Universidade Paulista - UNIP

Curso: Ciência da Computação

Guilherme Aparecido Faria de Souza – RA: T588fi5

Guilherme Augusto Sbizero Correa – RA: F235289

Lucas Cardoso de Oliveira – RA: N659572

Luiz Felipe Fernandes Martins – RA: N675365

Atividades Práticas Supervisionadas (APS)

Desenvolvimento de Uma Aplicação de Sistema Distribuído para Dispositivo Móvel

Atividades Práticas Supervisionadas para o

7º/8º Semestre do Curso de Ciência da Computação

Apresentado a disciplina Desenvolvimento de Sistemas

Distribuídos

Da Universidade Paulista – UNIP

São Paulo – SP

2023

UNIVERSIDADE PAULISTA – UNIP
CIÊNCIA DA COMPUTAÇÃO

Guilherme Aparecido Faria de Souza – RA: T588fi5

Guilherme Augusto Sbizero Correa – RA: F235289

Lucas Cardoso de Oliveira – RA: N659572

Luiz Felipe Fernandes Martins – RA: N675365

Atividades Práticas Supervisionadas (APS)

Desenvolvimento de Uma Aplicação de Sistema Distribuído para Dispositivo Móvel

Atividades Práticas Supervisionadas para o

7º/8º Semestre do Curso de Ciência da Computação

Apresentado a disciplina Desenvolvimento de Sistemas

Distribuídos

Da Universidade Paulista – UNIP

São Paulo – SP
2023

RESUMO

Com os avanços atuais das tecnologias de comunicação sem fio e a velocidade da utilização dos dispositivos móveis sem fio, cresce cada vez mais o uso deles para processamento e transmissão de informações. Este projeto tem como objetivo desenvolver um software de acessibilidade para cegos em supermercados usado em um dispositivo móvel que se comunica via internet 5G, 4G, 3G ou 2G com um servidor, e estudar conceitos, tecnologias e ferramentas utilizadas. O dispositivo móvel escolhido foi um Smartphone, o software desenvolvido usa a linguagem Kotlin através da ferramenta IntelliJ IDEA e o servidor é um Banco de Dados MySql. O software de acessibilidade tem como foco pessoas com deficiência visual, indicando os itens que o usuário aponta para a câmera do smartphone, aumentando a acessibilidade dos deficientes visuais para fazerem compras. O resultado obtido foi um software que possui aplicação abrangente e satisfatória, cumprindo com suas funções de maneira eficiente, rápida e segura, proporcionando assim confiabilidade e qualidade para seus usuários finais.

Palavras-chave: Smartphone. Dispositivos Móveis. Kotlin. IntelliJ IDEA. Banco de Dados Mysql.

ABSTRACT

With current advances in wireless communication technologies and the speed of use of wireless mobile devices, their use for processing and transmitting information is increasing. This project aims to develop accessibility software for the blind in supermarkets used on a mobile device that communicates via 5G, 4G, 3G or 2G internet with a server, and to study concepts, technologies and tools used. The mobile device chosen was a Smartphone, the software developed uses the Kotlin language through the IntelliJ IDEA tool and the server is a MySql Database. The accessibility software focuses on people with visual impairments, indicating the items that the user points to the smartphone's camera, increasing accessibility for visually impaired people to make purchases. The result obtained was software that has a comprehensive and satisfactory application, fulfilling its functions efficiently, quickly, and safely, thus providing reliability and quality for its end users.

Keywords: Smartphone. Mobile Devices. Kotlin. Mysql Database. Intellij IDEA

Índice

1 OBJETIVO DO TRABALHO

A visão é um dos sentidos mais importantes que o ser humano possui. São pelos olhos que temos praticamente oitenta por cento de toda a nossa percepção diária. Algumas pessoas jamais conseguiriam imaginar viver sem a visão. Em contrapartida, uma parcela da população mundial enfrenta e convive com esse tipo de dificuldade. A vida de pessoas com deficiência visual não é algo fácil. Elas enfrentam diversos tipos de dificuldades, como, por exemplo, a falta de autonomia para executarem determinadas tarefas diárias, precisando de alguém sempre para prestar auxílio. Com o passar do tempo e a evolução da tecnologia, foram surgindo dispositivos auxiliares utilizados para suprir determinadas necessidades, com o intuito de devolver parte da autonomia para pessoas com esse tipo de dificuldade. Além de desenvolver novos dispositivos, a área da Tecnologia Assistiva, responsável por desenvolver soluções que auxiliam pessoas com algum tipo de dificuldade física, buscou também utilizar alternativas já existentes, como os smartphones, que acabaram virando uma ótima opção para auxiliar pessoas com deficiência visual por meio dos mais diversos tipos de aplicações, desde aplicativos que as conectam com pessoas interessadas em ajudar até recursos básicos como reconhecimento de cédulas monetárias, leitura de texto e etc. Este trabalho visa mostrar uma solução que vai dar mais autonomia para essas pessoas na hora de fazer suas compras, por meio de uma aplicação com determinadas funções e capaz de reconhecer os produtos e devolver os dados como a marca do próprio, para que as pessoas não tenham só mais autonomia, mas também uma liberdade maior para escolher suas marcas preferidas.

2 INTRODUÇÃO

O presente trabalho apresenta o desenvolvimento de uma aplicação mobile em Kotlin voltada para pessoas com deficiência visual. O objetivo principal da aplicação é utilizar a inteligência artificial para identificar produtos, dessa forma, dar autonomia para os deficientes visuais em suas compras. Para alcançar esse objetivo, a aplicação utiliza a câmera do smartphone para capturar a imagem do produto. Em seguida, a imagem é processada por meio de técnicas e de inteligência artificial previamente treinada com imagens de diversos tipos de produtos. Essas técnicas de aprendizado de máquina permitem que o sistema seja capaz de identificar a marca e o tipo do produto, devolvendo as informações para o usuário por meio de áudio. A aplicação tem a capacidade de processar rapidamente a imagem capturada, permitindo que o usuário tenha acesso às informações necessárias. Além disso, o sistema é capaz de armazenar as informações sobre os produtos identificados para que o usuário possa ter acesso às informações

posteriormente. A aplicação também contém uma lista para que o usuário seja informado do que já pegou, evitando duplicidades. Com isso, o aplicativo possibilita que pessoas com deficiência visual tenham mais independência e segurança ao fazer compras no mercado. Essa Tecnologia Assistiva tem o potencial de mudar a vida dessas pessoas, garantindo-lhes acesso à informação e autonomia na realização de tarefas cotidianas. A inclusão social é um dos principais objetivos das tecnologias assistivas e esse projeto é um exemplo claro de como a tecnologia pode ser utilizada para ampliar a inclusão de pessoas com deficiência visual na sociedade. A aplicação que desenvolvemos é um passo importante nesse sentido, trazendo benefícios para muitas pessoas. Com a evolução constante da tecnologia, acreditamos que projetos como este têm o potencial de melhorar ainda mais a vida de pessoas com deficiência visual e outras deficiências.

2.1 **Contextualização**

A visão é um dos sentidos mais importantes que o ser humano possui. São pelos olhos que temos praticamente oitenta por cento de toda a nossa percepção diária. Algumas pessoas jamais conseguiriam imaginar viver sem a visão. Em contrapartida, uma parcela da população mundial enfrenta e convive com esse tipo de dificuldade. A vida de pessoas com deficiência visual não é algo fácil. Elas enfrentam diversos tipos de dificuldades, como, por exemplo, a falta de autonomia para executarem determinadas tarefas diárias, precisando de alguém sempre para prestar auxílio. Com o passar do tempo e a evolução da tecnologia, foram surgindo dispositivos auxiliares utilizados para suprir determinadas necessidades, com o intuito de devolver parte da autonomia para pessoas com esse tipo de dificuldade. Além de desenvolver novos dispositivos, a área da Tecnologia Assistiva, responsável por desenvolver soluções que auxiliam pessoas com algum tipo de dificuldade física, buscou também utilizar alternativas já existentes, como os smartphones, que acabaram virando uma ótima opção para auxiliar pessoas com deficiência visual por meio dos mais diversos tipos de aplicações, desde aplicativos que as conectam com pessoas interessadas em ajudar até recursos básicos como reconhecimento de cédulas monetárias, leitura de texto e etc.

