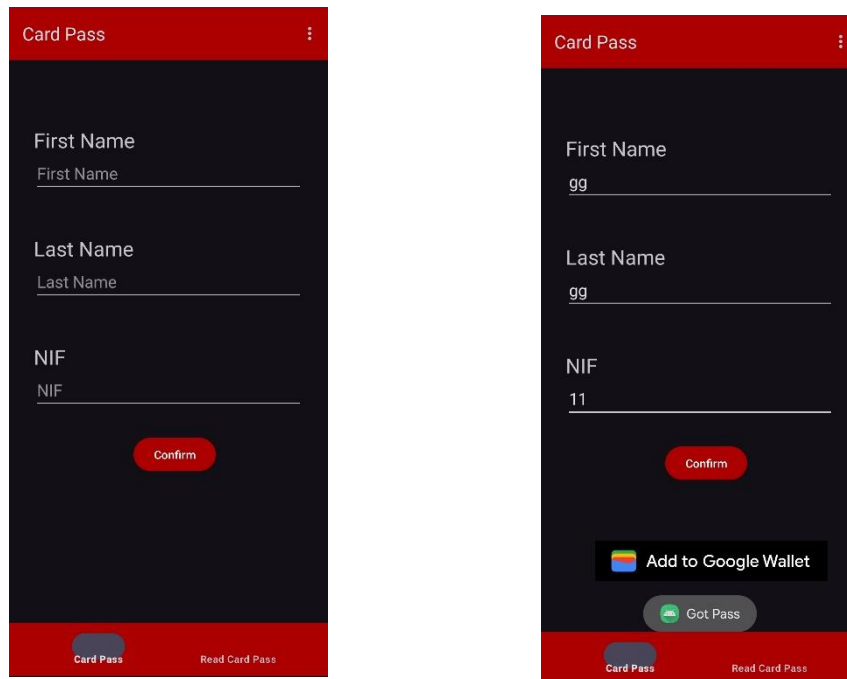
1. **createClass**: Esta função é utilizada apenas uma vez e deve ser executada antes de qualquer outra funcionalidade. Ela define a estrutura e os atributos do cartão de fidelidade.
2. **createObject**: Esta função é chamada sempre que um novo cliente deseja criar o seu cartão. O funcionário da entidade utiliza esta funcionalidade, preenchendo um formulário com os dados do cliente, especialmente um número único que identifica o cliente. Com este número, é criado um cartão de fidelidade único para o cliente.
3. **updatePass**: O funcionário também pode utilizar esta função para ler o QR code presente no cartão. Através dela, é possível atualizar as informações permitidas no cartão, como o número de pontos que o cliente possui.
4. **GetPass**: Do lado do cliente, esta função permite buscar as informações do cartão de fidelidade específico do cliente. Após recuperar essas informações, o cliente pode usar a função para adicionar o cartão à Google Wallet.

Essas funções foram desenvolvidas para garantir uma gestão eficiente e fácil dos cartões de fidelidade, tanto para os funcionários quanto para os clientes.

4.5. Design da Aplicação

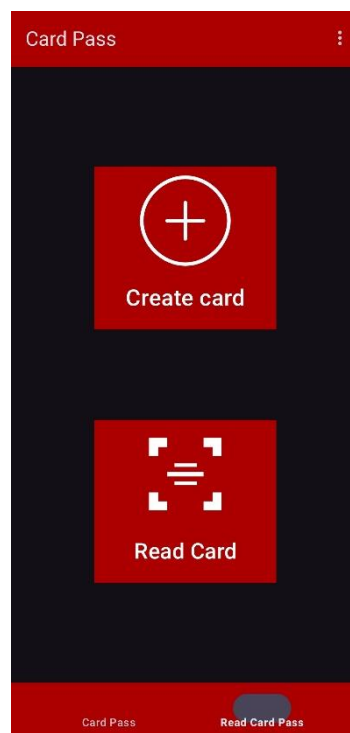
Para o visual da aplicação, foi adotado um design simples, visando não dificultar o seu uso. Como o principal objetivo do projeto é priorizar a utilização do Google Wallet, a interface foi desenvolvida para ser direta e focada na funcionalidade. A cor de fundo da aplicação adapta-

se automaticamente ao tema do telemóvel, garantindo uma experiência visual harmoniosa e consistente com as preferências do usuário.