Este trabalho visa mostrar uma solução que vai dar mais autonomia para essas pessoas na hora de fazer suas compras, por meio de uma aplicação com determinadas funções e capaz de reconhecer os produtos e devolver os dados como a marca do próprio, para que as pessoas não tenham só mais autonomia, mas também uma liberdade maior para escolher suas marcas preferidas.

Presumir que todos os consumidores de ambientes comerciais como supermercados possuem todos os cinco sentidos em pleno funcionamento é completamente errado, focando apenas na deficiência visual segundo os dados da Organização Mundial da Saúde estima-se que 2,2 bilhões de pessoas se encaixam dentro desse escopo (OMS, 2019).

A perda da visão total ou parcial da visão afeta drasticamente o dia a dia de uma pessoa, causando perda de autonomia em muitas atividades, uma das mais importantes e necessárias com certeza é a ida ao supermercado para fazer as compras do mês. Na hora da compra a maneira mais comum que se tem de identificar os produtos de desejo nas prateleiras são suas embalagens, as quais existem dos mais diversos tipos, cores e tamanhos. Para o consumidor visual isso funciona gerando assim até uma competição visual nas prateleiras, já para os consumidores sem visão é algo pouco aplicável para se destacar e até mesmo para identificar o produto que está exposto (LOPES, 2014).

A atmosfera desse tipo de ambiente é constituída principalmente por recursos visuais como iluminação e merchandising, podendo conter até outros recursos sensoriais tipo música e aroma (MCGOLDRICK; MODERATORS, 1998 apud CAVALCANTE et al., 2012), causando assim a percepção de que esse tipo de ambiente possa ser hostil é pouco pensado para pessoas com determinadas necessidades especiais, contendo pouca ou nenhuma adaptabilidade para eles. No caso dos deficientes visuais é possível observar que não existe nenhuma maneira com que ele identifique o produto, pois não existe nem Braille na maioria das embalagens. Isso acaba acarretando a busca de outros meios como aplicativos de entrega ou até mesmo a necessidade de auxílio de terceiros para a realização da atividade e muitas vezes tirando sua autonomia em uma tarefa simples e corriqueira.

Um recurso muito poderoso que temos ao nosso lado hoje com certeza é a inteligência artificial (IA), que deixou de ser apenas um assunto científico e acadêmico e começou a ganhar mais espaço no dia a dia das pessoas. Ela está presente em notícias de jornais, em estratégias de negócios para empresas aumentarem seus lucros e até mesmo em discussões governamentais, nas quais são elaborados planos para explorar melhor essa tecnologia. É um fato que, ao longo do tempo, demos grandes passos em relação a essa tecnologia, que a cada dia traz mais resultados práticos e se prova cada vez mais capaz de mudar nossas vidas (LEE, 2019). Juntamente com a Tecnologia Assistiva, que consiste na utilização de recursos e serviços tecnológicos para aumentar as habilidades funcionais de pessoas com deficiência e consequentemente, promover maior independência e inclusão (BERSCH, 2008), torna-se um recurso muito poderoso.

Em síntese, o projeto tem como objetivo demonstrar que a união da IA com a Tecnologia Assistiva promove a solução de alguns problemas de inclusão e autonomia para deficientes visuais em ambientes comerciais, como supermercados e lojas de conveniência.

3 FUNDAMENTOS DAS TECNOLOGIAS PARA DISPOSITIVOS MÓVEIS ESCOLHIDOS

A linguagem de programação Kotlin é uma linguagem amplamente usada por desenvolvedores Android em qualquer lugar, fornece uma série de mecanismos para trabalhar com variáveis anuláveis.

O Kotlin apresenta vários mecanismos para implementar a lógica condicional. O mais comum deles é uma *instrução if-else*. Se uma expressão entre parênteses ao lado de uma palavra-chave if for avaliada como true, o código dentro dessa ramificação (ou seja, o código imediatamente seguinte que é encapsulado entre chaves) será executado. Caso contrário, será executado o código dentro da ramificação else.

Kotlin é uma linguagem multiplataforma, orientada a objetos e funcional concisa e estaticamente tipada, desenvolvida pela JetBrains em 2011, que compila para a Máquina virtual Java e que também pode ser traduzida para a linguagem JavaScript e compilada para código nativo.

3.1 Programação Orientada a Objetos

Programar possui modos diferentes de se fazer esses modos são chamados de paradigmas de programação e, entre eles, estão a programação orientada a **objetos** (POO) e a programação estruturada. Quando começamos a utilizar linguagens como Java, C#, Python e outras que possibilitam o paradigma orientado a objetos, é comum errarmos e aplicarmos a programação estruturada achando que estamos usando recursos da orientação a objetos.

Na programação estruturada, um programa é composto por três tipos básicos de estruturas:

- sequências: são os comandos a serem executados
- condições: sequências que só devem ser executadas se uma condição for satisfeita (exemplos: if-else, switch e comandos parecidos)
- repetições: sequências que devem ser executadas repetidamente até uma condição for satisfeita (for, while, do-while etc)

Essas estruturas são usadas para processar a entrada do programa, alterando os dados até que a saída esperada seja gerada. Até aí, nada que a programação orientada a objetos não faça, também, certo?

A diferença principal é que na programação estruturada, um programa é tipicamente escrito em uma única rotina (ou função) podendo, é claro, ser quebrado em subrotinas. Mas o fluxo do programa continua o mesmo, como se pudéssemos copiar e colar o código das subrotinas diretamente nas rotinas que as chamam, de tal forma que, no final, só haja uma grande rotina que execute todo o programa.

Além disso, o acesso às variáveis não possui muitas restrições na programação estruturada. Em linguagens fortemente baseadas nesse paradigma, restringir o acesso à uma variável se limita a dizer se ela é visível ou não dentro de uma função (ou módulo, como no uso da palavra-chave static, na linguagem C), mas não se consegue dizer de forma nativa que uma variável pode ser acessada por apenas algumas rotinas do programa. O contorno para situações como essas envolve práticas de programação danosas ao desenvolvimento do sistema, como o uso excessivo de variáveis globais. Vale lembrar que variáveis globais são usadas tipicamente para manter estados no programa, marcando em qual parte dele a execução se encontra.

A programação orientada a objetos surgiu como uma alternativa a essas características da programação estruturada. O intuito da sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real, daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.

Esse novo paradigma se baseia principalmente em dois conceitos chave: classes e objetos. Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.

3.1.1 Encapsulamento, herança e polimorfismo: as principais características da POO

As duas bases da POO são os conceitos de classe e objeto. Desses conceitos, derivam alguns outros conceitos extremamente importantes ao paradigma, que não só o definem como são as soluções de alguns problemas da programação estruturada. Os conceitos em questão são o *encapsulamento*, a *herança*, as *interfaces* e o *polimorfismo*.

3.1.2 Encapsulamento

Ainda usando a analogia do carro, sabemos que ele possui atributos e métodos, ou seja, características e comportamentos. Os métodos do carro, como acelerar, podem usar atributos e outros métodos do carro como o tanque de gasolina e o mecanismo de injeção de combustível, respectivamente, uma vez que acelerar gasta combustível.

No entanto, se alguns desses atributos ou métodos forem facilmente visíveis e modificáveis, como o mecanismo de aceleração do carro, isso pode dar liberdade para que alterações sejam feitas, resultando em efeitos colaterais imprevisíveis. Nessa analogia, uma pessoa pode não estar satisfeita com a aceleração do carro e modifica a forma como ela ocorre, criando efeitos colaterais que podem fazer o carro nem andar, por exemplo.

Dizemos, nesse caso, que o método de aceleração do seu carro não é visível por fora do próprio carro. Na POO, um atributo ou método que não é visível de fora do próprio objeto é chamado de "privado" e quando é visível, é chamado de "público".

Mas então, como sabemos como o nosso carro acelera? É simples: não sabemos. Nós só sabemos que para acelerar, devemos pisar no acelerador e de resto o objeto sabe como executar essa ação sem expor como o faz. Dizemos que a aceleração do carro está *encapsulada*, pois sabemos o que ele vai fazer ao executarmos esse método,

mas não sabemos como - e na verdade, não importa para o programa como o objeto o faz, só que ele o faça.

Isso vale para atributos. Por exemplo: não sabemos como o carro sabe qual velocidade mostrar no velocímetro ou como ele calcula sua velocidade, mas não precisamos saber como isso é feito. Só precisamos saber que ele vai nos dar a velocidade certa. Ler ou alterar um atributo encapsulado pode ser feito a partir de *getters e setters* (colocar referência).

Esse *encapsulamento* de atributos e métodos impede o chamado *vazamento de escopo*, onde um atributo ou método é visível por alguém que não deveria vê-lo, como outro objeto ou classe. Isso evita a confusão do uso de variáveis globais no programa, deixando mais fácil de identificar em qual estado cada variável vai estar a cada momento do programa, já que a restrição de acesso nos permite identificar quem consegue modificá-la.

3.1.3 Herança

No nosso exemplo, você acabou de comprar um carro com os atributos que procurava. Apesar de ser único, existem carros com exatamente os mesmos atributos ou formas modificadas. Digamos que você tenha comprado o modelo Fit, da Honda. Esse modelo possui uma outra versão, chamada WR-V (ou "Honda Fit Cross Style"), que possui muitos dos atributos da versão clássica, mas com algumas diferenças bem grandes para transitar em estradas de terra: o motor é híbrido (aceita álcool e gasolina), possui um sistema de suspensão diferente, e vamos supor que além disso ele tenha um sistema de tração diferente (tração nas quatro rodas, por exemplo). Vemos então que não só alguns atributos como tambm alguns mecanismos (ou métodos, traduzindo para POO) mudam, mas essa versão "cross" ainda é do modelo Honda Fit, ou melhor, *é um tipo* do modelo.

3.1.4 Polimorfismo

Vamos dizer que um dos motivos de você ter comprado um carro foi a qualidade do sistema de som dele. Mas, no seu caso, digamos que a reprodução só pode ser feita via rádio ou *bluetooth*, enquanto no seu antigo carro, podia ser feita apenas via cartão SD e *pendrive*. Em ambos os carros está presente o método "tocar música”, mas, como o sistema de som deles é diferente, a forma como o carro toca as músicas é diferente. Dizemos que o método "tocar música" é uma forma de polimorfismo, pois dois objetos, de duas classes diferentes, têm um mesmo método que é implementado de formas diferentes, ou seja, um método possui *várias formas*, várias implementações diferentes em classes diferentes, mas que possuem o mesmo efeito ("polimorfismo" vem do grego *poli* = muitas, *morphos* = forma).

3.2 Banco de Dados

Um banco de dados é uma coleção organizada de informações ou dados estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS). Juntos, os dados e o DBMS, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.

Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o

processamento e a consulta de dados eficientes. Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados.

3.2.1 O que é SQL (Structured Query Language, Linguagem de Consulta Estruturada)?

SQL é uma linguagem de programação usada por quase todos os bancos de dados relacionais para consultar, manipular e definir dados e fornecer controle de acesso. O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI; o SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.

3.2.2 Bancos de dados relacionais

Bancos de dados relacionais se tornaram dominantes na década de 1980. Os itens em um banco de dados relacional são organizados como um conjunto de tabelas com colunas e linhas. A tecnologia de banco de dados relacional fornece a maneira mais eficiente e flexível de acessar informações estruturadas.

3.2.3 Bancos de dados orientados a objetos

As informações em um banco de dados orientado a objetos são representadas na forma de objetos, como na programação orientada a objetos.

3.2.4 Bancos de dados distribuídos

Um banco de dados distribuído consiste em dois ou mais arquivos localizados em sites diferentes. O banco de dados pode ser armazenado em vários computadores, localizados no mesmo local físico ou espalhados por diferentes redes.

3.2.5 Data warehouses

Um repositório central de dados, um data warehouse é um tipo de banco de dados projetado especificamente para consultas e análises rápidas.

3.2.6 Bancos de dados NoSQL

Um NoSQL, ou banco de dados não relacional, permite que dados não estruturados e semiestruturados sejam armazenados e manipulados (em contraste com um banco de dados relacional, que define como todos os dados inseridos no banco de dados devem ser compostos). Os bancos de dados NoSQL se tornaram populares à medida que os aplicativos web se tornaram mais comuns e mais complexos.

3.2.7 Bancos de dados gráficos

Um banco de dados gráfico armazena dados em termos de entidades e os relacionamentos entre entidades.

Bancos de dados OLTP: Um banco de dados OLTP é um banco de dados rápido e analítico projetado para muitas transações realizadas por vários usuários.

Esses são apenas alguns dos vários tipos de bancos de dados em uso atualmente. Outros bancos de dados menos comuns são adaptados para funções científicas, financeiras ou outras muito específicas. Além dos diferentes tipos de banco de dados, as mudanças nas abordagens de desenvolvimento de tecnologia e os avanços dramáticos, como a nuvem e a automação, estão impulsionando os bancos de dados em direções totalmente novas. Alguns dos mais recentes bancos de dados incluem

3.2.8 Bancos de dados de código aberto

Um sistema de banco de dados de código aberto é aquele cujo código-fonte é código aberto; esses bancos de dados podem ser bancos de dados SQL ou NoSQL.

3.2.9 Bancos de dados em nuvem

Um banco de dados em nuvem é uma coleção de dados, estruturados ou não estruturados, que residem em uma plataforma de computação em nuvem privada, pública ou híbrida. Existem dois tipos de modelos de banco de dados em nuvem: tradicional e banco de dados como serviço (DBaaS). Com o DBaaS, as tarefas administrativas e a manutenção são executadas por um provedor de serviços.

3.2.10 Banco de dados multimodelo

Bancos de dados multimodelo combinam diferentes tipos de modelos de banco de dados em um back-end único e integrado. Isso significa que eles podem acomodar vários tipos de dados.

3.2.11 Banco de dados de documentos/JSON

Projetado para armazenamento, recuperação e gerenciamento de informações orientadas a documentos, os bancos de dados de documentos são uma maneira moderna de armazenar dados no formato JSON, em vez de linhas e colunas.

3.2.12 Bancos de dados autônomos

Os bancos de dados independentes mais novos e inovadores (também conhecidos como bancos de dados autônomos) são baseados em nuvem e usam machine learning para automatizar o ajuste de banco de dados, segurança, backups, atualizações e outras tarefas de gerenciamento de rotina tradicionalmente executadas por administradores de banco de dados.

3.3 O que é MySQL

Uma empresa sueca chamada MySQL AB desenvolveu o MySQL em 1994. Então, a companhia norte-americana Sun Microsystems obteve controle total do software ao comprar a MySQL AB em 2008.

Já em 2010, a gigante Oracle, também norte-americana, por sua vez comprou a Sun Microsystems, e o MySQL MySQL tem sido da Oracle desde então.

Quanto a sua definição, MySQL é um Banco de Dados relacional (RDBMS – Relational Database Management Systems) com um modelo de cliente-servidor.

RDBMS é um software de código aberto ou serviço usado na criação e gerenciamento de bancos de dados baseados no modelo relacional. Agora vamos analisar cada termo.

3.4 Modelo Cliente-Servidor

Computadores que rodam softwares de RDBMS são chamados clientes. Sempre que precisam de dados, eles se conectam a um servidor RDBMS.

MySQL é um dos muitos clientes RDBMS disponíveis. RDBMS e MySQL são muitas vezes confundidos como sendo a mesma coisa devido a popularidade do MySQL.