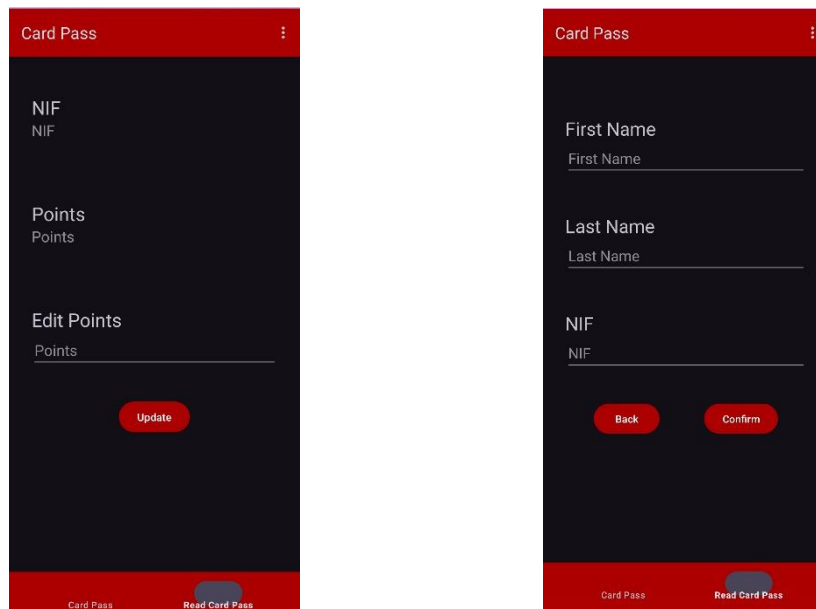


The image displays two side-by-side screenshots of a mobile application titled "Card Pass". Both screens feature a dark blue background with white text and red accents. The left screenshot shows the initial registration form with empty input fields for "First Name", "Last Name", and "NIF", and a red "Confirm" button. The right screenshot shows the same form after data entry: "First Name" and "Last Name" are both filled with "gg", and "NIF" is filled with "11". In this state, the "Confirm" button remains red, but a new button labeled "Add to Google Wallet" with a Google Wallet logo appears below it. Additionally, a "Got Pass" button with a green checkmark icon is visible. At the bottom of both screens, there are two small buttons: "Card Pass" and "Read Card Pass".

A primeira imagem acima mostra a interface inicial, antes de qualquer interação. Após o preenchimento do formulário, o botão para adicionar ao Google Wallet torna-se visível. Depois dessa interação, o cartão de fidelidade será adicionado ao Google Wallet.



Aqui, conseguimos ver a interface do funcionário, que oferece duas opções principais: "Criar um Novo Cartão" ou "Ler QR Code". A interface mantém-se simples e intuitiva, refletindo a simplicidade da aplicação.



A primeira imagem ilustra a interface quando a opção "Ler Cartão" é selecionada. Nesta tela, um leitor de QR code é ativado, permitindo que o usuário escaneie o código presente no cartão. Após a leitura, o sistema busca automaticamente as informações necessárias associadas ao cartão, proporcionando uma experiência rápida e eficiente ao usuário.

A segunda imagem representa a interface ao escolher a opção "Criar Novo Cartão". Esta interface é idêntica à do cliente, garantindo uma experiência de uso consistente e intuitiva. O usuário pode inserir os dados necessários para criar um novo cartão, incluindo informações pessoais e detalhes específicos que serão utilizados no sistema. Esta funcionalidade é projetada para ser simples e direta, facilitando a criação de novos cartões de forma eficiente.

Essas interfaces foram cuidadosamente desenhadas para oferecer uma experiência de usuário coesa e eficaz, permitindo tanto a leitura rápida de cartões existentes quanto a criação fácil de novos cartões.

5. Avaliação

O processo revelou-se bastante desafiador, especialmente no que diz respeito à integração com serviços de terceiros, particularmente os fornecidos pela Google. Trabalhar com um processo predefinido complicou a identificação dos requisitos exatos para o projeto. Esta

difficuldade foi maior pelo fato de, desde o início, termos utilizado um tipo de cartão que não estava tão desenvolvido e otimizado quanto o tipo final necessário para a implementação.

A transição e o tempo despendido para encontrar e ajustar ao tipo de cartão correto impactaram significativamente o andamento do projeto. Este atraso influenciou vários aspectos do funcionamento do projeto, dificultando a sua finalização. A complexidade adicional proveniente destas mudanças aumentou a necessidade de revisões contínuas e adaptações ao longo do desenvolvimento.

Apesar destes obstáculos, a utilização do Android Studio foi relativamente tranquila e intuitiva. A criação e configuração do servidor local também não apresentaram grandes dificuldades, mostrando-se processos bastante diretos. Estas ferramentas de desenvolvimento facilitaram a implementação e testes das funcionalidades do projeto, permitindo que focássemos mais na resolução dos problemas principais relacionados aos serviços de terceiros e ao tipo de cartão.

Em resumo, os principais desafios enfrentados foram a integração com os serviços da Google e a adaptação ao tipo correto de cartão.

6. Discussão

Infelizmente, não consegui completar o projeto da maneira esperada. Algumas partes do código ficaram por limpar e certas funcionalidades não operam com a eficiência desejada.

No entanto, a experiência foi bastante enriquecedora. Aprendi muito sobre a integração com terceiros e sobre como implementar um servidor local de maneira mais eficaz. Tive a oportunidade de aprofundar meus conhecimentos no uso do Android Studio IDE e, especialmente, na linguagem Kotlin.

Este projeto foi valioso para a aprendizagem de diversas ferramentas e técnicas que contribuem indiretamente para a implementação de projetos. Embora o resultado final não tenha atingido totalmente minhas expectativas, a experiência foi única e acredito que melhorou significativamente minha abordagem e habilidades em projetos similares no futuro.

7. Referências

- [1] “[Meet Android Studio | Android Developers](#)”
- [2] “[Kotlin and Android | Android Developers](#)”
- [3] “[Create passes on Android using the Google Wallet API](#)”
- [4] “[API Repository | Postman API Platform](#)”
- [5] “[Retrofit \(square.github.io\)](#)”

- [6] “[Setting up a Google Wallet API Issuer account | Loyalty cards | Google for Developers](#)”
- [7] “[Loyalty card overview | Loyalty cards | Google for Developers](#)”
- [8] “[Create passes on Android using the Google Wallet API](#)”
- [9] “[Loyalty Pass builder | Loyalty cards | Google for Developers](#)”