Algumas aplicações de renome, como Facebook, Twitter, YouTube, Google e Yahoo! utilizam MySQL para o armazenamento de dados.

Mesmo que tenha sido inicialmente criada para uso ilimitado, atualmente é ele compatível com muitas plataformas importantes como Linux, MacOS, Microsoft Windows e Ubuntu.

4 PLANO DE DESENVOLVIMENTO

O presente projeto tem como objetivo o desenvolvimento de uma aplicação mobile em Kotlin, voltada especificamente para pessoas com deficiência visual. A proposta central da aplicação é utilizar inteligência artificial para identificar produtos, proporcionando maior autonomia e independência aos usuários durante suas compras.

A aplicação utiliza a câmera do smartphone para capturar a imagem do produto desejado. Em seguida, por meio de técnicas de processamento de imagens e inteligência artificial previamente treinada com uma variedade de produtos, o sistema será capaz de identificar a marca e o tipo do produto em questão. Essas informações serão então transmitidas ao usuário por meio de áudio, permitindo que ele tenha acesso às informações necessárias sem depender exclusivamente da visão.

A acessibilidade do design foi cuidadosamente considerada durante o desenvolvimento do aplicativo para pessoas com deficiência visual. A interface foi projetada de forma a atender às necessidades específicas desse público, garantindo uma experiência de uso mais fácil e intuitiva. Além disso, o design do aplicativo foi cuidadosamente elaborado para ser compatível com tecnologias assistivas, como leitores de tela. Isso significa que os usuários com deficiência visual podem acessar

todas as informações do aplicativo de maneira eficiente, graças ao uso adequado de tags e descrições para os elementos visuais. Dessa forma, o aplicativo garante uma interação mais inclusiva e acessível para pessoas com deficiência visual, promovendo sua autonomia e independência durante as compras.

A acessibilidade do design foi cuidadosamente considerada durante o desenvolvimento do aplicativo para pessoas com deficiência visual. A interface foi projetada de forma a atender às necessidades específicas desse público, garantindo uma experiência de uso mais fácil e intuitiva. Além disso, o design do aplicativo foi cuidadosamente elaborado para ser compatível com tecnologias assistivas, como leitores de tela. Isso significa que os usuários com deficiência visual podem acessar todas as informações do aplicativo de maneira eficiente, graças ao uso adequado de tags e descrições para os elementos visuais. Dessa forma, o aplicativo garante uma interação mais inclusiva e acessível para pessoas com deficiência visual, promovendo sua autonomia e independência durante as compras.

A importância desses recursos vai além da simples identificação de produtos durante as compras. Eles contribuem para a inclusão e independência das pessoas com deficiência visual, permitindo que elas participem ativamente do processo de compra, façam escolhas informadas e tenham maior controle sobre suas decisões. Além disso, o acesso às informações por meio de áudio possibilita uma experiência mais imersiva e inclusiva, garantindo que todas as pessoas, independentemente de suas habilidades visuais, tenham igualdade de oportunidades no contexto de compras em ambientes comerciais.

4.1 Ambiente de Desenvolvimento

O IntelliJ IDEA é um ambiente de desenvolvimento integrado que visa aumentar a produtividade dos desenvolvedores (Java/Kotlin). Com isso, automatiza as tarefas de desenvolvimento que antes eram tediosas e repetitivas. A IDE possui recursos avançados na análise de código, sendo cada parte voltada para ter a melhor qualidade de código possível, de maneira mais rápida e eficiente.

O IntelliJ oferece uma versão integrada que controla sistemas e uma ampla variedade de idiomas suportados. Além disso, oferece excelente assistência no quadro de codificação e características de aumento de produtividade para Java EE, Spring, GWT, Grails, Play e outras estruturas. Claro, também há suporte no desenvolvimento mobile e web.

O IntelliJ permite que o usuário escreva códigos em Java e Kotlin, com mais qualidade e rapidez, contando com um sistema que identifica possíveis erros no código e fornece sugestões para que o usuário possa melhorar seu programa. Isso possibilita

que o usuário aprenda enquanto trabalha. Contando com um ambiente colaborativo e remoto, permite-se que o usuário compartilhe seu projeto com sua equipe em tempo real. Criando sessões em grupo para compartilhar o andamento do projeto e fazer revisões dele, por exemplo. E tudo isso é feito em um ambiente completamente remoto e prático.

4.2 Interface Gráfica

A interface gráfica será desenvolvida fazendo uso da biblioteca gráfica Swing. Todas as JRE, a partir da 1.2, trazem esta biblioteca como o padrão gráfico. Sua principal vantagem reside no fato de ser multiplataforma. Permitindo que as interfaces possam ser utilizadas em qualquer sistema operacional que suporte aJRE, apresentando os componentes da janela com os mesmos atributos (cores, tamanhos, margens, espaçamento etc.), independentemente da plataforma em que está sendo executada.

4.3 Leitor de Imagens

Para a realização deste projeto a leitura de imagens será feita através de imagens adquiridas no google, onde serão cadastradas e será feita a leitura e comparação com as que existem no banco de dados.

4.4 Computador e Sistema Operacional

As configurações do Notebook utilizado neste projeto, tanto para o desenvolvimento do sistema quanto em seu teste, estão na tabela abaixo.

Notebook Lenovo Ideapad S145	
Processador	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Memória RAM	12,0 GB (utilizável: 9,88 GB)
HD	1 tb
Monitor	15.6 polegadas
Sistema Operancional	Windows 11 Home Single Language 64 bits

Fonte: Elaborada pelos autores

4.5 Banco de Dados

Como o SDK utilizado no sistema somente permite a armazenamento dos templates no banco de dados proprietário da Neurotechnology, este será o utilizado. Além dos templates será preciso armazenar os dados dos usuários, como seu nome e seu nome de usuário. Isto será realizado através do salvamento do objeto do usuário em um arquivo. Este arquivo será armazenado em formato binário, utilizando a funcionalidade de Serialização disponibilizada pelo Kotlin.

Desta forma o sistema terá dois bancos de dados, um contendo os templates e outro os dados dos usuários. Os dois serão arquivos e estarão armazenadas na pasta do sistema. Desta forma torna extremamente fácil mover a aplicação de um smartphone para outro, ou até mesmo, no futuro, centralizar o banco de dados em uma única máquina que terá conexão com várias aplicações de cadastro e validação de usuário.

5 PROJETO

Optamos por adotar o padrão de arquitetura MVVM (Model-View-ViewModel) para o nosso projeto, uma escolha fundamentada na busca por uma estrutura organizada e eficiente. Este modelo proporciona uma clara separação de responsabilidades entre a lógica de dados (Model), a interface do usuário (View) e a lógica de apresentação (ViewModel). A introdução da "ViewModel" como intermediária entre a "View" e o "Model" permite uma gestão mais eficaz da apresentação de dados, facilitando o desenvolvimento, manutenção e testes do aplicativo.

5.1 Model

O Model é uma componente fundamental na arquitetura de software e representa a camada responsável pela manipulação e gestão dos dados, regras de negócios e lógica do aplicativo. Ele encapsula a estrutura dos dados, interações com o banco de dados e operações necessárias para garantir a integridade e consistência das informações dentro do sistema. Em resumo, o Model concentra-se nas funcionalidades relacionadas aos dados e na implementação das regras específicas do domínio do aplicativo.

5.2 View

A View, como a interface visual do aplicativo, é composta por elementos interativos, como botões e listas, desempenhando o papel crucial de apresentar informações e capturar interações do usuário. Em paralelo, a ViewModel atua como uma camada intermediária, conectando a View ao sistema. Ela prepara dados para a exibição na View, interagindo com as Models para obter e atualizar informações. A ViewModel gerencia a lógica de apresentação, traduzindo dados brutos para a interface do usuário e lidando com comandos iniciados pela View, garantindo uma interação eficaz.

Essa comunicação entre View e ViewModel ocorre por meio do padrão de Data Binding, permitindo a vinculação direta de propriedades. Isso possibilita atualizações automáticas na interface do usuário quando os dados na ViewModel mudam, proporcionando uma experiência dinâmica e em tempo real para o usuário. Essa abordagem não apenas simplifica a implementação da interface, mas também promove

a reutilização de código e facilita os testes unitários, uma vez que a lógica da interface do usuário está isolada na ViewModel, contribuindo para uma arquitetura mais robusta e de fácil manutenção.

Data Binding é uma técnica que estabelece uma conexão automática e bidirecional entre os elementos da interface do usuário e os dados subjacentes em um aplicativo. Essa abordagem permite que as alterações nos dados sejam refletidas automaticamente na interface do usuário e vice-versa, sem a necessidade de intervenção manual. O data binding simplifica o desenvolvimento de aplicativos, melhora a consistência na exibição de informações e aumenta a eficiência ao automatizar a sincronização entre os componentes da interface e os dados associados.

5.3 ViewModel

A ViewModel é uma camada intermediária vital na arquitetura do sistema, conectando a View às demais partes. Sua função central é preparar dados para exibição na interface do usuário, mantendo uma clara separação entre lógica de apresentação e implementação de interface. Interage principalmente com as Models para obter e atualizar dados, gerenciando comandos da View, como cliques em botões. Destaca-se por notificar automaticamente a View sobre mudanças nos dados, utilizando mecanismos como Data Binding, garantindo uma representação precisa e em tempo real da interface do usuário.

5.4 MVVM

MVVM, ou Model-View-ViewModel, é um padrão de arquitetura de software que separa a lógica de apresentação (ViewModel) da interface do usuário (View) e dos dados (Model). A "ViewModel" atua como um intermediário, preparando os dados para exibição na "View" e gerenciando a interação entre a "View" e o "Model". Essa abordagem promove uma estrutura mais organizada, facilitando o desenvolvimento e a manutenção de aplicativos, especialmente em ambientes que exigem interfaces de usuário interativas e reativas. O Model representa os dados e as regras de negócios, a View lida com a apresentação e a interface do usuário, e o Controller atua como um intermediário, manipulando as interações do usuário e atualizando o Model e a View conforme necessário.

5.5 BarcodeModel

O BarcodeModel, ou Modelo de Coordenação de Identificação de Códigos de Barras, é essencial para o sistema, concentrando-se na administração do processo de identificação de códigos de barras. Suas responsabilidades incluem a habilidade de

empregar bibliotecas especializadas ou serviços dedicados para essa tarefa. Além disso, o BarcodeModel desempenha um papel crucial na eficiente coordenação entre as diferentes partes do sistema.

Dependendo do design adotado, o BarcodeModel pode invocar métodos específicos da Model de Reconhecimento de Produtos para obter detalhes adicionais sobre os produtos relacionados aos códigos de barras identificados. Essa interação dinâmica aprofunda a compreensão dos produtos em questão. Além disso, o BarcodeModel desempenha uma função fundamental na transmissão de dados, fornecendo os códigos de barras identificados à Model de Lista de Produtos. Essa transferência de informações é vital para assegurar uma atualização precisa e oportuna da lista de produtos, contribuindo assim para a integração efetiva das informações no ecossistema do aplicativo e proporcionando uma experiência de usuário coesa e atualizada.

5.6 AIModel

O Modelo de Reconhecimento de Produtos (AIModel) desempenha um papel central no aplicativo, sendo responsável por implementar a lógica de inteligência artificial para identificar produtos a partir de imagens ou dados. Utiliza algoritmos avançados, como processamento de imagem e aprendizado de máquina, para essa finalidade. Dependendo da eficiência, pode recorrer a bibliotecas ou serviços externos de inteligência artificial.

Além disso, o AIModel desempenha um papel crucial na comunicação, fornecendo informações detalhadas sobre produtos identificados para a Model de Coordenação de Identificação de Códigos de Barras e a Model de Lista de Produtos. Essa interconexão é fundamental para garantir a integração eficaz das informações sobre produtos no ecossistema do aplicativo, proporcionando uma experiência unificada ao usuário.

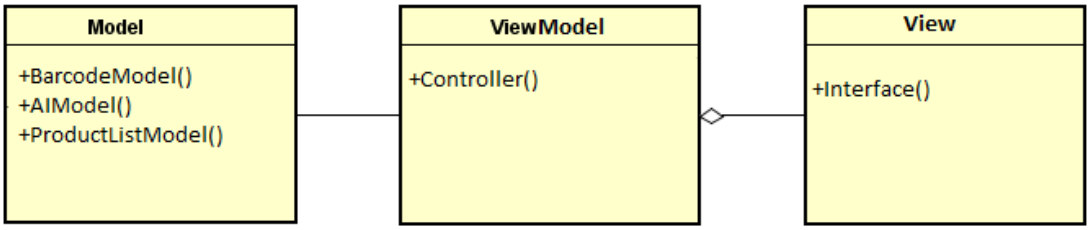
5.7 ProductListModel

O Modelo de Lista de Produtos, denominado ProductListModel, assume um papel central no gerenciamento da lista de produtos identificados pelo aplicativo. Suas responsabilidades abrangem operações como adição, remoção e atualização de produtos na lista, garantindo uma gestão eficiente e dinâmica.

No que diz respeito às relações, o ProductListModel recebe informações sobre produtos tanto da Model de Reconhecimento de Produtos quanto da Model de Coordenação de Identificação de Códigos de Barras. Essa troca de dados é fundamental

para manter a lista de produtos devidamente atualizada. Além disso, o modelo tem a capacidade de notificar a interface do usuário sobre quaisquer alterações na lista, proporcionando uma experiência interativa e em tempo real. Essa capacidade de atualização instantânea contribui para uma interface de usuário dinâmica e responsiva, refletindo com precisão as mudanças na lista de produtos.

5.8 Diagrama de Classes:



Fonte: Elaborada pelo autor

6 RELATÓRIO COM AS LINHAS DE CÓDIGO DO PROJETO

Ao longo deste texto, você percorrerá e entenderá o design do aplicativo e como ele é composto. No código a seguir, é possível observar que a estrutura do layout é criada usando XML (Extensible Markup Language). A estruturação por layouts é bastante simples e oferece uma ampla variedade de opções para a construção das interfaces, além de facilitar a legibilidade do código. Observe agora a parte inicial da nossa interfade:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="414dp"
        android:layout_height="733dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        tools:layout_editor_absoluteY="1dp">

        <ImageView
            android:id="@+id/MyLogo"
            android:layout_width="match_parent"
            android:layout_height="129dp"
            app:srcCompat="@drawable/ic_launcher_foreground"
            tools:ignore="MissingConstraints" />

        <VideoView
            android:id="@+id/videoView"
            android:layout_width="match_parent"
            android:layout_height="434dp"
            tools:ignore="MissingConstraints" />
```

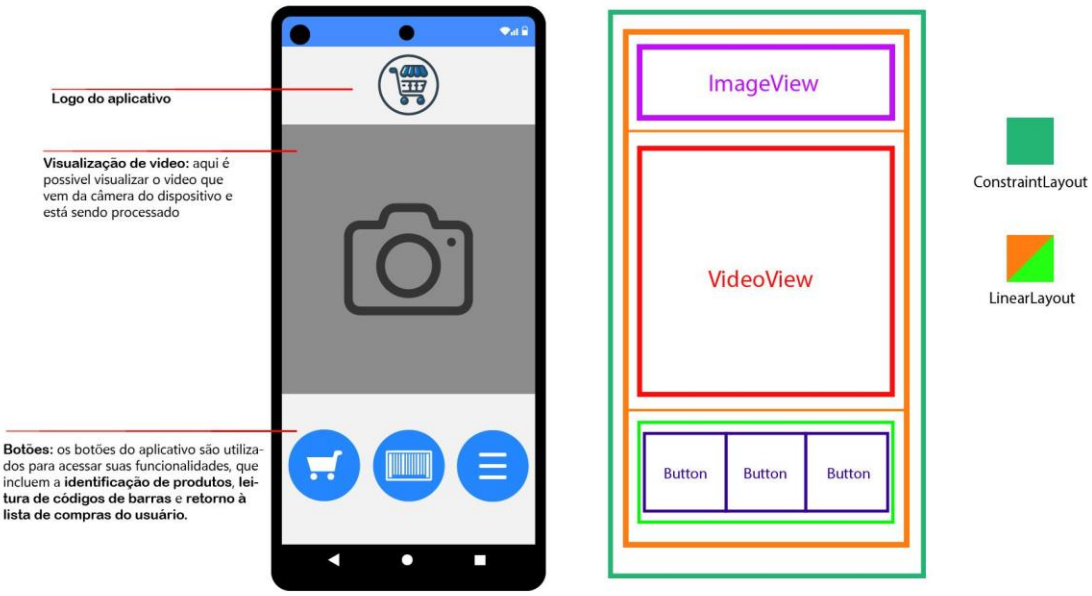
Fonte:Elaborada pelos autores

Logo após a declaração do XML, percebemos a presença do invólucro principal. Essa declaração inicial serve como base para todos os outros invólucros. Adicionamos um <LinearLayout/> com orientação vertical, que armazenará todo o conteúdo do aplicativo. Dentro desse container, incluímos um <ImageView>, um componente utilizado para adicionar imagens ao layout, onde colocaremos a logo do aplicativo. Também adicionamos um <VideoView> para possibilitar a análise da imagem necessária para o reconhecimento do produto.

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:orientation="horizontal"
            tools:ignore="MissingConstraints">

            <Button
                android:id="@+id/button"
                android:layout_width="match_parent"
                android:layout_height="122dp"
                android:layout_margin="8dp"
                android:layout_weight="1"
                android:text="Product" />

            <Button
                android:id="@+id/button2"
                android:layout_width="match_parent"
                android:layout_height="122dp"
                android:layout_margin="8dp"
                android:layout_weight="1"
                android:text="Bar code" />

            <Button
                android:id="@+id/button3"
                android:layout_width="match_parent"
                android:layout_height="122dp"
                android:layout_margin="8dp"
                android:layout_weight="1"
                android:text="My list" />
        </LinearLayout>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Fonte: Elaborada pelos autores

Após o <VideoView>, inserimos outro <LinearLayout>, desta vez na orientação horizontal, para posicionar os três botões de ação lado a lado, sendo assim a ideia final do layout do aplicativo após todo processo de estilização é essa:



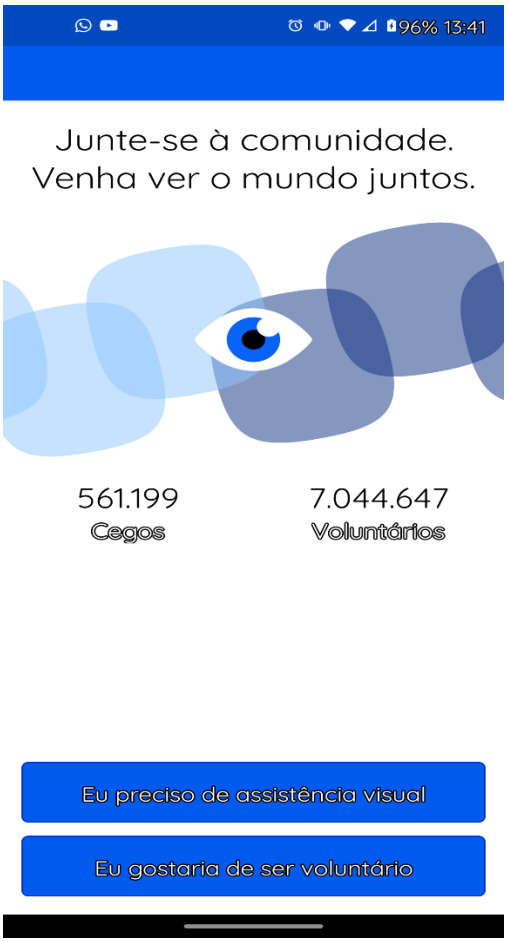
Fonte: Elaborada pelos autoes

A ideia principal é que as ações do aplicativo sejam acionadas pelos botões, cada um com uma função específica. Essas funções incluem o reconhecimento do produto a partir de sua imagem, o reconhecimento e leitura do código de barras, e o retorno da listagem de compras. Todas essas ações são baseadas nas imagens capturadas pela câmera do dispositivo, que está posicionada no centro do layout. Para implementar essa lógica, adicionamos botões apropriados ao nosso segundo <LinearLayout> na orientação horizontal. Cada botão terá sua própria função específica associada a uma ação relevante no aplicativo. A estrutura do código XML continuará a se desenvolver para acomodar esses elementos e suas respectivas funcionalidades.

O objetivo é proporcionar ao usuário uma experiência intuitiva e eficiente, onde as funcionalidades do aplicativo são facilmente acessíveis através desses botões de ação, utilizando as capacidades da câmera do dispositivo para obter informações relevantes.

7 APRESENTAÇÃO DO PROGRAMA EM FUNCIOINAMENTO EM UM CONPUTADOR

A interface do aplicativo ACESSMARKET é simples e intuitiva, permitindo que os usuários solicitem ou ofereçam ajuda visual de forma rápida e fácil. A interface consiste em quatro telas principais: a tela inicial, a tela de chamada, a tela de avaliação e a tela de configurações.



Fonte: Elaborada pelos autores

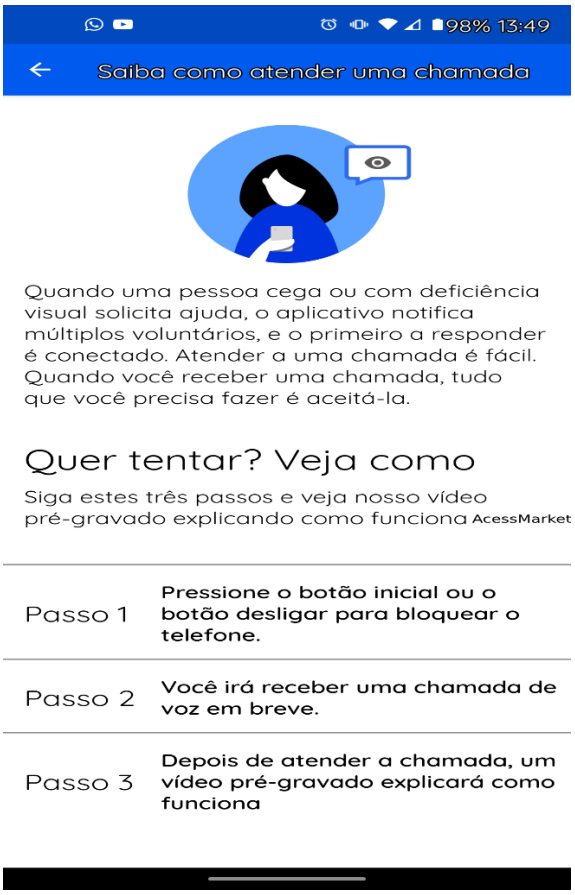
7.1 Tela Inicial

A tela inicial é a primeira tela que aparece quando o usuário abre o aplicativo. Nesta tela, o usuário pode escolher se quer pedir ajuda ou oferecer ajuda, tocando em um dos dois botões grandes. O usuário também pode ver o número de usuários cegos/com visão limitada e voluntários que estão online no momento, bem como o número de chamadas realizadas até agora. No canto superior direito, há um ícone de sino que leva à tela de notificações, onde o usuário pode ver as últimas novidades e atualizações do aplicativo. No canto superior esquerdo, há um ícone de menu que leva à tela de configurações, onde o usuário pode personalizar suas preferências e opções.

Fonte: Elaborada pelos autores

7.2 Tela de Chamada

A tela de chamada é a tela que aparece quando o usuário solicita ou recebe uma chamada de vídeo ao vivo. Nesta tela, o usuário pode ver e ouvir a pessoa que está do outro lado da linha, e conversar com ela usando o microfone do dispositivo. O usuário também pode usar alguns recursos adicionais, como o flash, o zoom, o mudo e o viva-voz, tocando nos ícones correspondentes na parte inferior da tela. No canto superior direito, há um ícone de ponto de interrogação que leva à tela de ajuda, onde o usuário pode encontrar dicas e orientações sobre como usar o aplicativo. No canto superior esquerdo, há um ícone de telefone vermelho que permite ao usuário encerrar a chamada.



Fonte: Elaborada pelos autores

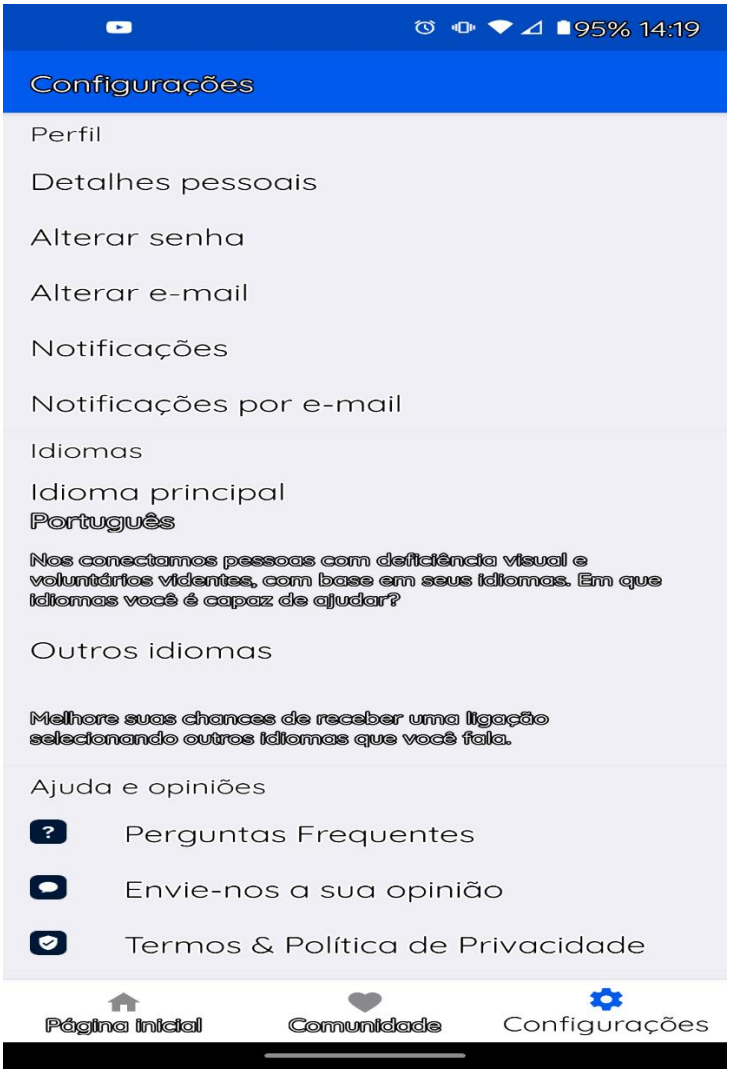
7.3 Tela de Avaliação

A tela de avaliação é a tela que aparece depois que o usuário encerra uma chamada. Nesta tela, o usuário pode avaliar a qualidade da chamada, dando uma nota de uma a cinco estrelas. O usuário também pode deixar um comentário opcional, agradecendo ou elogiando a pessoa que o ajudou. No canto superior direito, há um ícone de compartilhamento que permite ao usuário compartilhar sua experiência nas redes sociais. No canto superior esquerdo, há um ícone de seta que leva de volta à tela inicial.

Fonte: Elaborada pelos autores

7.4 Tela de Configuração

A tela de configurações é a tela que permite ao usuário personalizar suas preferências e opções no aplicativo. Nesta tela, o usuário pode alterar seu idioma, seu fuso horário, seu perfil, sua senha, sua privacidade, suas notificações, seu som, sua acessibilidade, seu feedback, sua ajuda especializada e sua assinatura. O usuário também pode ver seus dados pessoais, seus históricos de chamadas, seus convites, seus amigos, seus parceiros, seus termos de uso, sua política de privacidade, seu contato e sua versão do aplicativo. No canto superior direito, há um ícone de saída que permite ao usuário sair do aplicativo. No canto superior esquerdo, há um ícone de seta que leva de volta à tela inicial.



Fonte: Elaborada pelos autores

8 REFFERÊNICAS

BERSCH, Rita. Introdução à Tecnologia Assistiva. Porto Alegre: CEDI**, v. 21, 2008. Acesso em: 11 mai. 2023.**

CARLOS MACORATTI, José. .Net – O padrão MVVM (Model-View-ViewModel) Revisado, **Disponível em:** .NET - O padrão MVVM (Model-View-ViewModel) revisitado (macoratti.net) **v. 15, 12, 2015. Acesso em: 16 nov. 2023**

Developer Android. Conheça a linguagem de programação Kotlin. **Disponível em**: https://developer.android.com/kotlin/learn?hl=pt-br. **Acesso em: 04 mai. 2023**

HENRIQUE,João. Programação orientada a objetos e programação estruturada.**Disponível em:** https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos **Acesso em: 04 mai. 2023**

L, Andrei. O que é mysql, **28 jul 2023. Disponível em:** https://www.hostinger.com.br/tutoriais/o-que-e-mysql **Acesso em: 04 mai. 2023**

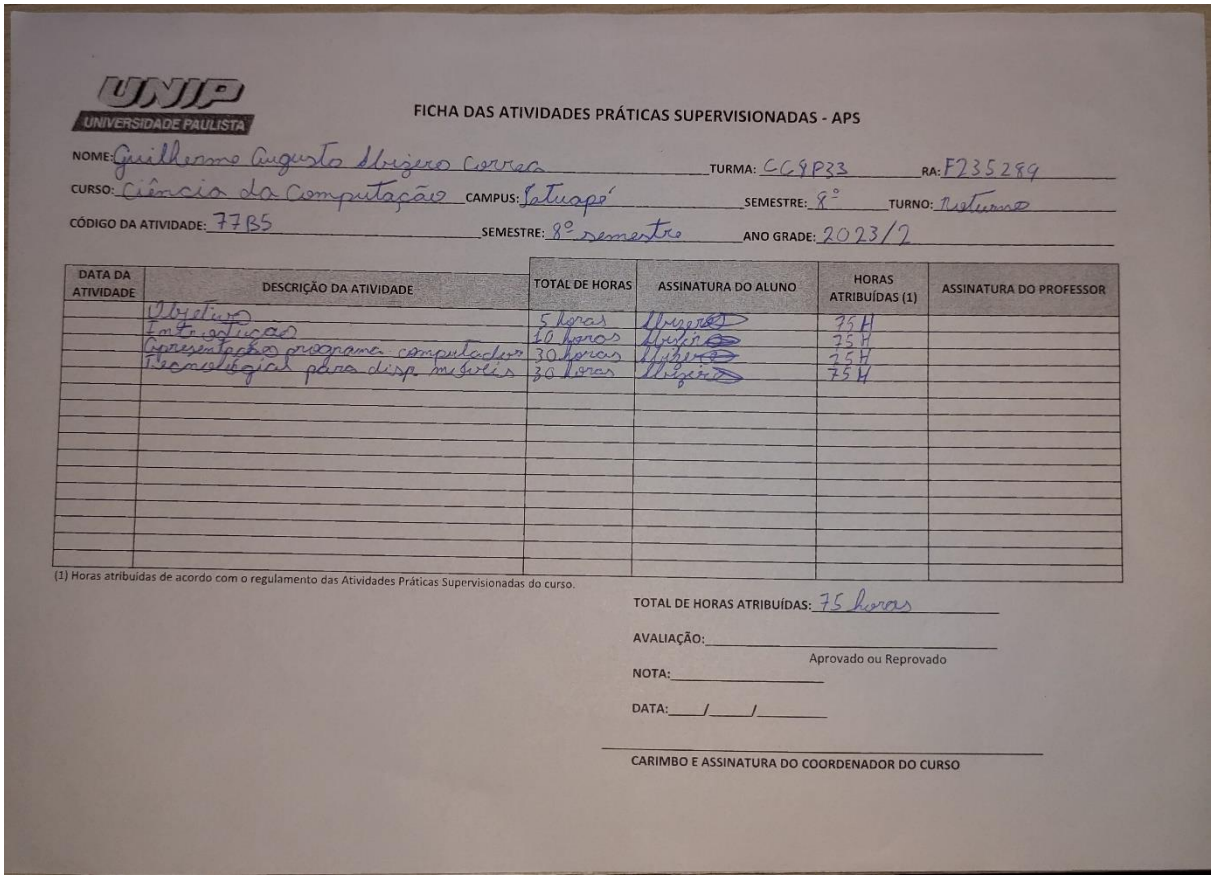
LEE, Kai-Fu. Inteligência artificial. 2019. Acesso em: 03 mai. 2023

LOPES, Ana Carolina Aoki. Análise de acessibilidade para pessoas cegas às embalagens. 2015. **Acesso em: 03 mai. 2023.**

OMS. World report on vision (Relatório Mundial sobre visão). **Disponível em:** http://www.who.int/publications/i/item/9789241516570. **Acesso em: 14 mai. 2023.**

ORACLE. what is database. **Disponível em:** https://www.oracle.com/br/database/what-is-database/. **Acesso em: 08 set. 2023.**

9 FICHAS DA ATIVIDADE PRÁTICA SUPERFISONADA



UNIP
UNIVERSIDADE PAULISTA

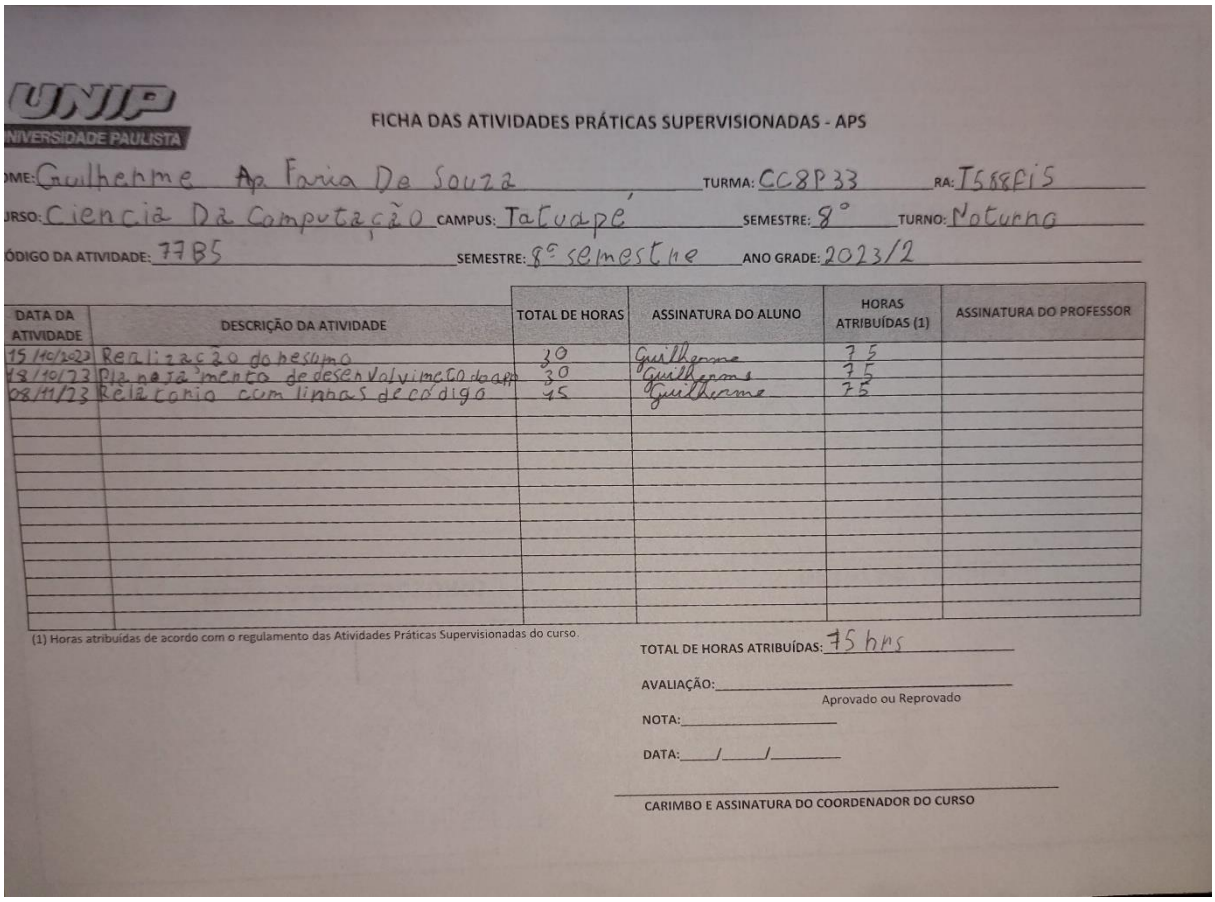
FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Guilherme Augusto Moreira Correa TURMA: CC8P33 RA: F235284
CURSO: Ciência da Computação CAMPUS: Tatuapé SEMESTRE: 8° TURNO: Noturno
CÓDIGO DA ATIVIDADE: 77B5 SEMESTRE: 8° semestre ANO GRADE: 2023/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
	Objetivo	5 horas	Guilherme	75H	
	Introdução	10 horas	Guilherme	75H	
	Apresentação programa computador	30 horas	Guilherme	75H	
	Tecnológica para disp. móveis	30 horas	Guilherme	75H	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 horas
AVALIAÇÃO: _____ Aprovado ou Reprovado
NOTA: _____
DATA: ____/____/_____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Guilherme Ap. Faria De Souza TURMA: CC8P33 RA: T588FI5
CURSO: Ciencia Da Computação CAMPUS: Tatuapé SEMESTRE: 8° TURNO: Noturno
CÓDIGO DA ATIVIDADE: 77B5 SEMESTRE: 8° semestre ANO GRADE: 2023/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
15/10/2023	Realização do resumo	30	Guilherme	75	
18/10/23	Planejamento de desenvolvimento do app	30	Guilherme	75	
08/11/23	Relatorio com linhas de código	45	Guilherme	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 hrs
AVALIAÇÃO: _____ Aprovado ou Reprovado
NOTA: _____
DATA: ____/____/_____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Luiz Felipe Fernandes Martins TURMA: CC8P33 RA: N675365

CURSO: Ciências da Computação CAMPUS: SEMESTRE: 8º TURNO: Noturno

CÓDIGO DA ATIVIDADE: 7785 SEMESTRE: Oitavo ANO GRADE: 2023/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
15/11/23	Plano de desenvolvimento	30 horas	Luiz Felipe F. Martins		
16/11/23	Projeto	30 horas	Luiz Felipe F. Martins		
16/11/23	Diagrama de classes	15 horas	Luiz Felipe F. Martins		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Pedro Cardoso de Oliveira TURMA: CC8P33 RA: N659172

CURSO: Ciência da computação CAMPUS: TATUAPÉ SEMESTRE: 8º TURNO: NOTURNO

CÓDIGO DA ATIVIDADE: 7785 SEMESTRE: Oitavo ANO GRADE: 2023/2

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
15/11/23	Banco de dados	45 horas	Pedro C.		
16/11/23	Projeto	30 horas	Pedro C.		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 horas

AVALIAÇÃO:
Aprovado ou Reprovado

NOTA:

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO