UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Daniel Felipe Schröder

Sistema de Gestão de Compras para a Empresa Xibuco

Florianópolis
2023

Daniel Felipe Schröder

Sistema de Gestão de Compras para a Empresa Xibuco

Trabalho de Conclusão do Curso de Graduação em Ciência da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina, para a obtenção do título de bacharel em Ciência da Computação.
Orientadora: Profa. Dra. Jerusa Marchi

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Schröder, Daniel Felipe
Sistema de gestão de compras para a empresa Xibuco /
Daniel Felipe Schröder ; orientadora, Jerusa Marchi, 2023.
78 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Ciências da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciências da Computação. 2. Desenvolvimento. 3.
Aplicativo. 4. Kotlin. I. Marchi, Jerusa. II. Universidade
Federal de Santa Catarina. Graduação em Ciências da
Computação. III. Título.

Daniel Felipe Schröder

Sistema de Gestão de Compras para a Empresa Xibuco

Este Trabalho de Conclusão foi julgado adequado para obtenção do Título de "bacharel em Ciência da Computação" e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação.

Florianópolis, 20 de Novembro de 2023.

Profa. Dra. Lúcia Helena Martins Pacheco
Coordenadora do Curso

Banca Examinadora:

Profa. Dra. Jerusa Marchi
Orientadora

Prof. Dr. Elder Rizzon Santos
Avaliador
UFSC

Prof. Dr. Maicon Rafael Zatelli
Avaliador
UFSC

RESUMO

Este trabalho tem como objetivo principal desenvolver um sistema de gestão de compras de roupas, calçados e acessórios, tanto novos quanto seminovos, com foco na otimização do processo dentro da empresa Xibuco Comércio Infantil Online. Para atingir esse objetivo, foram estabelecidos objetivos específicos que serviram como marcos para o projeto. Em primeiro lugar, foi elaborado um modelo de persistência para os dados do sistema de gestão de compras da empresa. Em seguida, foram criados e implementados os serviços de backend necessários para interagir com o modelo de persistência de dados desenvolvido. Além disso, foi desenvolvida uma versão preliminar de um aplicativo móvel destinado aos clientes da empresa, permitindo a interação com o sistema de gestão de compras e facilitando a venda de produtos para a Xibuco. Por fim, foram especificados os requisitos e a arquitetura de operação de um ambiente de administração do sistema, voltado para os colaboradores da empresa. Esta especificação foi então utilizada para a implementação de uma versão preliminar deste ambiente. Esses objetivos, quando atingidos, abriram espaço para a decisão estratégica da empresa de colocar o sistema em produção, contribuindo para a eficiência e aprimorando o processo de gestão de compras, promovendo uma experiência mais fluida tanto para os clientes quanto para os colaboradores da Xibuco Comércio Infantil Online.

Palavras-chave: sistema, gestão, compras, aplicativo, móvel, requisitos, brechó, comércio, arquitetura

ABSTRACT

This project aims to develop a purchasing management system for clothing, footwear, and accessories, both new and second-hand, with a focus on optimizing the process within the company Xibuco Comércio Infantil Online. To achieve this goal, specific objectives were established that served as milestones for the project. Firstly, a persistence model for the company's purchasing management system data was developed. Subsequently, the necessary backend services were created and implemented to interact with the developed data persistence model. Additionally, a preliminary version of a mobile application for the company's customers was developed, allowing interaction with the purchasing management system and facilitating product sales to Xibuco. Finally, the requirements and operational architecture of a system administration environment, aimed at company employees, were specified. This specification was then used for the implementation of a preliminary version of this environment. When these objectives were achieved, they created room for the company's strategic decision to implement the system in production, contributing to efficiency and improving the purchasing management process, promoting a smoother experience for both customers and employees of Xibuco Comércio Infantil Online.

Keywords: system. management. purchases. mobile. application. requirements. commerce. architecture.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
B2C	Business to Consumer
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBGE	Instituto Brasileiro de Geografia e Estatística
IOT	Internet of Things
JSON	JavaScript Object Notation
LGPD	Lei Geral de Proteção de Dados
MVVM	Model View Viewmodel
PWA	Progressive Web Application
REST	Representational State Transfer
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UML	Unified Modeling Language
URL	Uniform Resource Locator
WCAG	Web Content Accessibility Guidelines
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A indústria têxtil ocupa o segundo lugar no ranking das atividades que mais prejudicam o meio ambiente global. Isso ocorre principalmente devido à sua grande contribuição para as emissões de dióxido de carbono (CO2), que representam cerca de 10% das emissões totais. Além disso, essa indústria produz uma enorme quantidade de resíduos ao longo de sua cadeia de produção e consome quantidades significativas de água na fabricação de roupas e acessórios. (PORTOGENTE, 2020)

A cada ano, estima-se que sejam produzidas cerca de 300 milhões de toneladas de produtos têxteis em todo o mundo, mas apenas 10% desse volume é reciclado. O *fast fashion*, com seus preços baixos e lançamentos frequentes de coleções, agrava ainda mais esse problema, incentivando o consumo excessivo. (PORTOGENTE, 2020)

Como resposta à tendência de consumo excessivo na indústria da moda e aos impactos ambientais que ela causa, existe um crescimento da procura por artigos usados e, como resultado, um aumento na popularidade das lojas de segunda mão, os brechós.

Os brechós surgiram em Paris por volta de 1860 e chegaram ao Rio de Janeiro durante o século XIX com a "Casa de Belchior", uma loja de artigos usados. Devido à dificuldade em pronunciar o nome Belchior, a loja se popularizou como brechó, inspirando outros estabelecimentos semelhantes. No Brasil, os artigos de segunda mão mais vendidos incluem artigos esportivos, joias, carros, discos de vinil, eletrônicos, ferramentas, livros, móveis, roupas de bebê e de grifes. (SEBRAE, 2019)

O setor de segunda mão apresenta um imenso potencial de crescimento, considerando que aproximadamente 70 milhões de brasileiros possuem itens não utilizados em suas casas, e 84% deles expressam o desejo de vendê-los. Projeções indicam que até 2027, a quantidade de roupas de segunda mão em guarda-roupas pode superar a de itens de *fast fashion*. Além disso, no Brasil, o mercado de segunda mão está expandindo a uma taxa 24% mais rápida do que a do varejo tradicional, e entre 2010 e 2015, a quantidade de brechós no país aumentou cerca de 210%. (SEBRAE, 2019)

Roupas de bebês são extremamente vantajosas para esse tipo de comércio, pois costumam ser pouco usadas em virtude do rápido crescimento das crianças. As peças que não servem mais podem ser vendidas, gerando renda para adquirir outras peças maiores. Além disso, peças infantis são frequentemente encontradas com descontos de até 85% nas lojas de brechó.

Neste contexto encontra-se a Xibuco, uma microempresa localizada na região da Grande Florianópolis. A Xibuco atua como um e-commerce varejista especializado em roupas, calçados e acessórios infantis, oferecendo tanto produtos novos quanto seminovos desde setembro de 2018. Seu principal objetivo é reduzir o consumo em massa de roupas infantis, promovendo uma abordagem mais consciente para a compra desses itens. A

empresa tem a ambição de se fortalecer no mercado e se tornar um dos maiores brechós infantis do Brasil. Para alcançar esse objetivo, é essencial que a empresa evolua os seus processos internos, reduzindo custos e aumentando a agilidade no tratamento dos seus produtos. Portanto, o foco deste trabalho é criar e implementar uma versão preliminar de um Sistema de Gestão de Compras para a Empresa Xibuco.

1.2 OBJETIVOS

Os objetivos deste projeto foram separados em duas categorias: geral e específicos. O objetivo geral delineia o propósito fundamental do estudo, enquanto os objetivos específicos estabelecem as metas essenciais para alcançar com êxito o objetivo geral. A seguir, descrevem-se o objetivo geral e os objetivos específicos:

1.2.1 Objetivo geral

O alvo deste trabalho é desenvolver uma versão preliminar de um sistema de gestão de compras de roupas, calçados e acessórios, novos e seminovos, para evoluir este processo dentro da empresa Xibuco Comércio Infantil Online.

1.2.2 Objetivos específicos

Para atingir o objetivo geral deste trabalho, são estabelecidos os seguintes mas marcos:

a) elaborar um modelo de persistência para os dados do sistema de gestão de compras da empresa;

b) criar e implementar os serviços de *backend* para interagir com o modelo de persistência de dados elaborado;

c) desenvolver uma versão preliminar de um aplicativo móvel, a ser utilizado pelos clientes da empresa, para interagir com o sistema de gestão de compras e vender os seus produtos para a Xibuco;

d) especificar os requisitos e a arquitetura de operação, bem como implementar uma versão preliminar de um ambiente de administração do sistema para os colaboradores da empresa.

1.3 JUSTIFICATIVA

A empresa Xibuco Comércio Infantil Online busca se tornar um dos maiores brechós do Brasil e, para que isso ocorra, é necessário que a empresa conquiste uma fatia maior do mercado. Contudo, em se tratando de uma microempresa, os investimento e apostas

realizadas precisam ser acertadas, afinal, expandir um empreendimento é um processo complexo, caótico e que demanda testes e planejamento. (ZOOK; ALLEN, 2003)

A organização em questão foi escolhida devido ao fato de que o autor é o cofundador da empresa. Adicionalmente, ao conduzir o trabalho atual, está ampliando seu entendimento sobre aplicativos móveis em sistemas complexos, o que inevitavelmente beneficiará o seu desempenho profissional em um mercado de trabalho em expansão.

A proposta desse projeto é original e visa desenvolver uma versão preliminar de um sistema de gestão de compras para evoluir os processos dentro da empresa Xibuco Comércio Infantil Online.

Assim, ao final do trabalho, a Xibuco Comércio Infantil Online, terá uma versão preliminar de um novo sistema de gestão de compras, bem fundamentado e estruturado em torno de um novo processo mais eficiente.

Por fim, o resultado deste trabalho contribui como fonte bibliográfica para novos acadêmicos interessados na área de desenvolvimento de aplicativos móveis e pode ser utilizado como base de criação para trabalhos futuros de extensão e evolução do sistema proposto.

1.4 ESTRUTURA DO TRABALHO

A estrutura do trabalho começa por estabelecer o tema que será abordado e por que é importante abordá-lo. Em seguida, define o objetivo geral e seus desdobramentos específicos e fornece a base teórica necessária para sustentar a implementação de um sistema de compras para a empresa Xibuco. Após isso, descreve a metodologia de desenvolvimento que será utilizada e apresenta o projeto elaborado para solucionar o problema apresentado. Por fim, as considerações finais avaliam os resultados obtidos e concluem a apresentação deste trabalho.

2 PROJETO XIBUCO

Este capítulo apresenta brevemente a empresa Xibuco Comércio Infantil Online. Nele descreve-se o modelo de negócio da empresa na seção 2.1, faz-se um resumo do processo de compra de roupas, calçados e acessórios utilizado pela empresa na seção 2.2 e, por fim, na seção 2.3, apresenta-se as principais dificuldades técnicas encontradas neste processo.

2.1 MODELO DE NEGÓCIO

A Xibuco Comércio Infantil Online é uma empresa de comércio eletrônico B2C (*Business to Consumer*) que busca promover estilo e sustentabilidade a partir da comercialização de roupas, calçados e acessórios, novos e seminovos, para crianças de 0 a 6 anos.

Os itens comercializados pela empresa chegam até o brechó através das mães que desejam repassar as peças que não servem mais em seus filhos para outras pessoas. O contato inicial destas mães para a venda dos produtos para a Xibuco ocorre através dos canais públicos de comunicação da empresa, como o *WhatsApp* ou o *Instagram*, por exemplo. A partir deste ponto, o processo para a aquisição de um inventário, que é o nome dado ao conjunto de peças que um cliente deseja vender para a Xibuco, segue uma coleção de regras internas para a pré-classificação e avaliação da viabilidade de compra deste inventário por parte da empresa. Por fim, este inventário segue um roteiro de negociação, transporte e curadoria até que seja de fato adquirido, processado e disponibilizado na loja virtual.

O restante do modelo de negócio segue a tradicional razão de uma loja virtual, cujos detalhes não abrangem o escopo relevante deste trabalho.

2.2 FLUXO DE AQUISIÇÃO DE PRODUTOS

Para a clara compreensão do problema, descreve-se abaixo o fluxo detalhado de aquisição de produtos utilizado pela equipe de comunicação da Xibuco para a compra de inventários:

a) Contato do cliente através do *WhatsApp* ou *Instagram* para a venda de produtos para a Xibuco;

b) Repasse, por parte dos colaboradores da Xibuco, das informações essenciais e das regras de funcionamento do processo de compra de um inventário ao cliente;

c) Envio, por parte do cliente, das informações dos produtos entre roupas, calçados e acessórios que este deseja vender para a empresa;

d) Execução da pré-classificação e avaliação da viabilidade de compra do inventário por parte da Xibuco, levando-se em consideração a quantidade de peças, o tipo e o valor estimado para cada uma dessas peças, a distância do cliente até a empresa, em virtude dos custos de logística reversa, entre outros;

e) Envio, por parte dos colaboradores da Xibuco, de uma proposta inicial para a aquisição do inventário em negociação;

f) Aceitação, por parte do cliente, da proposta inicial;

g) Elaboração e repasse da etiqueta de logística reversa para o cliente;

h) Envio dos produtos do cliente para a Xibuco;

i) Recepção e curadoria das peças recebidas;

j) Envio, por parte dos colaboradores da Xibuco, da proposta final para a aquisição do inventário em negociação;

k) Aceitação, por parte do cliente, da proposta final.

É importante ressaltar que o fluxo previamente descrito apresenta-se de forma bastante simplificada, não levando em consideração os casos negativos e/ou de falta de acordo entre a Xibuco e o cliente durante a negociação.

Além disto, no segmento de brechós, cada produto é único e, mesmo após todos os investimentos em divulgação e marketing, resulta em apenas uma venda, sendo um fator estritamente limitante para a escalabilidade do negócio. Ou seja, é de crucial importância que este processo de aquisição de novos produtos opere dentro da empresa da forma mais eficiente e automatizada possível.

2.3 DIFICULDADES TÉCNICAS

Mantendo-se em vista o fluxo de aquisição de produtos descrito na seção 2.2, imediatamente anterior, podemos identificar uma série de pontos de melhoria no processo atualmente adotado pela empresa. Estes pontos, dada a importância do processo em questão, podem ser vistos como restritivos para o crescimento e para a escalabilidade do negócio. Entre os postos identificados estão:

a) Atendimento não padronizado e altamente dependente do colaborador que realiza a interação com o cliente;

b) Operação com custo elevado, afinal, é necessário que os colaboradores interajam com os clientes para a aquisição de novos inventários;

c) Tempo de resposta elevado para os clientes. Com a situação atual, cada cliente precisa aguardar pelo menos a proposta inicial para decidir seguir com o processo de venda de um inventário ou não;

d) Grande quantidade de tempo desperdiçado em interações com clientes que apenas desejam avaliar as suas peças, sem necessariamente concretizar uma venda para a empresa.

3 FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento do Sistema de Gestão de Compras para a Empresa Xibuco, são apresentadas as perspectivas de diversas fontes e autores acerca do tema proposto. Este capítulo proporciona o embasamento teórico para que se possa compreender com clareza o que este trabalho entrega de resultado, bem como os elementos técnicos envolvidos no levantamento de requisitos do projeto, na elaboração do modelo de persistência dos dados e na escolha das tecnologias utilizadas nos serviços de *backend* e nos aplicativos móveis usados pelos clientes do sistema na empresa.

3.1 ENGENHARIA DE SOFTWARE

A Engenharia de Software compreende todas as etapas do ciclo de vida de um produto de software, partindo da análise dos requisitos do sistema até o projeto, implementação, teste, documentação e manutenção do software ao longo deste seu ciclo de vida. Ela se preocupa não apenas com o código-fonte, mas também com a arquitetura do sistema, com o gerenciamento do projeto, com a garantia da qualidade e com a segurança do produto de software entregue. (SOMMERVILLE, 2015)

De acordo com (SOMMERVILLE, 2015) a Engenharia de Software é uma disciplina da ciência da computação que se concentra na aplicação de princípios sistemáticos, métodos e técnicas para projetar, desenvolver, testar e manter sistemas de software de alta qualidade. Ela visa criar produtos de software de forma eficaz, eficiente e confiável, seguindo padrões e boas práticas que garantam a satisfação das necessidades dos usuários e a entrega de produtos de software de alta qualidade.

3.1.1 Modelos de ciclo de vida de software

Para atingir tais objetivos, (PRESSMAN; MAXIM, 2014) descrevem em sua obra seis modelos de ciclo de vida de desenvolvimento de software:

a) **Modelo em Cascata:** é o modelo de desenvolvimento mais antigo e tradicional, sendo caracterizado por uma abordagem sequencial, onde cada fase do desenvolvimento de software (requisitos, projeto, implementação, teste, manutenção) é concluída antes de passar para a próxima fase. Este modelo é adequado para projetos bem compreendidos e estáveis, onde os requisitos são claros desde o início;

b) **Modelo em V:** é uma extensão do modelo em cascata, enfatizando a correspondência entre cada fase de desenvolvimento e sua fase de teste. Isso significa que os testes são planejados em paralelo com as atividades de desenvolvimento;

c) **Modelo de Prototipagem:** envolve a criação rápida de um protótipo do software para entender os requisitos do cliente e refiná-los posteriormente. O protótipo é

aprimorado iterativamente até que os requisitos estejam claros e, em seguida, o sistema é desenvolvido usando um modelo mais adequado;

d) **Modelo em Espiral:** enfatiza a gestão de riscos e é iterativo. Ele divide o projeto em ciclos ou espirais, onde cada ciclo envolve quatro principais atividades: identificação de riscos, análise de requisitos, projeto e implementação. Ao final de cada ciclo é realizada uma etapa de avaliação do progresso. Este modelo é adequado para projetos complexos e de longa duração;

e) **Modelo Incremental:** divide o sistema em partes menores ou incrementos, que são desenvolvidos e entregues iterativamente. Cada incremento adiciona funcionalidade ao sistema, sendo um modelo extremamente útil quando é importante entregar partes do sistema em intervalos regulares;

f) **Modelo Ágil:** é uma abordagem iterativa e colaborativa que enfatiza a entrega rápida e a adaptação aos requisitos em constante mudança. Valoriza a interação e a constante comunicação entre a equipe e os clientes do projeto, sendo um modelo ideal para projetos de software com requisitos voláteis.

3.1.2 Engenharia de requisitos

Segundo (KOTONYA; SOMMERVILLE, 1998), a Engenharia de Requisitos é uma disciplina fundamental dentro da Engenharia de Software, concentrando-se na definição, coleta, análise, especificação e no gerenciamento de requisitos para o desenvolvimento de sistemas de software.

Ele discorre em seu livro sobre a importância do processo sistemático e disciplinado na definição e gestão de requisitos de software, definindo sete diretrizes para guiar este processo durante o desenvolvimento:

a) Atuar sobre os requisitos do software como a base para todo e qualquer desenvolvimento;

b) Identificar e compreender claramente as necessidades das partes interessadas;

c) Definir o processo para a coleta, análise e especificação dos requisitos;

d) Documentar de forma completa e precisa os requisitos e rastrear as necessidades ao longo do ciclo de vida do projeto;

e) Implementar estratégias para a gestão de mudanças e a resolução de conflitos de requisitos;

f) Revisar e verificar os requisitos para identificar inconsistências antes de seguir com o desenvolvimento;

g) Favorecer a comunicação eficaz entre os membros do time e as demais partes interessadas.

3.1.2.1 Requisitos funcionais

Requisitos funcionais são uma categoria de requisitos de software que descrevem as funcionalidades ou comportamentos específicos que um sistema de software deve ter para atender às necessidades dos usuários e das partes interessadas. Em outras palavras, eles definem as ações que o software deve ser capaz de executar, as operações que deve realizar e como ele deve responder a entradas específicas. (PRESSMAN; MAXIM, 2014)

Esses requisitos geralmente incluem descrições detalhadas das interações do sistema com os usuários, os processos de negócios que o sistema deve automatizar e as saídas esperadas em resposta a determinadas entradas. Os requisitos funcionais são essenciais para delinear o escopo do software e servem como base para o projeto, implementação e teste do sistema.

3.1.2.2 Requisitos não funcionais

Requisitos não funcionais são uma especificação que descreve as qualidades e restrições que não se referem diretamente às funcionalidades do sistema de software, mas sim às suas características de desempenho, segurança, usabilidade, confiabilidade e outros atributos de qualidade de uma aplicação. (PRESSMAN; MAXIM, 2014)

Os requisitos não funcionais podem ser categorizados nos seguintes grupos:

a) **Desempenho:** definem como o sistema deve se comportar em termos de tempo de resposta, capacidade de processamento e uso de recursos;

b) **Segurança:** estabelecem os mecanismos e controles de segurança necessários para proteger os dados e garantir a integridade do sistema. Isso pode incluir requisitos de autenticação, autorização e criptografia;

c) **Usabilidade:** descrevem as características de interface do usuário que tornam o sistema fácil de usar e compreender. Isso pode abranger requisitos relacionados à acessibilidade, à clareza da interface e à eficiência da interação;

d) **Confiabilidade:** definem a capacidade do sistema de funcionar sem falhas durante um período específico de tempo;

e) **Escalabilidade:** indicam como o sistema deve crescer para atender a um aumento na carga ou demanda. Esse crescimento pode ocorrer de forma horizontal (aumentar o número de servidores que oferecem um serviço) ou vertical (melhorar o desempenho dos servidores);

f) **Manutenibilidade:** descrevem a facilidade com que o sistema pode ser mantido e atualizado. Isso inclui requisitos de documentação adequada, modularidade e capacidade de depuração;

g) **Compatibilidade:** estabelecem os requisitos para garantir que o software seja compatível com diferentes sistemas operacionais, navegadores ou dispositivos, por exemplo;

h) **Regulamentação:** se o software estiver sujeito a regulamentações específicas, os requisitos não funcionais podem incluir conformidade com estas regulamentações, como por exemplo, respeitar a LGPD (*Lei Geral de Proteção de Dados*) e demais regulamentações de privacidade de dados;

i) **Disponibilidade:** define a quantidade de tempo que o sistema deve estar disponível para uso;

j) **Interoperabilidade:** especifica como o sistema deve se comunicar e interagir com outros sistemas ou componentes.

Estes requisitos não funcionais são aspectos críticos de um sistema de software. Por exemplo, um requisito de desempenho pode estipular que um sistema de reservas de passagens aéreas deve ser capaz de processar pelo menos 1000 transações por minuto durante o pico de uso. Um requisito de segurança pode exigir que um aplicativo de banco online adote a autenticação de dois fatores para proteger as informações dos clientes. Requisitos de usabilidade podem especificar que um site de comércio eletrônico deve ser acessível e de fácil navegação para usuários com deficiências visuais. Outros exemplos incluem requisitos de disponibilidade, como garantir que um sistema de telefonia esteja disponível 99,99% do tempo, e requisitos de escalabilidade, como permitir que um sistema de armazenamento em nuvem dimensione horizontalmente para acomodar aumentos na demanda. Portanto, os requisitos não funcionais são fundamentais para garantir que o software atenda às expectativas dos usuários e funcione de maneira confiável e eficiente.

3.1.2.3 Documentação de requisitos

Na seção 3.1.2 sobre Engenharia de Requisitos, descreve-se a importância da documentação dos requisitos de forma completa e precisa ao longo do ciclo de vida do projeto, conforme indica (SOMMERVILLE, 2015).

Segundo (FOWLER, 2003), a UML (*Unified Modeling Language*) ou Linguagem de Modelagem Unificada, é uma linguagem de modelagem visual amplamente utilizada na Engenharia de Software e desempenha um papel valioso na documentação de requisitos, permitindo que os engenheiros de software comuniquem de maneira clara e eficaz os requisitos funcionais e estruturais de um sistema para as partes interessadas. Ela facilita

a compreensão mútua, a análise e o desenvolvimento de sistemas de software complexos, melhorando a qualidade geral do processo de Engenharia de Requisitos.

Contudo, (FOWLER, 2003) argumenta que a UML não deve ser aplicada de forma dogmática e que a eficácia de seu uso depende do contexto e das necessidades do projeto.

3.1.3 Arquitetura de software

A arquitetura de software refere-se à estrutura fundamental e a organização de um sistema de software, abrangendo a divisão de responsabilidades entre os componentes, a interação entre eles e os padrões que guiam a sua composição para o funcionamento do sistema como um todo. Ela visa fornecer uma visão generalista do sistema, delineando seus principais elementos e suas relações, a fim de facilitar o desenvolvimento, a compreensão e a manutenção do software. (GAMMA *et al.*, 1994)

Além disso, essa disciplina envolve a tomada de decisões de design significativas, como por exemplo a escolha de padrões arquiteturais, de modelos de comunicação, a distribuição de tarefas e o gerenciamento de dados na aplicação. Uma arquitetura bem projetada contribui para a escalabilidade, flexibilidade e manutenibilidade do software, além de garantir a satisfação dos requisitos funcionais e não funcionais do sistema, conceitos estes apresentados na seção anterior.

Vale destacar que diferentes sistemas de software podem adotar diversas arquiteturas, cada uma com as suas características distintas. Por exemplo: a arquitetura em camadas organiza o sistema em níveis hierárquicos, enquanto a orientada a serviços favorece a modularidade por meio de serviços independentes. Sistemas monolíticos integram todos os componentes em uma única estrutura, facilitando o desenvolvimento, mas limitando a escalabilidade. Por outro lado, a arquitetura baseada em eventos promove a comunicação assíncrona entre os componentes, favorecendo a escalabilidade e a resiliência do sistema. Além disso, existem outros tipos de arquitetura, como o modelo cliente-servidor, a arquitetura hexagonal e a arquitetura em fases. (GAMMA *et al.*, 1994)

A correta seleção destas arquiteturas está intrinsecamente vinculada aos requisitos específicos do projeto, bem como aos seus objetivos de manutenção, escalabilidade e aos outros requisitos não funcionais a serem atendidos a médio e longo prazo.

3.1.3.1 Arquitetura limpa

A arquitetura limpa é um conceito proposto por Robert C. Martin, também conhecido como Uncle Bob. Essa abordagem de arquitetura de software visa criar sistemas que são modularizados, testáveis, e que sigam os princípios da responsabilidade única e da separação de preocupações. A ideia fundamental por trás da arquitetura limpa é criar um sistema que seja independente de *frameworks* externos, bancos de dados ou detalhes da interface do usuário, tornando-o mais flexível e fácil de manter a longo prazo. (MARTIN, 2017)

Este padrão arquitetural está conceitualmente dividido em 4 camadas principais, que são:

a) **Entidades:** Esta camada contém os conceitos e representações do domínio do negócio. Em geral, esta camada sofre poucas alterações e costuma estar diretamente ligada a conceitos e a abstrações do mundo real;

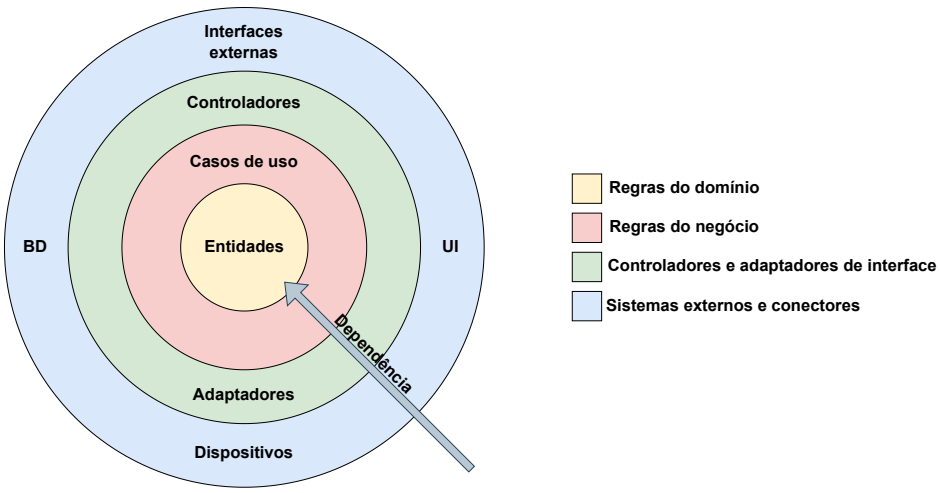
b) **Casos de uso:** Como o nome já indica, esta camada contém a lógica de negócio da aplicação e os casos de uso do sistema;

c) **Controladores:** Esta camada contém todos os controladores e adaptadores entre as camadas mais externas da aplicação e as camadas internas da lógica de negócio do problema;

d) **Interfaces externas:** Esta camada é composta por todos os subsistemas que fazem conexão entre a aplicação e o mundo exterior. Ou seja, as interfaces com o usuário, as conexões com os sistemas externos, entre outros.

Importante destacar que neste padrão arquitetural o sentido das dependências deve ocorrer sempre de uma camada mais externa para uma camada mais interna, ou seja, as interfaces externas dependem dos controladores e dos adaptadores, que dependem dos casos de uso do sistema, que, por sua vez, dependem das regras de domínio definidas na camada das entidades. Essa estrutura pode ser vista em mais detalhes na Figura 1 abaixo.

Figura 1 – Camadas da arquitetura limpa



Fonte: Adaptado pelo autor de (MARTIN, 2017)

3.1.3.2 MVVM: Model-View-Viewmodel

A arquitetura MVVM (*Model View Viewmodel*), é um padrão arquitetural utilizado no desenvolvimento de software para isolar a lógica de apresentação da aplicação da

lógica do sistema e das regras de negócio. Essa separação de responsabilidades facilita a manutenção do código e aumenta muito a flexibilidade em ambientes onde a interface do usuário precisa ser rapidamente adaptada e testada. (CHUGH, 2022)

A estrutura do MVVM é composta por três componentes principais:

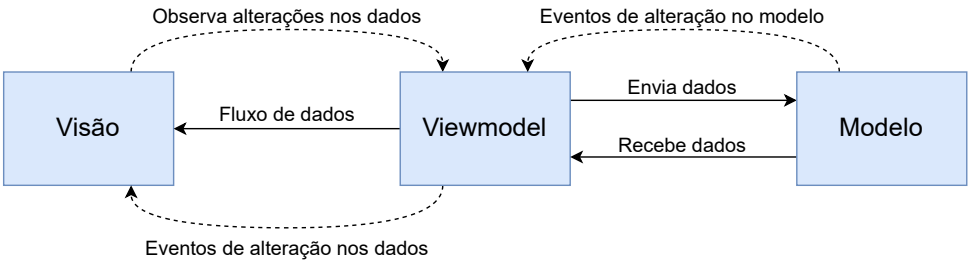
a) ***Model* (Modelo):** Representa a camada de dados e a lógica de negócio da aplicação. O principal ponto de destaque deste componente é que ele não possui conhecimento direto das demais camadas, ou seja, ele atua de forma completamente independente da camada de apresentação do sistema;

b) ***View* (Visão):** Este componente contém a interface do usuário. Ele é o único responsável por expor as informações ao usuário e coletar os comandos necessários para a execução de algum dos caso de uso do sistema;

c) ***Viewmodel*:** Atua como um intermediário entre as duas camadas anteriormente descritas. A sua função é conter a lógica necessária para a apresentação e manipulação dos dados, bem como armazenar o estado da camada de visão, interagindo com o modelo quando necessário.

A Figura 2, apresentada a seguir, detalha o fluxo de dados entre os diferentes componentes da arquitetura MVVM, bem como a origem dos eventos enviados entre as diferentes camadas.

Figura 2 – Estrutura do modelo MVVM



Fonte: Adaptado pelo autor de (CHUGH, 2022)

Nesta representação, pode-se observar que a camada da *viewmodel* atua fortemente em conjunto com a camada do modelo, trocando informações de forma direta para a execução dos casos de uso do sistema. Além disso, esta mesma camada reage às alterações realizadas pelos usuários na camada da visão, comunicando esta última sempre que alguma informação relevante no modelo for alterada.

3.2 SISTEMAS WEB

Um sistema web é uma coleção de componentes interligados que funcionam juntos para fornecer funcionalidades e serviços por meio da internet. Estes componentes incluem

servidores, interfaces de usuário, bancos de dados, linguagens de programação, protocolos de comunicação e outros elementos que permitem a interação entre os usuários e os recursos fornecidos pelo sistema em questão. (FELKE-MORRIS, 2016)

Embora a evolução da web tenha suas raízes em trabalhos e experimentos anteriores, os sistemas web como os conhecemos hoje começaram a surgir na década de 1990. Em 1989, Tim Berners-Lee, um cientista da computação britânico, inventou a World Wide Web como um sistema de informações baseado na internet. Ele desenvolveu o primeiro navegador web e o primeiro servidor web no ano seguinte. Isso marcou o início da web como a conhecemos hoje. (W3C, 2023)

Desde então, os sistemas web continuaram a evoluir, tornando-se cada vez mais complexos e abrangentes. Hoje, a web desempenha um papel fundamental em todos os aspectos da vida moderna, desde a comunicação e o entretenimento até o comércio eletrônico e a colaboração online. Ela continua a evoluir com avanços em tecnologias como os PWAs (*Progressive Web Application*), passando pela computação em nuvem, até chegar aos dispositivos IOT (*Internet of Things*) ou a internet das coisas.

3.2.1 Linguagens e frameworks

De acordo com (SEBESTA, 2018), uma linguagem de programação é um conjunto de regras léxicas, sintáticas e semânticas que permitem que os desenvolvedores escrevam instruções que um computador pode compreender e executar. Ela atua como uma ponte entre a linguagem humana e a linguagem de máquina, facilitando a comunicação entre os desenvolvedores e servindo de especificação formal para softwares e sistemas de computação.

Um framework, por outro lado, é uma estrutura ou conjunto de bibliotecas de código que oferece funcionalidades comuns para facilitar o desenvolvimento de software. Ele fornece um esqueleto ou um conjunto de ferramentas para simplificar tarefas comuns, como a manipulação de bancos de dados, a criação de interfaces de usuário, o gerenciamento de sessões, entre outras. Frameworks são usados para acelerar o desenvolvimento de aplicativos, tornando-os mais eficientes e consistentes.

3.2.1.1 HTML, CSS e Javascript

O HTML (*HyperText Markup Language* ou Linguagem de Marcação de Hipertexto) é uma linguagem de marcação utilizada para criar páginas da web. Ela é a espinha dorsal da maioria dos sites na internet e permite que os desenvolvedores estruturem o conteúdo de uma página web, definindo elementos como textos, imagens, links, formulários e outros elementos interativos de forma hierárquica. (ROBBINS, 2012)

Segundo (ROBBINS, 2012), o CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata), é uma linguagem de estilo utilizada para controlar a apresentação e a formatação de documentos HTML. Enquanto o HTML é responsável pela estrutura e pelo conteúdo

de uma página web, o CSS permite que os desenvolvedores controlem o design, o layout e a aparência visual deste documento.

JavaScript é uma linguagem de programação de alto nível amplamente utilizada para desenvolver sistemas web interativos. Ela pode ser executada tanto no lado do cliente, ou seja, diretamente no navegador web do usuário, quanto no lado do servidor, permitindo que os desenvolvedores adicionem funcionalidades dinâmicas e interatividade às suas aplicações. (HAVERBEKE, 2018)

3.2.1.2 Kotlin e Ktor

De acordo com (JEMEROV; ISAKOVA, 2017), o Kotlin é uma linguagem de programação moderna e versátil criada pela empresa tcheca JetBrains. Esta, conhecida por suas ferramentas de desenvolvimento de software, lançou a linguagem no ano de 2011. Desde então o Kotlin tem ganhado popularidade não somente para o desenvolvimento de aplicativos Android, mas também para o desenvolvimento de sistemas web, haja vista a sua interoperabilidade com a plataforma Java e, mais recentemente, o surgimento do framework *Kotlin Multiplatform*.

O Kotlin se destaca por ser uma linguagem de programação extremamente flexível, oferecendo vantagens como a interoperabilidade com a plataforma Java, conforme dito anteriormente, bem como uma série de recursos que aumentam a produtividade e a segurança do desenvolvedor. Entre eles estão, por exemplo: a possibilidade de utilizar programação orientada à objetos e programação funcional no mesmo ambiente; possuir tipagem estática, com diferenciação clara entre tipos que podem conter o valor nulo ou não; possuir funções de extensão e permitir a sobrecarga de operadores, o que torna a linguagem extremamente concisa e expressiva; entre outros.

O Ktor é um framework de desenvolvimento de sistemas web implementado em Kotlin. Ele foi criado inicialmente também pela empresa JetBrains e hoje está disponível como um projeto de código aberto com contribuições constantes da comunidade. As suas principais vantagens são a ampla gama de módulos e recursos prontos para uso, incluindo roteamento, manipulação de solicitações e respostas, suporte a autenticação, WebSockets, entre outros. Além disso o Ktor foi construído desde o início com suporte para programação assíncrona, o que o torna adequado para lidar com um grande número de conexões simultâneas e, por consequência, ideal para sistemas web em tempo real e de alto desempenho. Por fim, ele é altamente extensível e permite que os desenvolvedores integrem facilmente bibliotecas de terceiros, criem seus próprios módulos e personalizem o comportamento do framework de acordo com as necessidades do projeto. (SIVAN, 2023)

3.2.2 Protocolos e padrões arquiteturais

Os sistemas web possuem costumeiramente uma arquitetura no padrão backend-frontend, o que permite a separação das responsabilidades entre as camadas de apresen-

tação (frontend) e a camada da lógica do servidor (backend). Essa divisão permite que as equipes de desenvolvimento se especializem em áreas específicas, facilitando o desenvolvimento, a manutenção e a escalabilidade de sistemas web. Além disso, a comunicação entre o frontend e o backend ocorre por meio de solicitações HTTP (*Hypertext Transfer Protocol* ou Protocolo de Transferência de Hipertexto), geralmente usando APIs (*Application Programming Interface* ou Interface de Programação de Aplicativos) para transmitir dados entre as camadas. (FELKE-MORRIS, 2016)

3.2.2.1 HTTP e HTTPs

O HTTP é um protocolo de comunicação utilizado para transferir dados pela internet. Ele é fundamental para a troca de informações entre um cliente, geralmente um navegador web, e um servidor de um sistema web. Além disso, o protocolo foi desenvolvido para ser simples, capaz de ser facilmente compreendido pelos desenvolvedores; extensível, através da gestão de cabeçalhos pré-definidos entre cliente e servidor e; sem estado, o que significa que cada solicitação é tratada de forma independente, sem que o servidor mantenha informações sobre as solicitações anteriores. (MOZILLA, 2023)

Contudo, o HTTP é um protocolo de texto simples e não criptografado, o que significa que os dados transferidos podem ser interceptados por terceiros. Para solucionar esta questão, foi criado o HTTPS (*Hypertext Transfer Protocol Secure* ou Protocolo de Transferência de Hipertexto Seguro), que utiliza as tecnologias SSL/TLS para proteger a comunicação entre o cliente e o servidor.

3.2.2.2 JSON

O JSON (*JavaScript Object Notation*) é um formato de dados utilizado para representar e trocar informações estruturadas entre sistemas. Ele é amplamente utilizado na programação e na comunicação entre clientes e servidores, especialmente em sistemas web, devido à sua simplicidade e flexibilidade. (BASSETT, 2015)

O formato teve a sua origem na década de 1990, quando foi formalizado e popularizado por Douglas Crockford, um desenvolvedor de software e autor de livros sobre JavaScript. Em 2001, Crockford publicou um memorando chamado "The Fat-Free Alternative to XML"(A alternativa sem gordura ao XML), onde introduziu o JSON como uma alternativa mais leve ao XML para troca de dados.

No ano de 2013, o JSON foi padronizado pela RFC7159, que descreve a sua sintaxe e as regras de codificação utilizadas amplamente no desenvolvimento de sistemas com o uso desse formato.

3.2.2.3 Serviços web e APIs

Um serviço web é um sistema ou aplicação de software que permite a comunicação e a interação entre diferentes programas ou sistemas de software pela internet ou por uma rede local. Eles permitem que aplicativos diversos, que podem estar em plataformas diferentes e usar linguagens de programação diferentes, troquem dados e realizem ações, possibilitando que funcionem juntos de forma eficiente.

Para atingir esta interoperabilidade característica dos serviços web, eles usam protocolos e formatos de dados universalmente conhecidos na indústria, como por exemplo o HTTP, descrito na seção 3.2.2.1, e o JSON, descrito na seção 3.2.2.2.

Além disso, os serviços web são projetados com o desacoplamento em mente, podendo ser utilizados por sistemas terceiros sem que estes precisem ter o conhecimento detalhado das operações internas do serviço com o qual estão interagindo. Esta característica se faz viável através da publicação da API do serviço web em questão e promove a flexibilidade e a modularidade na arquitetura do software desenvolvido.

Esta API nada mais é do que um conjunto de regras, ou um contrato, que permite a comunicação entre as diferentes camadas de um sistema de software. Este contrato define os métodos, os formatos de dados e a forma de interação que os desenvolvedores podem usar para interagir com um determinado serviço, biblioteca, ou mesmo com o sistema operacional. (JACOBSON; BRAIL; WOODS, 2012)

Ainda segundo (JACOBSON; BRAIL; WOODS, 2012), vale ressaltar uma última característica dos serviços web que é a sua escalabilidade. Ou seja, estes serviços podem ser facilmente expandidos, adicionando-se mais recursos de servidor, ou mesmo mais servidores, tornando-os adequados para lidar com um grande número de solicitações simultâneas.

3.2.2.4 REST

Uma API RESTful (*Representational State Transfer* ou Transferência de Estado Representacional) é uma API que respeita um conjunto de princípios arquiteturais e regras para projetar e interagir com sistemas web. Ele é um estilo de arquitetura de software que usa protocolos e métodos HTTP para criar, ler, atualizar e excluir recursos, geralmente representados em formato JSON, de forma simples e eficaz. O termo representacional, em REST, refere-se à ideia de que cada recurso pode ser representado de maneira clara e significativa por meio de uma URL (*Uniform Resource Locator* ou Localizador Uniforme de Recursos). (RICHARDSON; AMUNDSEN; RUBY, 2013)

Também de acordo com (RICHARDSON; AMUNDSEN; RUBY, 2013), uma API RESTful utiliza como resposta em seus recursos, também chamados de *endpoints*, o conjunto de códigos de status HTTP, como *200 OK*, *201 Created*, *400 Bad Request*, *404 Not Found*, entre outros. Estes códigos padronizam o resultado esperado de uma consulta à um serviço web e facilitam a interpretação destes resultados pelo cliente do serviço.

Dessa forma, as APIs RESTful são amplamente utilizadas na construção de serviços web, pois são simples de entender, escaláveis e altamente interoperáveis.

3.2.3 Armazenamento de dados

O armazenamento de dados em sistemas web é uma parte fundamental do desenvolvimento e da infraestrutura de aplicativos online. Ele envolve o gerenciamento e a persistência de informações que são usadas pelos aplicativos da web para fornecer funcionalidades aos usuários. Existem várias tecnologias para se armazenar estas informações nos sistemas web, e a sua escolha depende diretamente dos requisitos funcionais e não funcionais, apresentados nas seções 3.1.2.1 e 3.1.2.2, respectivamente, destes sistemas web. (KLEPPMANN, 2017)

Também de acordo com (KLEPPMANN, 2017), existem quatro principais formatos e tecnologias para o armazenamento de dados em sistemas web, que são:

a) **Banco de dados relacional:** é uma das formas mais populares para o armazenamento de dados em sistemas web, operando com a persistência de dados estruturados em tabelas e com esquemas previamente definidos. São ideais para serviços que exigem integridade referencial e consistência dos dados, como por exemplo, sistemas de gerenciamento de conteúdo, sistemas de gerenciamento de pedidos, sistemas de transações bancárias, entre outros. Alguns exemplos de produtos de armazenamento nesta categoria são: o MySQL, o PosgreSQL, o Microsoft SQLServer e o Oracle Database;

b) **Banco de dados não relacional:** também chamados de bancos de dados NoSQL, são utilizados para armazenar dados não estruturados ou semi-estruturados. Ou seja, este formato de persistência não exige um esquema previamente definido para armazenar as informações da aplicação. Por conta dessa característica, eles são mais escaláveis e flexíveis do que os bancos de dados relacionais, tornando-se ideais para aplicativos que precisam lidar com grandes volumes de dados, como redes sociais e aplicativos em tempo real, por exemplo. Alguns exemplos de produtos de persistência nesta categoria são: o MongoDB, o Cassandra, o Neo4j e o Scylla;

c) **Banco de dados em memória:** esta tecnologia é costumeiramente utilizada para a persistência de dados de forma temporária, sendo altamente eficaz para acelerar o acesso aos dados frequentemente utilizados, melhorando o desempenho do serviço e reduzindo a carga nos bancos de dados principais. Alguns exemplos deste tipo de tecnologia são: o Memcached e o Redis;

d) **Armazenamento de objetos:** o armazenamento de objetos é uma necessidade comum dos sistemas web, afinal, muitos destes sistemas precisam armazenar arquivos como documentos, áudios, imagens e vídeos. Esta persistência é realizada diretamente

no sistema de arquivos de um servidor ou através de um serviço de armazenamento de objetos como o Amazon S3, o Google Cloud Storage ou o Azure Blob Storage, por exemplo.

Assim, um sistema de armazenamento desempenha um papel crítico em uma aplicação web, sendo responsável por manter, organizar e disponibilizar os dados necessários para o funcionamento do serviço, garantindo a persistência dos dados do usuário, de conteúdo dinâmico, das configurações e informações do sistema e permitindo a sua recuperação rápida e confiável quando necessário.

3.3 SISTEMAS PARA DISPOSITIVOS MÓVEIS

Em sua origem, os computadores eram máquinas suficientemente grandes ao ponto de ocupar mais de uma sala em instituições de ensino, grandes organizações e órgãos do governo. (ALECRIM, 2013) Com o avanço das tecnologias, estes computadores têm se tornado cada vez mais compactos, eficientes, convenientes e acessíveis, possibilitando a sua portabilidade e utilização por qualquer pessoa em praticamente qualquer local. Essas tecnologias deram origem aos pequenos dispositivos que hoje chamamos de dispositivos móveis.

3.3.1 Dispositivos móveis

Um dispositivo móvel é um dispositivo eletrônico portátil projetado para facilitar a comunicação, o acesso a informações, a produtividade e o entretenimento enquanto o usuário está em movimento. Esses dispositivos são caracterizados por sua portabilidade e são geralmente pequenos o suficiente para serem carregados facilmente em uma bolsa ou no bolso. (LEE; SCHNEIDER; SCHELL, 2005)

Alguns exemplos comuns de dispositivos móveis incluem: os smartphones, que são os dispositivos mais abrangentes, oferecendo recursos de comunicação, acesso à internet, câmeras e entretenimento; os tablets, que assim como os smartphones oferecem uma ampla gama de funcionalidades em uma tela maior, tornando-os dispositivos ótimos para leitura e navegação na internet; os dispositivos vestíveis (ou *wearables*), que são as pulseiras fitness e relógios digitais com monitores ativos de saúde e, em alguns casos, comunicação; entre vários outros.

Segundo (CAMARGO, 2023), os dispositivos móveis já fazem parte da vida de milhares de brasileiros e estima-se que mais da metade dos usuários de eletrônicos acessam a internet por um destes dispositivos ao invés do computador tradicional. Ainda, citando uma pesquisa realizada pelo IBGE (Instituto Brasileiro de Geografia e Estatística) no ano de 2021, existem em média dois celulares no país para cada brasileiro.

À medida em que o uso de dispositivos móveis continua a crescer, é fundamental destacar que essa tendência impulsiona uma forte demanda por soluções de software

compatíveis com essa tecnologia. Estas soluções, comumente chamadas de aplicativos móveis, serão apresentadas na seção 3.3.2 à seguir.

3.3.2 Aplicativos móveis

Um aplicativo móvel é um software desenvolvido especificamente para ser executado em dispositivos móveis, como smartphones e tablets. Esses aplicativos são projetados para realizar uma variedade de tarefas e funções que podem incluir comunicação, entretenimento, produtividade, educação, saúde, jogos e muito mais.

Estes aplicativos são instalados nos dispositivos dos usuários a partir das lojas de aplicativos, como a App Store da Apple ou a Play Store do Google, e podem ser usados para acessar serviços online, interagir com dispositivos externos e executar funções específicas no dispositivo do usuário. Eles são uma parte fundamental da experiência de uso dos dispositivos móveis e desempenham um papel significativo na forma como as pessoas interagem com a tecnologia no seu dia a dia. (RODRIGUES, 2023)

Ainda de acordo com (RODRIGUES, 2023), apontando os resultados do relatório *State of Mobile 2023* e os dados de uma pesquisa realizada pela empresa *Statista*, a receita total no mercado de aplicativos está projetada para atingir US$ 614,40 bilhões até o ano de 2026, com uma taxa de crescimento anual de 7,77%. Além disso, a mesma pesquisa mostra que o smartphone continua sendo o dispositivo móvel preferido dos usuários, com um tempo diário gasto por pessoa de mais de 5 horas (um aumento de 3% em relação ao período anterior).

Nas seções 3.3.3 e 3.3.4 à seguir, estão dispostas informações sobre os dois sistemas operacionais mais utilizados no mundo dos dispositivos móveis, o Android e o iOS. Juntos, estes sistemas correspondem a praticamente 99% do mercado de smartphones e tablets no ano de 2023. (STATISTA, 2023)

3.3.3 Android

O sistema operacional Android é uma plataforma móvel desenvolvida pelo Google, que se tornou uma das mais populares e amplamente utilizadas em dispositivos móveis em todo o mundo. (GOOGLE, 2023)

Algumas das principais características do sistema operacional Android são:

a) **Código aberto:** uma das características mais distintivas do Android é que ele é baseado em código aberto, o que significa que seu código-fonte está disponível para desenvolvedores e fabricantes de dispositivos. Isso permite uma grande flexibilidade e personalização, bem como uma ampla variedade de dispositivos Android no mercado;

b) **Baseado no Linux:** o núcleo de operação do sistema operacional ainda é fortemente baseado no Linux. Este núcleo sofreu significativas alterações ao longo dos anos,

focando nos principais aspectos dos dispositivos móveis mas, as bases principais ainda persistem até hoje;

c) **Interface customizável:** o Android apresenta uma interface de usuário altamente personalizável, com áreas de trabalho onde os usuários podem adicionar widgets, atalhos de aplicativos e personalizar sua aparência de acordo com suas preferências;

d) **Integração com o Google:** o Android é altamente integrado com os ecossistema do Google, contando com aplicativos e serviços como o Google Search, Google Maps, Gmail e Google Drive. Isso oferece uma experiência de usuário coesa e integrada para quem utiliza esses serviços;

e) **Desenvolvimento em Kotlin:** o desenvolvimento de aplicativos para a plataforma Android é feito principalmente usando a linguagem de programação Kotlin, embora também seja possível usar o Java e outras linguagens. Além disso, o Android Studio é a principal ferramenta para a criação de aplicativos para a plataforma.

Dessa forma, o Android é um sistema operacional móvel flexível, de código aberto e altamente personalizável, que oferece uma ampla gama de funcionalidades e aplicativos. Ele desempenha um papel fundamental na popularização de smartphones e tablets em todo o mundo e continua a ser uma plataforma líder no mercado de dispositivos móveis.

3.3.4 iOS

Assim como o Android, apresentado na seção 3.3.3 anterior, o iOS é uma plataforma móvel desenvolvida pela Apple, sendo exclusivamente utilizada em seus dispositivos móveis como o iPhone, iPad e iPod Touch. (APPLE, 2023)

As características de destaque deste sistema operacional são:

a) **Ecossistema integrado:** o iOS é projetado para integrar-se perfeitamente com os outros produtos e serviços da Apple, como o iCloud, o Apple Music, o Apple Pay e o Apple Watch. Isso cria um ecossistema coeso que permite aos usuários sincronizar informações de forma rápida e fácil entre os dispositivos;

b) **Interface intuitiva:** o iOS é conhecido por sua interface de usuário limpa, simples e intuitiva. Ele utiliza ícones na tela inicial e uma barra de navegação inferior para acesso rápido aos aplicativos e funções essenciais, enfatizando os recursos de acessibilidade e a praticidade no uso do sistema operacional pelos usuários;

c) **Segurança:** a Apple coloca uma forte ênfase na segurança e na proteção dos dados dos usuários. O iOS inclui recursos de segurança, como a autenticação biométrica (Touch ID e Face ID), a criptografia de dados em repouso e em trânsito, além de permissões de aplicativos que os usuários podem controlar.

d) **Desenvolvimento em Swift:** para desenvolver aplicativos para iOS, os desenvolvedores geralmente usam a linguagem de programação Swift, que é desenvolvida pela própria Apple e oferece um ambiente de desenvolvimento chamado Xcode. A Apple fornece extensas ferramentas de desenvolvimento e documentação para criar aplicativos iOS de alta qualidade.

Assim, o iOS é conhecido por seu design elegante, segurança robusta e integração profunda com o ecossistema da Apple. Esses elementos combinados tornam o iOS uma escolha popular entre os usuários que buscam uma experiência móvel coesa e confiável.

4 DESENVOLVIMENTO

Neste capítulo é feita uma descrição abrangente da metodologia e das técnicas utilizadas para o desenvolvimento do Sistema de Gestão de Compras para a Empresa Xibuco. A seção 4.1 irá apresentar a análise dos requisitos do sistema, com o detalhamento dos requisitos funcionais e não funcionais identificados no processo; a seção 4.2 apresentará a arquitetura geral do sistema e como ele irá operar; a seção 4.3 contém todas as informações relativas ao desenvolvimento dos serviços de *backend* e de persistência dos dados; a seção 4.4, apresentará o processo de desenvolvimento do aplicativo móvel que será utilizado pelos clientes da empresa Xibuco; e, por fim, a seção 4.5 mostrará o processo de desenvolvimento da interface de administração e operação do sistema por parte dos colaboradores da empresa.

4.1 ANÁLISE DE REQUISITOS

Esta seção tem como finalidade apresentar, de forma detalhada, os resultados do processo de análise de requisitos realizado para o desenvolvimento do Sistema de Gestão de Compras para a Empresa Xibuco. Este processo foi realizado seguindo o modelo descrito na fundamentação teórica deste trabalho, na seção 3.1.2.

Para que a documentação das próximas seções seja clara e objetiva, se faz necessária a identificação dos principais usuários que vão interagir diretamente com o sistema. A estes usuários se dá o nome de atores que, no caso do sistema em questão, são:

a) **Cliente:** é o principal usuário do sistema. Ele é o consumidor final da empresa Xibuco e o seu primeiro objetivo dentro do sistema é vender as suas peças de roupa, calçados e acessórios para a Xibuco;

b) **Funcionário:** é um colaborador da empresa Xibuco. Ele é um ator crucial para a operação do sistema pois atua diretamente em resposta às ações do cliente, avaliando as peças de roupa, calçados e acessórios que os clientes desejam vender para a empresa;

c) **Administrador:** é um funcionário da empresa Xibuco cujo papel de gestor lhe dá direitos administrativos sobre a operação do sistema. Este ator supervisiona as ações dos funcionários e possui permissões de interação com o sistema que os funcionários não possuem.

4.1.1 Requisitos funcionais

A seção de requisitos funcionais descreve as funcionalidades e comportamentos específicos do Sistema de Gestão de Compras para a Empresa Xibuco. Estes requisitos estão apresentados abaixo, separados por ator, para facilitar a compreensão do trabalho.

4.1.1.1 Requisitos funcionais do cliente

a) **RF.C.001:** O cliente pode criar uma nova conta no sistema. Deve ser disponibilizado um formulário de entrada de dados com os campos nome, email e senha. As informações fornecidas pelo cliente devem ser verificadas e, se tudo estiver correto, uma nova conta deve ser criada para que o cliente acesse as funcionalidades internas do sistema;

b) **RF.C.002:** O cliente pode acessar a sua conta. Deve ser disponibilizado um formulário de entrada de dados com os campos email e senha. O sistema deve verificar as informações fornecidas pelo cliente e, se tudo estiver correto, o cliente deve ser redirecionado para a parte interna do sistema;

c) **RF.C.003:** O cliente pode listar os seus inventários previamente cadastrados. A listagem deve apresentar os dados básicos de cada inventário, como o nome, a sua situação e a quantidade de produtos no inventário. Enquanto o inventário ainda não foi submetido para avaliação, a listagem deve disponibilizar opções para editar e para excluir um inventário;

d) **RF.C.004:** O cliente pode cadastrar novos inventários no sistema. Deve ser disponibilizado um formulário de entrada de dados com um campo de nome previamente preenchido pelo sistema de forma automática com a palavra Inventário, seguida de um número sequencial cuja contagem inicia em um. O cliente deve poder alterar o nome do inventário;

e) **RF.C.005:** O cliente pode editar os inventários ainda não enviados para avaliação. Deve ser disponibilizada na listagem de inventários cadastrados uma opção para que o cliente acesse um inventário previamente cadastrado e altere as suas informações internas;

f) **RF.C.006:** O cliente pode excluir os inventários ainda não enviados para avaliação. Deve ser disponibilizada na listagem de inventários cadastrados uma opção para que o cliente exclua um inventário previamente cadastrado;

g) **RF.C.007:** O cliente pode listar os produtos previamente cadastrados em um inventário. A listagem deve apresentar os dados básicos de cada produto, como o tipo do produto, o nome e o tamanho. Enquanto o inventário ainda não foi submetido para avaliação, a listagem deve disponibilizar opções para editar e para excluir um produto de um inventário;

h) **RF.C.008:** O cliente pode inserir um novo produto em um inventário ainda não enviado para avaliação. Deve ser disponibilizado um formulário de entrada de dados com os campos tipo do produto, nome, tamanho, marca e condição. O campo tipo

do produto deve conter um dos valores entre roupa, calçado ou acessório. O campo de nome deve ser previamente preenchido pelo sistema de forma automática com o tipo do produto, seguido de um número sequencial cuja contagem inicia em um. O cliente pode alterar o nome do produto. Os campos tamanho e marca são campos abertos que devem ser preenchidos pelo cliente. O campo condição deve conter um dos valores entre novo, aparentemente novo, gentilmente usado ou pequeno detalhe. Todos os campos devem ser obrigatoriamente preenchidos pelo cliente para que o produto seja adicionado ao inventário;

i) **RF.C.009:** O cliente pode editar os produtos inseridos em um inventário ainda não enviado para avaliação. Deve ser disponibilizado um formulário de entrada de dados semelhante ao especificado no requisito funcional do cliente de número 8 (RF.C.008);

j) **RF.C.010:** O cliente pode excluir os produtos inseridos em um inventário ainda não enviado para avaliação. Deve ser disponibilizada na listagem de produtos inseridos no inventário uma opção para que o cliente exclua um produto previamente cadastrado no inventário;

k) **RF.C.011:** O cliente pode adicionar imagens do produto ao inserir um novo produto (RF.C.008) ou ao editar um produto previamente cadastrado (RF.C.009) em um inventário ainda não enviado para avaliação;

l) **RF.C.012:** O cliente pode remover imagens previamente adicionadas ao produto, seguindo as especificações do requisito funcional do cliente de número 11 (RF.C.011), em um inventário ainda não enviado para avaliação;

m) **RF.C.013:** O cliente pode visualizar um resumo do seu inventário antes de enviá-lo para avaliação. Este resumo deve apresentar os dados básicos do inventário, como o nome, a sua situação, a quantidade de produtos no inventário e uma estimativa de valor monetário que poderá ser pago ao cliente pela empresa Xibuco caso a venda se concretize;

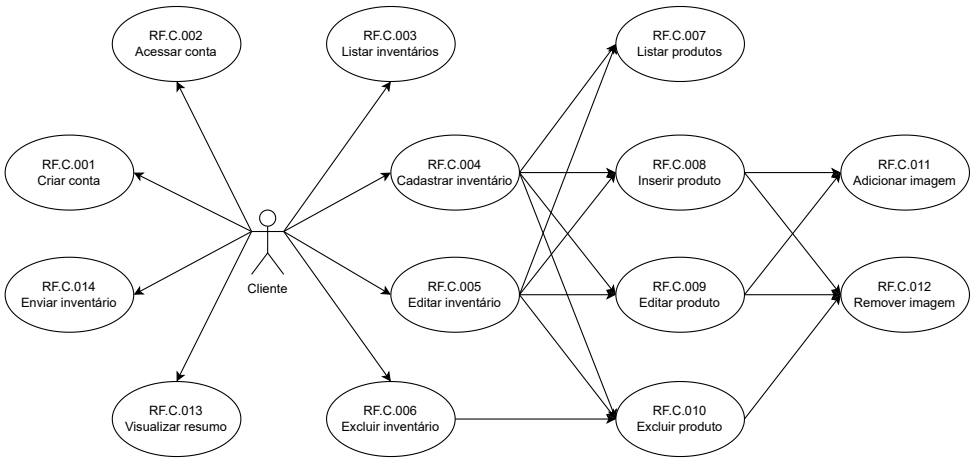
n) **RF.C.014:** O cliente pode enviar o inventário para avaliação, alterando a situação do inventário para avaliação e restringindo posteriores edições e ou deleções.

A Figura 3, a seguir, apresenta o diagrama de casos de uso com os requisitos do cliente. Este diagrama mostra de forma simples e direta a interação do ator com os diferentes pontos de entrada do sistema.

4.1.1.2 Requisitos funcionais do funcionário

a) **RF.F.001:** O funcionário pode acessar a sua conta. Deve ser disponibilizado um formulário de entrada de dados com os campos email e senha. O sistema deve verificar

Figura 3 – Diagrama de casos de uso do cliente



Fonte: Elaborado pelo autor

as informações fornecidas pelo funcionário e, se tudo estiver correto, o funcionário deve ser redirecionado para a parte interna do sistema;

b) **RF.F.002:** O funcionário pode listar os inventários previamente cadastrados pelos clientes. A listagem deve apresentar os dados básicos de cada inventário, como o nome, a sua situação e a quantidade de produtos no inventário. Enquanto o inventário ainda não foi avaliado, a listagem deve disponibilizar uma opção para avaliar o inventário;

c) **RF.F.003:** O funcionário pode avaliar os produtos de um inventário previamente cadastrado por um cliente. Deve ser disponibilizado um formulário de entrada de dados que apresenta as informações de todos os produtos adicionados ao inventário pelo cliente. O funcionário deve poder editar os campos tipo do produto, nome, tamanho, marca e condição; visualizar o preço sugerido para o produto pelo sistema e informar o preço que realmente a empresa Xibuco está disposta a pagar pelo produto caso a venda se concretize, em conjunto com uma observação opcional para cada produto;

d) **RF.F.004:** O funcionário pode visualizar as imagens adicionadas aos produtos dos inventários cadastrados pelos clientes;

e) **RF.F.005:** O funcionário pode marcar um inventário previamente cadastrado por um cliente como avaliado se todos os produtos deste inventário já estiverem avaliados de acordo com o requisito funcional do funcionário de número 3 (RF.F.003), alterando a situação do inventário para avaliado;

f) **RF.F.006:** O funcionário pode listar os inventários previamente cadastrados pelos clientes que já estejam avaliados. A listagem deve apresentar os dados básicos de cada

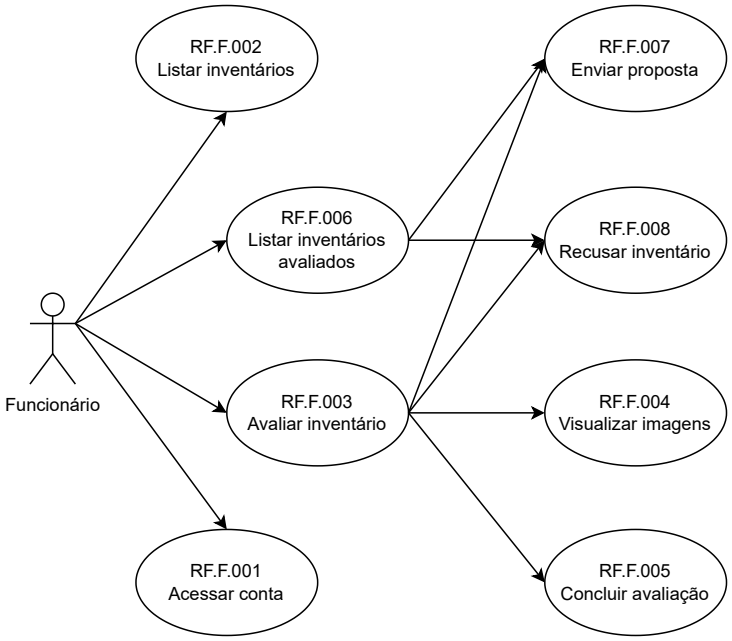
inventário, como o nome, a sua situação, a quantidade de produtos no inventário e o valor total do inventário após a avaliação. Nesta condição, devem ser disponibilizadas opções para enviar uma proposta de compra do inventário ao cliente e para recusar a compra do inventário;

g) **RF.F.007:** O funcionário pode enviar uma proposta de compra de um inventário para um cliente;

h) **RF.F.008:** O funcionário pode enviar uma comunicação ao cliente de que a compra do seu inventário foi recusada.

A Figura 4, a seguir, apresenta o diagrama de casos de uso com os requisitos do funcionário. Este diagrama mostra de forma simples e direta a interação do ator com os diferentes pontos de entrada do sistema.

Figura 4 – Diagrama de casos de uso do funcionário



Fonte: Elaborado pelo autor

4.1.1.3 Requisitos funcionais do administrador

Conforme apresentado na descrição dos usuários do sistema na seção 4.1, um administrador é também um funcionário da empresa Xibuco. Portanto, os requisitos funcionais do funcionário descritos na seção 4.1.1.2 também se aplicam aos administradores do sistema.

Além deles, os seguintes requisitos se aplicam exclusivamente aos administradores do sistema:

a) **RF.A.001:** O administrador pode criar uma nova conta no sistema. Deve ser disponibilizado um formulário de entrada de dados com os campos nome, email, senha e tipo de usuário. As informações fornecidas pelo administrador devem ser verificadas e, se tudo estiver correto, uma nova conta deve ser criada para que o usuário acesse as funcionalidades internas do sistema;

b) **RF.A.002:** O administrador pode listar os usuários previamente cadastrados no sistema. A listagem deve apresentar os dados básicos de cada usuário, como o nome, o e-mail, o tipo de usuário e a sua situação;

c) **RF.A.003:** O administrador pode alterar o tipo de usuário de uma conta;

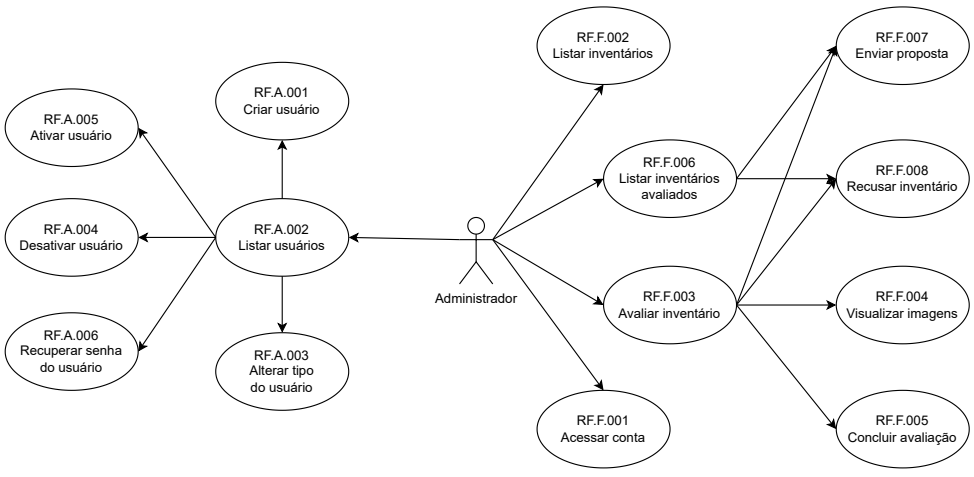
d) **RF.A.004:** O administrador pode desativar uma conta de usuário, impedindo que o usuário acesse as funcionalidades internas do sistema;

e) **RF.A.005:** O administrador pode ativar uma conta de usuário, permitindo que o usuário acesse as funcionalidades internas do sistema;

f) **RF.A.006:** O administrador pode recuperar a senha de uma conta de usuário.

A Figura 5, a seguir, apresenta o diagrama de casos de uso com os requisitos do administrador. Este diagrama mostra de forma simples e direta a interação do ator com os diferentes pontos de entrada do sistema.

Figura 5 – Diagrama de casos de uso do administrador



Fonte: Elaborado pelo autor

4.1.2 Requisitos não funcionais

Conforme explicitado na seção 3.1.2.2, sobre requisitos não funcionais, da fundamentação teórica, os requisitos não funcionais são uma parte crucial para o bom funcionamento

de um sistema. Assim, ficam definidos alguns requisitos não funcionais baseados em cada uma das categorias apresentadas naquela seção:

a) **RNF.001 - Desempenho:** O sistema deve ser capaz de executar as ações previstas na seção 4.1.1, sobre requisitos funcionais, sem que seja percebida demora por parte do usuário. Dessa forma, todas as interações do usuário devem ter resposta do sistema em menos de 1 segundo;

b) **RNF.002 - Segurança:** Todos os dados trafegados e persistidos pelo sistema devem estar obrigatoriamente criptografados, garantindo que os dados pessoais dos usuários estejam protegidos contra acesso não autorizado;

c) **RNF.003 - Usabilidade:** O sistema deve seguir as diretrizes de acessibilidade da plataforma a que se destina. Ou seja, deve respeitar as orientações do WCAG (*Web Content Accessibility Guidelines*) no caso do sistema web e as respectivas diretrizes de acessibilidade no caso das plataformas móveis Android e iOS;

d) **RNF.004 - Confiabilidade:** O sistema deve ser capaz de executar as suas funções sem falhas em todas as suas interações críticas. Assim, falhas previstas devem ser tratadas e informadas aos usuários sem degradação significativa do desempenho ou interrupções;

e) **RNF.005 - Escalabilidade:** O sistema deve ser capaz de suportar um aumento na demanda de até 10 vezes a sua média de usuários mensal;

f) **RNF.006 - Manutenibilidade:** O código-fonte do sistema deve ser bem documentado e seguir boas práticas de programação, facilitando a manutenção e atualização futuras;

g) **RNF.007 - Compatibilidade:** O sistema deve ser totalmente compatível com dispositivos, sistemas operacionais e navegadores lançados nos últimos 3 anos;

h) **RNF.008 - Regulamentação:** Como o sistema armazena e trafega informações e dados pessoais dos seus usuários, este deverá respeitar integralmente a LGPD;

i) **RNF.009 - Disponibilidade:** O sistema deve estar disponível para uso durante 24 horas por dia, 7 dias por semana, com um tempo de inatividade planejado não superior a 1 hora por mês para manutenção programada;

j) **RNF.010 - Interoperabilidade:** O sistema deve ser capaz de se integrar facilmente com sistemas de terceiros por meio de APIs, permitindo que os usuários com as devidas permissões acessem e compartilhem dados de maneira eficiente com outras aplicações e serviços.

4.2 ARQUITETURA GERAL DO SISTEMA

4.2.1 Componentes do sistema

Considerando-se o levantamento de requisitos realizado no escopo deste trabalho e apresentado na seção 4.1, podemos identificar 3 grandes componentes no sistema de gestão de compras da Xibuco. Estes componentes atuam de forma completamente independente no ponto de vista dos requisitos do sistema, contudo, para a otimização do processo de desenvolvimento e para aumentar a manutenibilidade a médio e longo prazo, é de suma importância ressaltar que eles compartilham o domínio, a lógica e, em virtude da tecnologia adotada neste trabalho, boa parte da implementação e do seu código fonte. Assim, os 3 componentes identificados são:

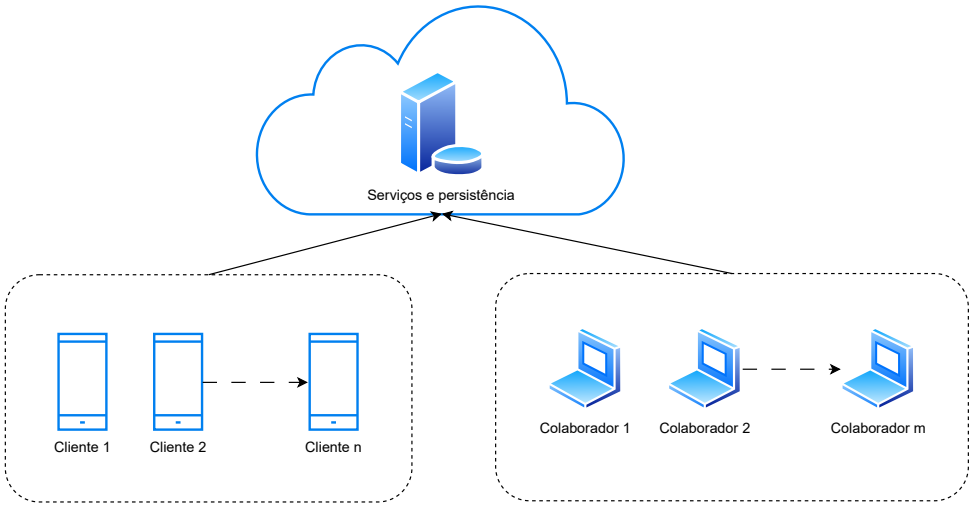
a) **Serviços e persistência:** Acessível através de um servidor centralizado, este componente do sistema atua de forma oculta para o cliente final da empresa. Dessa maneira, operando na retaguarda do funcionamento do sistema, ele é responsável por armazenar, tratar e abastecer os demais componentes com informações corretas e confiáveis, mantendo a integridade e a aderência às regras de negócio especificadas na análise de requisitos da seção 4.1 e arbitrando em casos de divergência entre quaisquer outros componentes do sistema. Ou seja, este é o ponto central de toda a operação, interligando os demais componentes de forma rápida e eficiente;

b) **Aplicativo móvel:** Esta é a porta de entrada do usuário mais importante deste sistema, o cliente final da Xibuco. Este aplicativo móvel é responsável por apresentar para este cliente as suas opções de forma fácil, clara e rápida, estimulando uma boa interação e garantindo que os processos da empresa sejam realizados de forma correta e segura. Todas as suas informações relevantes são fornecidas pela camada de serviços e persistência, sendo posteriormente organizadas e apresentadas ao usuário;

c) **Ambiente de administração:** Este ambiente será utilizado pelos colaboradores da Xibuco. Nele serão realizadas as interações necessárias para a operação do sistema como um todo mas, principalmente, as interações necessárias para a conclusão dos processos de aquisição de novos inventários, que são o foco deste projeto. Além da operacionalização deste processo de aquisição de inventários, este componente serve também de interface para a administração do sistema, permitindo edições avançadas nos inventários, além da gestão centralizada das contas dos usuários em todos os demais componentes.

Cada um destes componentes será descrito e apresentado em maiores detalhes nas próximas seções 4.3, 4.4 e 4.5 deste trabalho, respectivamente, podendo ser visualizados em um contexto geral do sistema através da Figura 6 a seguir.

Figura 6 – Arquitetura geral dos componentes do sistema



Fonte: Elaborado pelo autor

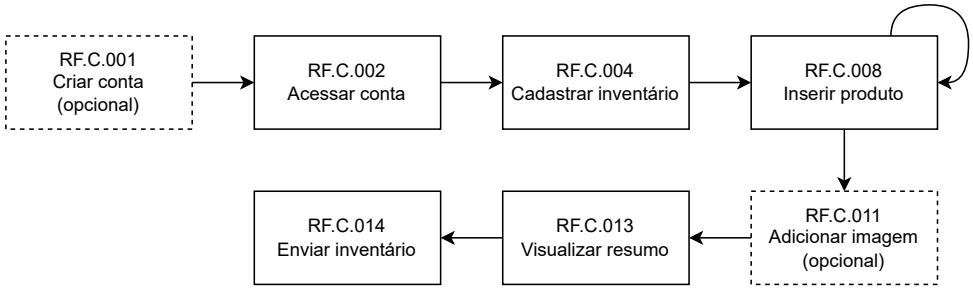
4.2.2 Fluxos de operação

Existem dois fluxos de operação principais que podem ser identificados no sistema através dos requisitos funcionais do cliente, apontados na seção 4.1.1.1, e dos requisitos funcionais do funcionário, elencados na seção 4.1.1.2. Estes fluxos de operação representam a execução ordenada de um ou mais casos de uso previamente descritos para a realização de uma tarefa com um objetivo claramente definido.

4.2.2.1 Enviar inventário para avaliação

Sendo a principal atividade realizada pelos clientes da empresa Xibuco no sistema, esta ação segue o fluxo de operações apresentado na Figura 7 abaixo, e representa toda a sequência de ações que um cliente da empresa deve realizar no sistema para acessar a sua conta, criar o seu novo inventário, adicionar os seus produtos, conferir o resumo gerado do inventário e, por fim, enviar este inventário para a avaliação dos colaboradores da Xibuco.

Figura 7 – Fluxo de operações para o envio de inventário para avaliação

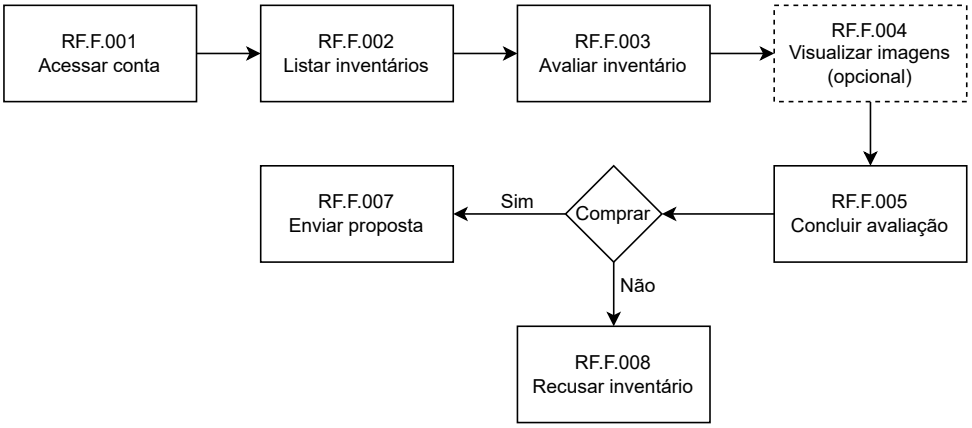


Fonte: Elaborado pelo autor

4.2.2.2 Avaliar inventário

Em contrapartida às ações realizadas pelos clientes da empresa no fluxo de envio de inventários para avaliação, descrito na seção 4.2.2.1, imediatamente anterior, o fluxo de avaliação de um desses inventários é realizado por um colaborador da Xibuco, buscando pré-classificar e avaliar a viabilidade de compra do inventário cadastrado pelo cliente no sistema. Esta atividade segue o fluxo de operações apresentado na Figura 8 a seguir.

Figura 8 – Fluxo de operações para avaliação de inventário



Fonte: Elaborado pelo autor

4.3 SERVIÇOS E PERSISTÊNCIA

Estes serviços, também chamados de *backend*, são serviços web que permitem a comunicação e a interação entre os demais componentes do sistema através da internet ou por meio de uma rede local. Além disso, eles são responsáveis por armazenar, tratar e abastecer os demais componentes com informações corretas e confiáveis, mantendo a integridade e a aderência às regras de negócio previamente especificadas na seção 4.1 e arbitrando em casos de divergência entre quaisquer outros componentes do sistema.

Assim, para assegurar a realização deste objetivo, descreve-se nas seções seguintes o conjunto de tecnologias, padrões arquiteturais e decisões de projeto adotadas, bem como apresenta-se o modelo de dados que será responsável por armazenar as informações estruturadas do sistema ao longo da sua operação.

4.3.1 Tecnologia

Em virtude da natureza das informações que se necessita armazenar, a primeira decisão de projeto foi a adoção de um banco de dados relacional, conceito apresentado na fundamentação teórica deste trabalho na seção 3.2.3, e que atenderá à estas necessidades deste projeto sem maiores preocupações.

O sistema de gerenciamento de banco de dados escolhido para o ambiente de produção foi o PostgreSQL, em sua versão 16.1. Contudo, para facilitar o trabalho de desenvolvimento e os testes do sistema, adotou-se também a utilização de uma biblioteca chamada SQLite, em sua versão 3.44.0. Assim, é necessário esclarecer que todas as interações com o sistema de gerenciamento de banco de dados, bem como o modelo de dados em si, seguem o padrão ISO/IEC 9075:2023, compatível com ambas as ferramentas.

Definida a primeira camada de persistência não volátil do sistema, a próxima decisão foi direcionada para o armazenamento das imagens dos produtos dentro do próprio modelo do banco de dados relacional. Neste ponto podemos citar algumas alternativas como: o armazenamento das imagens em um sistema de arquivos local, externo ao banco de dados, com o apontamento apenas do caminho da imagem no modelo do banco de dados relacional em si; ou mesmo a utilização de um sistema de armazenamento de objetos em nuvem, referenciando apenas a URL destes recursos à partir do modelo do banco de dados.

Contudo, nenhuma destas alternativas se faz necessária no momento, haja vista a baixa complexidade do sistema e também os níveis de utilização previstos para o sistema ao longo destas fases iniciais.

Além disso, outra decisão tecnológica relacionada ao componente dos serviços e da persistência de dados foi a escolha da linguagem de programação a ser utilizada para a implementação dos serviços propriamente ditos. Nesse ponto, a opção adotada foi a linguagem Kotlin, apresentada na seção 3.2.1.2 em conjunto com o framework de desenvolvimento de sistemas web Ktor. Esta mesma linguagem de programação foi utilizada como base tecnológica para a implementação do aplicativo móvel para a plataforma Android, que será apresentado neste trabalho na seção 4.4, e permitirá uma engenharia avançada através do uso da plataforma de desenvolvimento *Kotlin Multiplatform*, focada na reutilização de módulos entre os diferentes componentes do sistema e na definição única das regras de negócio e do domínio do problema.

Por fim, vale ressaltar a adoção de bibliotecas menores, utilizadas para garantir a implementação correta e segura de soluções como o acesso ao banco de dados relacional, através do SqlDelight, por exemplo, e a serialização de objetos para transferência via internet, através da própria solução de serialização de objetos da própria linguagem de desenvolvimento Kotlin.

4.3.2 Arquitetura

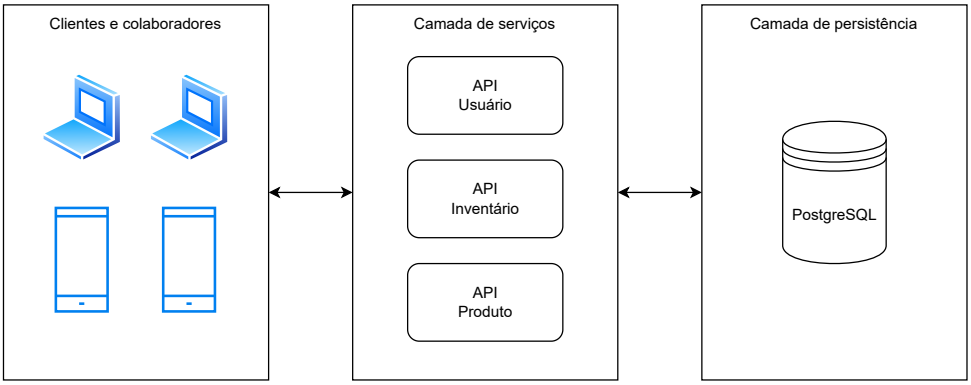
A arquitetura adotada para a implementação dos serviços de *backend* do sistema segue o padrão REST, descrito em detalhes na seção 3.2.2.4 da fundamentação teórica deste trabalho, usa os métodos do protocolo de comunicação HTTP, apresentado na seção 3.2.2.1, e define os seus modelos de dados de transporte com base no protocolo JSON, apresentado na seção 3.2.2.2 da mesma fundamentação teórica.

A estrutura geral dos serviços foi dividida em duas grandes camadas: a camada de

serviços e a camada de persistência. A camada de serviços foi subdividida em 3 módulos, com uma API de serviços para cada domínio do problema, que são: usuários, inventários e produtos. Por outro lado, a camada de persistência mantém em isolamento o sistema de gerenciamento do banco de dados, cujo modelo de armazenamento de informações está apresentado na próxima seção deste trabalho.

De forma simplificada, a Figura 9 abaixo, ilustra esta estrutura para um melhor entendimento.

Figura 9 – Arquitetura do componente de serviços e persistência



Fonte: Elaborado pelo autor

4.3.2.1 Módulo da API de usuários

O módulo da API de usuários é responsável por todos os casos de uso relacionados ao usuário propriamente dito. Este módulo expõe publicamente uma interface de métodos que estão diretamente relacionados com a entidade *user*, que será apresentada no modelo de dados do sistema na seção 4.3.3 deste relatório. Os métodos disponíveis nesta API, bem como a sua funcionalidade, estão descritos abaixo:

a) **GET /user:** Retorna a lista de usuários cadastrados no sistema. Este método aceita filtros no formato de parâmetros de consulta;

b) **POST /user:** Cria uma nova conta de usuário no sistema;

c) **GET /user/{userId}:** Retorna o usuário cadastrado no sistema com *id* igual a *userId*;

d) **PATCH /user/{userId}:** Altera informações do usuário cadastrado no sistema com *id* igual a *userId*;

e) **POST /user/auth:** Autentica um usuário no sistema.

4.3.2.2 Módulo da API de inventários

O módulo da API de inventários é responsável por todos os casos de uso relacionados ao inventário propriamente dito. Este módulo expõe publicamente uma interface de métodos que estão diretamente relacionados com a entidade *inventory*, que será apresentada no modelo de dados do sistema na seção 4.3.3 deste relatório. Os métodos disponíveis nesta API, bem como a sua funcionalidade, estão descritos abaixo:

a) **GET /inventory:** Retorna a lista de inventários cadastrados no sistema. Este método aceita filtros no formato de parâmetros de consulta;

b) **POST /inventory:** Cria um novo inventário no sistema;

c) **GET /inventory/{inventoryId}:** Retorna o inventário cadastrado no sistema com *id* igual a *inventoryId*;

d) **PATCH /inventory/{inventoryId}:** Altera informações do inventário cadastrado no sistema com *id* igual a *inventoryId*;

e) **DELETE /inventory/{inventoryId}:** Exclui o inventário cadastrado no sistema com *id* igual a *inventoryId*.

4.3.2.3 Módulo da API de produtos

O módulo da API de produtos é responsável por todos os casos de uso relacionados ao produto propriamente dito e às suas imagens. Este módulo expõe publicamente uma interface de métodos que estão diretamente relacionados com as entidades *inventory* e *image*, que serão apresentadas no modelo de dados do sistema na seção 4.3.3 deste relatório. Os métodos disponíveis nesta API, bem como a sua funcionalidade, estão descritos abaixo:

a) **GET /product:** Retorna a lista de produtos cadastrados no sistema. Este método aceita filtros no formato de parâmetros de consulta;

b) **POST /product:** Cria um novo produto no sistema;

c) **GET /product/{productId}:** Retorna o produto cadastrado no sistema com *id* igual a *productId*;

d) **PATCH /product/{productId}:** Altera informações do produto cadastrado no sistema com *id* igual a *productId*;

e) **DELETE /product/{productId}:** Exclui o produto cadastrado no sistema com *id* igual a *productId*;

f) **GET /product/{productId}/image:** Retorna a lista de imagens associada ao produto cadastrado no sistema com *id* igual a *productId*;

g) **PUT /product/{productId}/image:** Adiciona uma nova imagem ao produto com *id* igual a *productId*;

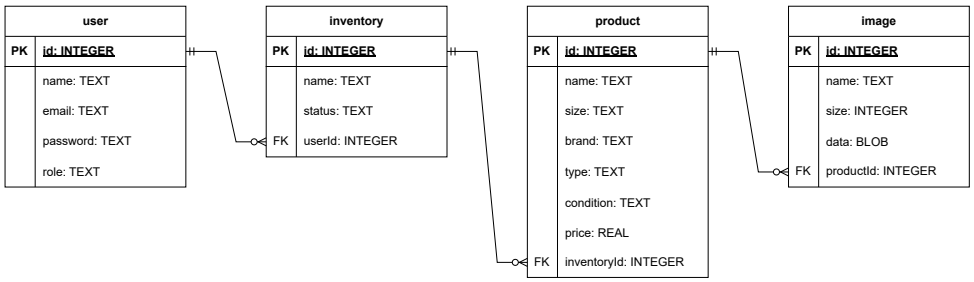
h) **GET /product/{productId}/image/{imageId}:** Retorna a imagem com *id* igual a *imageId*, associada ao produto com *id* igual a *productId*;

i) **DELETE /product/{productId}/image/{imageId}:** Exclui a imagem com *id* igual a *imageId*, associada ao produto com *id* igual a *productId*.

4.3.3 Modelo de dados

O modelo relacional de persistência de dados, desenvolvido para o sistema de compras da empresa Xibuco, contém apenas 4 entidades bastante simples. São elas: *user*, para armazenar os usuários do sistema; *inventory*, para registrar os inventários enviados ao sistema; *product*, que armazena as informações dos produtos de cada um dos inventários; e, por fim, *image*, que armazena as imagens dos produtos em formato binário. O diagrama completo do modelo entidade-relacionamento pode ser visto na Figura 10 a seguir.

Figura 10 – Modelo de dados da camada de persistência



Fonte: Elaborado pelo autor

4.4 APLICATIVO MÓVEL

O aplicativo móvel, conforme descrito na seção 4.2, sobre a arquitetura geral do sistema, é a porta de entrada do usuário mais importante deste sistema, o cliente final da Xibuco. O aplicativo móvel é responsável por apresentar para este cliente as suas opções de forma fácil, clara e rápida, estimulando uma boa interação e garantindo que os processos da empresa sejam realizados de forma correta e segura.

Com este objetivo em mente, descreve-se nas seções seguintes o conjunto de tecnologias escolhidas, apresenta-se o protótipo e as interfaces das diferentes telas do sistema, bem como os padrões arquiteturais e as decisões de projeto adotadas.

4.4.1 Tecnologia

Uma das primeiras decisões relacionadas à tecnologia do aplicativo em questão, foi a escolha das plataformas para as quais ele seria desenvolvido e publicado. Por se tratar de uma plataforma de código aberto, facilmente acessível e amplamente utilizada no mercado brasileiro, a plataforma adotada para este desenvolvimento foi o Android.

Contudo, é interessante ressaltar que, como a estrutura do sistema como um todo foi implementado fazendo uso do *Kotlin Multiplatform*, boa parte de uma eventual implementação futura para a plataforma iOS, da Apple, já está encaminhada e pode ser reutilizada a partir da implementação existente. Todavia, este esforço encontra-se fora do escopo deste trabalho.

Desta forma, as principais tecnologias adotadas para o desenvolvimento deste aplicativo estão listadas abaixo:

a) **Kotlin:** Sendo a linguagem tradicionalmente adotada para o desenvolvimento de aplicativos na plataforma Android, e já estando em uso no sistema para a implementação dos serviços de backend, conforme descrito na seção 4.3.1, a utilização da linguagem de programação Kotlin, em sua versão 1.9.20, foi uma decisão automática;

b) **Ktor:** Assim como a linguagem Kotlin já estava em uso na implementação dos serviços de backend, o framework de desenvolvimento de sistemas web Ktor também estava. Ou seja, esta foi mais uma decisão automática, dadas as possibilidades de compartilhamento de código entre os diferentes componentes do sistema. Dessa forma, o Ktor foi utilizado para auxiliar na implementação de toda a camada de sincronização do aplicativo com os serviços de backend;

c) **SqlDelight:** Assim como nos serviços de backend, a biblioteca SqlDelight foi novamente utilizada na camada de abstração do banco de dados local do aplicativo. Esta biblioteca realiza a geração de classes fortemente tipadas em Kotlin, o que proporciona uma experiência de desenvolvimento mais segura e eficiente na interação com o banco de dados local;

d) **Coroutines:** A biblioteca do próprio ecossistema Kotlin que oferece suporte à programação assíncrona e concorrente é o Coroutines. Usada em sua versão 1.7.3, o Kotlin Coroutines introduz um modelo de programação assíncrona baseado em suspensão, permitindo que o código assíncrono seja escrito de maneira sequencial, sem bloqueios;

e) **Jetpack Compose:** O framework de desenvolvimento do Jetpack Compose contém uma coleção de bibliotecas para a implementação declarativa de interfaces com o usuário. Ela foi utilizada na implementação das telas do aplicativo através da utilização de programação funcional e reativa, com as definições da interface implementadas diretamente na linguagem Kotlin.

4.4.2 Protótipos e interfaces

Com base nos requisitos do cliente que direcionam fortemente a implementação do aplicativo móvel, conforme visto na seção 4.1.1.1 deste trabalho, pode-se extrapolar a implementação do sistema para 5 telas de interação com o usuário final. Cada uma destas telas está apresentada em detalhes nas seções seguintes.

4.4.2.1 Tela de criação de conta do usuário

Esta é a tela utilizada pelo usuário para criar uma nova conta no sistema, caso ele ainda não a possua. Assim, esta interface é responsável por atender o requisito do cliente RF.C.001. Veja a Figura 11 abaixo com o protótipo da tela.

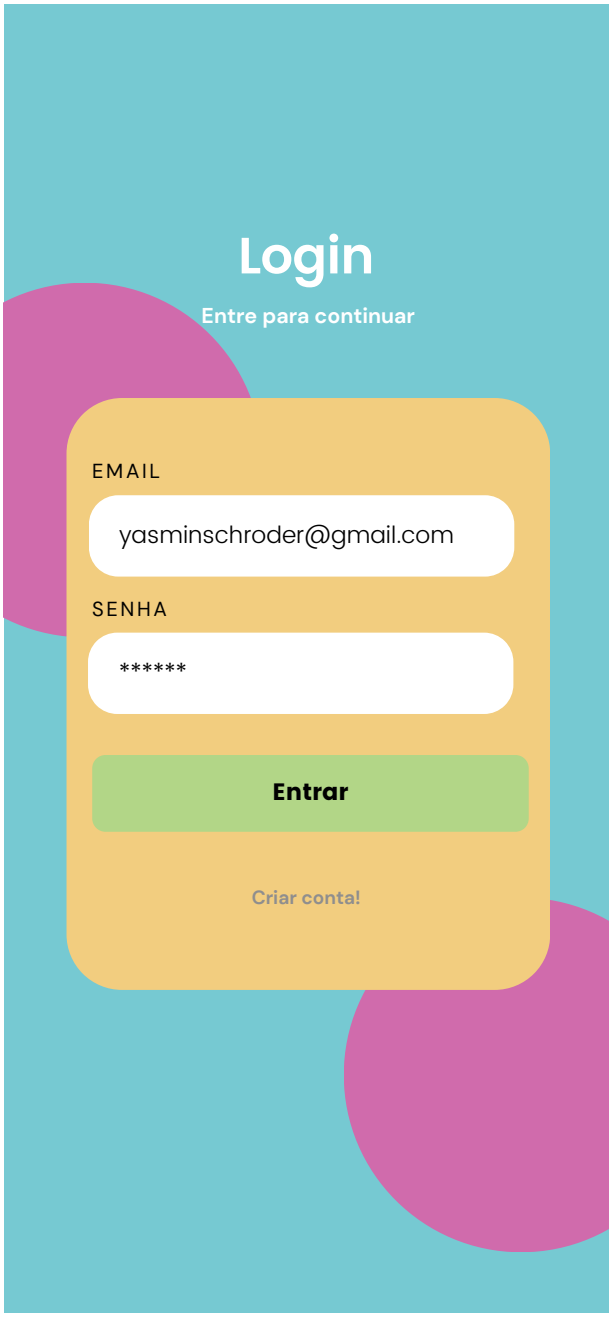
Figura 11 – Protótipo da tela de criação de conta do usuário

Fonte: Elaborado pelo autor

4.4.2.2 Tela de autenticação do usuário

Esta é a primeira tela vista pelo usuário ao acessar o aplicativo. Ela é responsável por coletar os dados de autenticação do usuário e redirecioná-lo para a tela da lista de inventários descrita na sequência. Importante destacar que esta tela atende o requisito do cliente RF.C.002. Veja a Figura 12 a seguir com o protótipo da tela.

Figura 12 – Protótipo da tela de autenticação do usuário

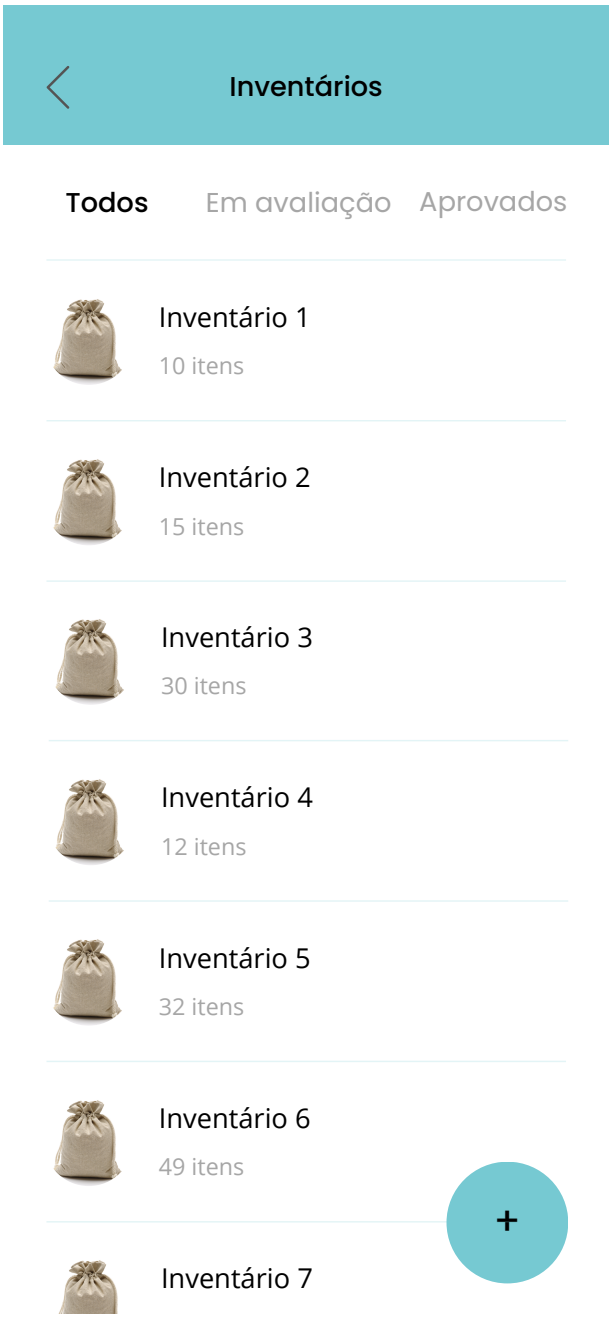


Fonte: Elaborado pelo autor

4.4.2.3 Tela da lista de inventários

Após a autenticação do usuário, ele é imediatamente direcionado para esta interface, podendo iniciar a execução do fluxo de envio de um inventário, conforme descrito na seção 4.2.2.1 deste trabalho. A Figura 13 abaixo mostra os detalhes desta tela.

Figura 13 – Protótipo da tela da lista de inventários

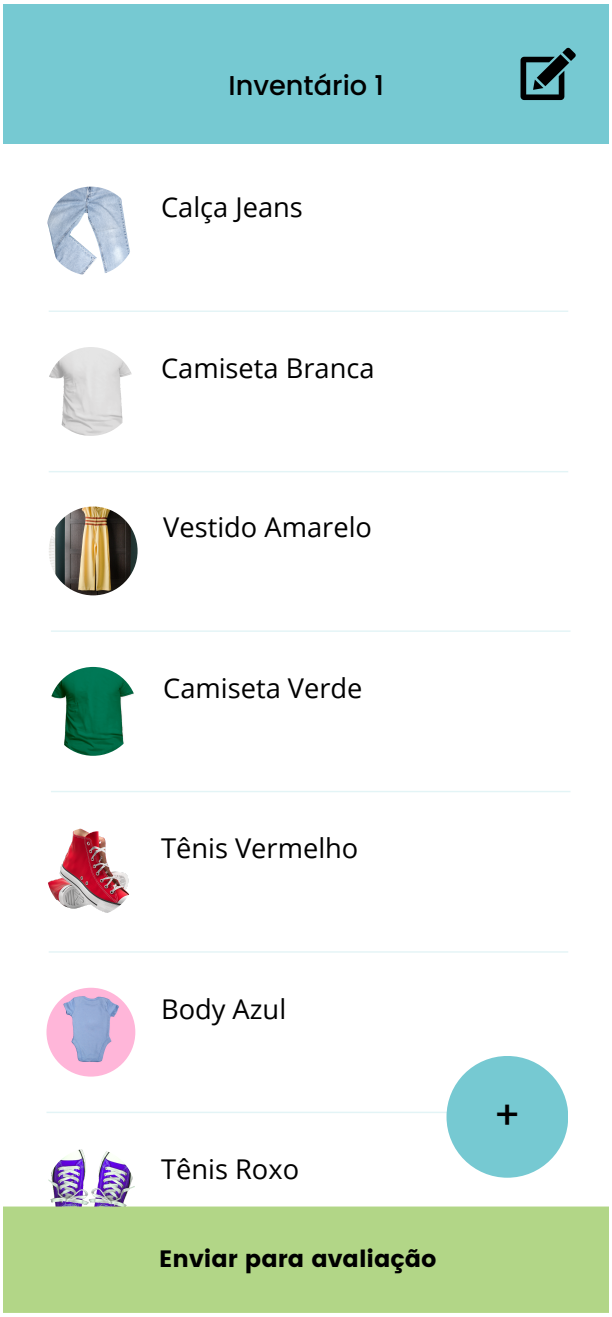


Fonte: Elaborado pelo autor

4.4.2.4 Tela da lista de produtos

Esta interface apresenta a lista de produtos cadastrada em um determinado inventário do sistema, podendo ser utilizada para executar os casos de uso relacionados aos produtos deste inventário. Veja a Figura 14 abaixo com os detalhes do protótipo.

Figura 14 – Protótipo da tela da lista de produtos

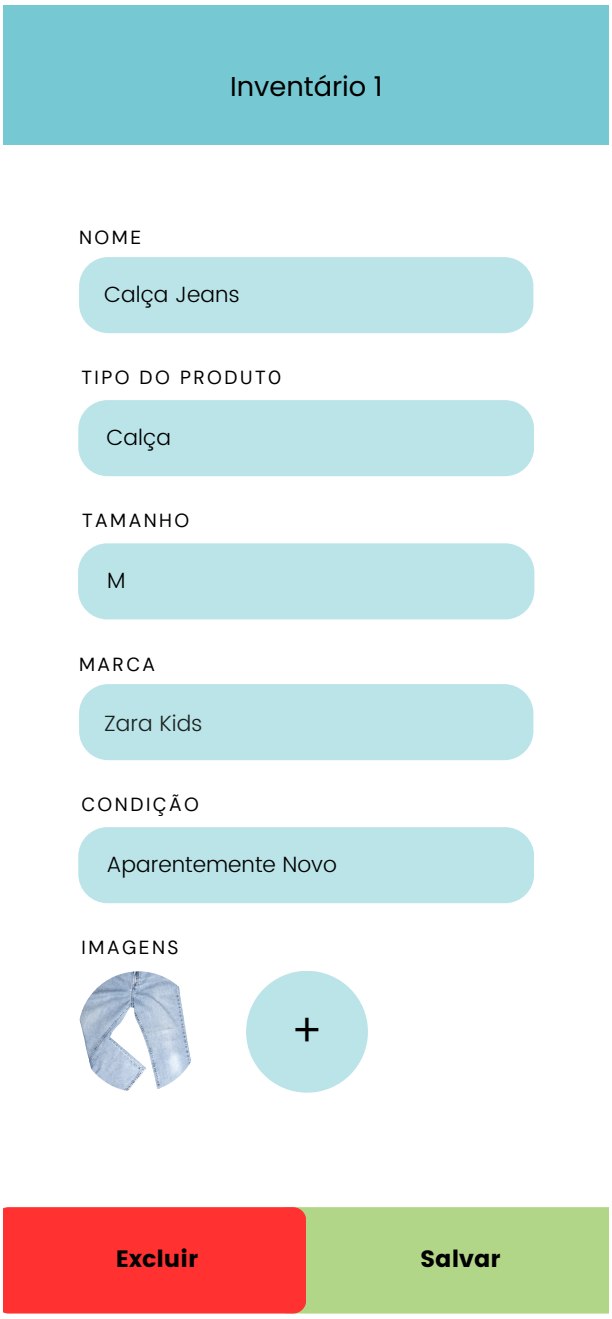


Fonte: Elaborado pelo autor

4.4.2.5 Tela do formulário do produto

Esta tela apresenta o formulário de cadastro de um produto em um inventário. Ela é a principal tela de coleta das informações que o usuário precisa fornecer para enviar um inventário para avaliação na Xibuco. Veja a Figura 15 abaixo com os detalhes deste formulário.

Figura 15 – Protótipo da tela do formulário do produto



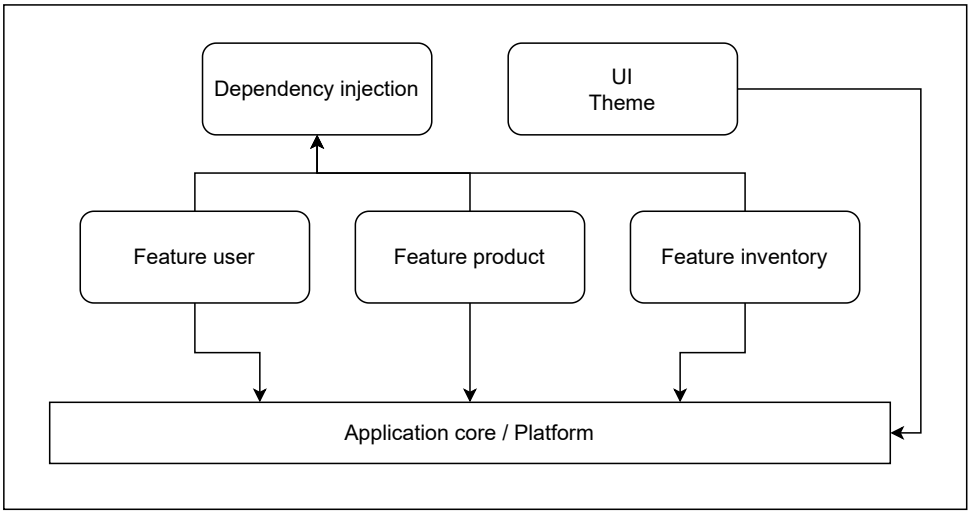
Fonte: Elaborado pelo autor

4.4.3 Arquitetura

A arquitetura escolhida para a implementação do aplicativo Android segue o padrão de desenvolvimento MVVM, conforme elucidado na seção 3.1.3.2 deste relatório, adotando também as boas práticas definidas pelo autor Robert C. Martin em seu livro *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, descritas com maior profundidade na seção 3.1.3.1 deste trabalho.

A estrutura geral dos componentes do aplicativo Android foi dividida em 6 grandes componentes: o primeiro sendo a base da aplicação, também chamado de *core*; um segundo chamado de *dependency injection*, responsável por conectar os componentes internos de forma dinâmica; um outro componente responsável pelas definições de interface, chamado de *ui theme* e, por fim, um componente para cada grande funcionalidade do sistema, que são: usuários, inventários e produtos. Mais detalhes dessa estrutura podem ser compreendidos com a Figura 16 a seguir.

Figura 16 – Arquitetura geral dos componentes do aplicativo Android



Fonte: Elaborado pelo autor

4.4.3.1 Base da aplicação e componentes utilitários

A base da aplicação, também chamada de *core*, em conjunto com os dois módulos utilitários do aplicativo Android, *dependency injection* e *ui theme*, são a fundação sobre a qual as funcionalidades do sistema são construídas.

A base da aplicação em si contém as estruturas e arquivos necessários para que o software seja corretamente reconhecido como um aplicativo pelo sistema operacional Android. Além disso, ela contém as informações necessárias para iniciar a aplicação, uma vez que o seu ícone seja acionado na tela do sistema operacional do usuário, inicializando os seus serviços internos, carregando as funcionalidades do sistema e apresentando a primeira tela ao usuário.

Já o componente de *dependency injection*, em tradução livre chamado de injeção de dependências, executa exatamente a função que o seu nome sugere. Este componente é responsável por prover para os demais componentes do sistema tudo aquilo que eles necessitarem para funcionar. Este padrão de injeção de dependências promove a separação de responsabilidades entre os diferentes módulos do sistema, aumentando assim a sua manutenibilidade e, principalmente, a sua testabilidade.

Por fim, o último componente utilitário desta aplicação é o *ui theme*. Este componente define estaticamente todas as informações necessárias para que a interface do usuário seja apresentada de forma correta. Isto é, ele contém as definições de cores, espaçamentos, tipos e tamanhos das fontes, formatos de apresentação dos estilos do aplicativo, entre outros. A principal razão para a existência deste módulo é a definição de um local central com todas as informações relativas ao design da aplicação, novamente aumentando a manutenibilidade do sistema a longo prazo.

4.4.3.2 Componente dos usuários

O componente dos usuários é responsável por oferecer ao sistema os casos de uso relacionados aos usuários do sistema. Para a aplicação em questão, estes casos de uso são a criação de novas contas de usuário, RF.C.001, e a autenticação dos usuários, RF.C.002.

De forma abrangente, todos os componentes que implementam funcionalidades do sistema possuem a mesma arquitetura interna, sendo subdivididos em três partes menores, que são: a camada de dados, chamada no sistema de *data*; a camada do domínio, chamada no sistema de *domain*; e a camada de apresentação, chamada no sistema de *presentation*. Cada uma dessas camadas possui subdivisões internas que estão descritas abaixo:

a) ***Data source:*** Pertencente a camada de dados, contém os objetos de acesso ao banco de dados local do aplicativo, relacionados ao componente em questão, além de explicitar as consultas realizadas sobre a base de dados;

b) ***Repository implementation:*** Pertencente a camada de dados, contém a implementação real do conceito de um repositório de informações, relacionado ao componente em questão. Este repositório segue os padrões de acesso às informações definidas pela plataforma, sejam elas no banco de dados local, disponível no pacote *data source*, ou mesmo através da chamada de algum serviço externo via internet, como a API REST do módulo de serviços e persistência do sistema, por exemplo;

c) ***Model:*** Pertencente a camada do domínio, este pacote contém os objetos da modelagem do problema. No caso do componente de usuários, este pacote contém a definição clara e objetiva do que é um usuário dentro do sistema;

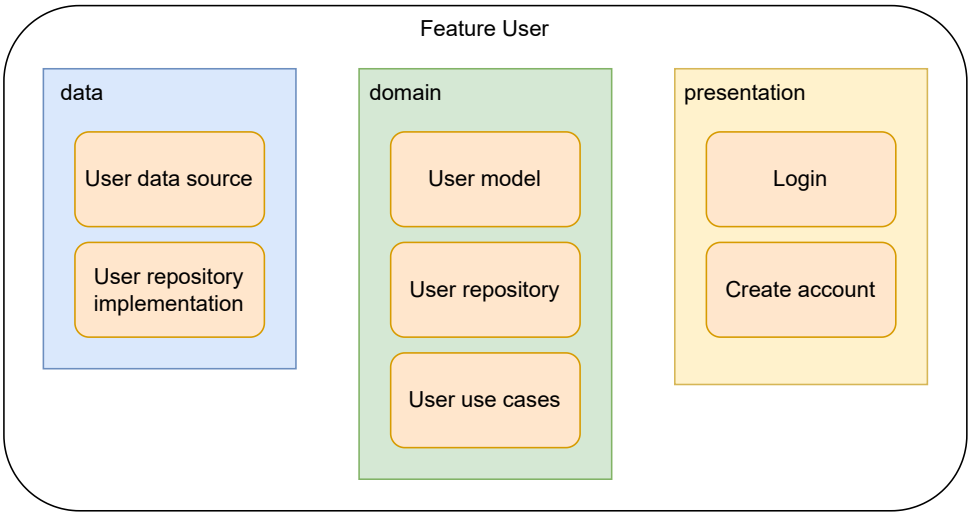
d) ***Repository:*** Pertencente a camada do domínio, contém, dentro do domínio do componente, a definição de como as informações serão acessadas através da imple-

mentação de um repositório, que está disponível no pacote *repository implementation*. Este pacote define apenas as interfaces de acesso aos dados;

e) ***Use cases:*** Pertencente a camada do domínio, contém toda a lógica e as regras de negócio necessárias para implementar cada um dos casos de uso do componente em questão.

Os elementos descritos anteriormente podem ser observados, dentro de suas respectivas camadas *data* e *domain*, na Figura 17 abaixo:

Figura 17 – Estrutura interna do componente dos usuários



Fonte: Elaborado pelo autor

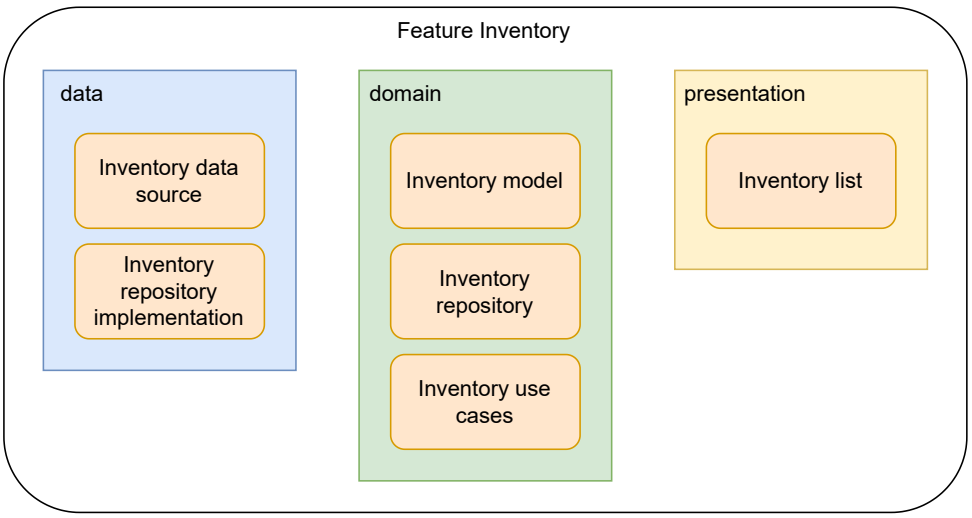
Além dos pacotes descritos anteriormente, pode-se observar a camada de apresentação na Figura 17. Camada esta que varia de componente para componente. No caso do componente dos usuários, esta camada trás as implementações concretas de toda a interface do usuário para as respectivas telas de criação de contas e de autenticação do usuário. Ou seja, esta camada está extremamente suscetível a mudanças, sendo responsável por acomodar qualquer alteração necessária na comunicação com os usuários do aplicativo.

Por fim, é importante destacar que esta camada contém as estruturas da *view* e da *viewmodel*, descritas em detalhes na seção 3.1.3.2 deste trabalho.

4.4.3.3 Componente dos inventários

O componente dos inventários, assim como o componente dos usuários, descrito anteriormente, possui as mesmas subdivisões internas com a mesma estrutura arquitetural para implementar as suas funcionalidades. Os detalhes sobre o conteúdo de cada pacote podem ser verificados na seção 4.4.3.2 imediatamente anterior. Abaixo, podemos observar a Figura 18, que apresenta a estrutura completa do componente dos inventários.

Figura 18 – Estrutura interna do componente dos inventários



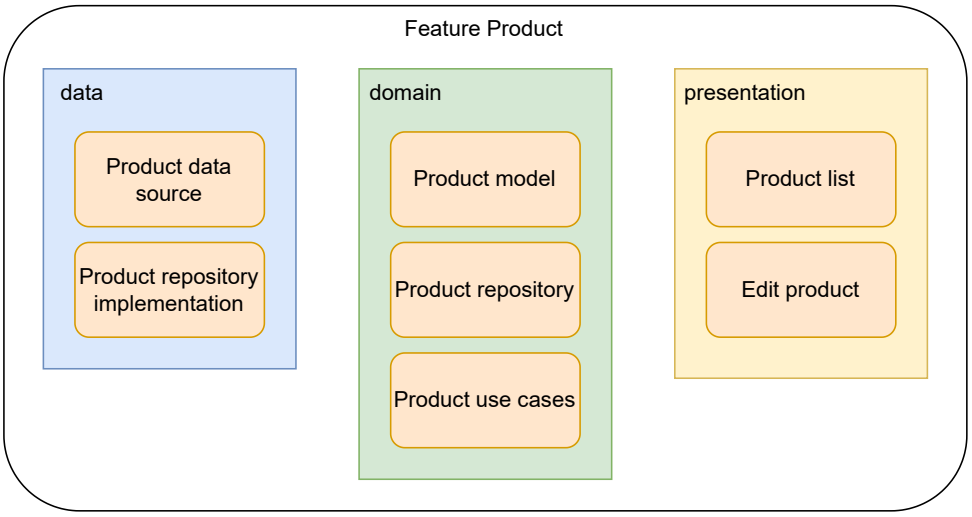
Fonte: Elaborado pelo autor

Pode-se observar que a camada de apresentação da Figura 18 oferece apenas uma funcionalidade nova ao sistema, sendo responsável por exibir a lista de inventários cadastrados no sistema ao usuário final do aplicativo.

4.4.3.4 Componente dos produtos

Por fim, o componente dos produtos também segue a mesma estrutura arquitetural descrita nos dois componentes anteriores. Assim, os detalhes sobre o conteúdo de cada pacote podem ser verificados na seção 4.4.3.2, sobre o componente dos usuários. Abaixo, podemos observar a Figura 19, que apresenta a estrutura completa do componente dos produtos.

Figura 19 – Estrutura interna do componente dos produtos



Fonte: Elaborado pelo autor

Novamente, como nos dois componentes anteriormente descritos, o componente de produtos exibido na Figura 19, trás duas novas funcionalidades à interface do usuário do aplicativo dentro da sua camada de apresentação: a lista de produtos e a capacidade de se criar e editar produtos.

4.5 AMBIENTE DE ADMINISTRAÇÃO

O ambiente de administração do sistema, de acordo com o que foi apresentado na seção 4.2, sobre a arquitetura geral do sistema, será utilizado pelos colaboradores da Xibuco para realizar todas as ações necessárias para operar o sistema e interagir com os clientes da empresa.

Desta maneira, o ambiente de administração precisa operar de forma rápida e objetiva, minimizando o tempo e as ações necessárias para a execução dos casos de uso anteriormente descritos nas seções 4.1.1.2 e 4.1.1.3.

Com este objetivo em mente, descreve-se nas seções seguintes o conjunto de tecnologias escolhidas, apresenta-se o protótipo e as interfaces das diferentes telas do sistema, bem como os padrões arquiteturais e as decisões de projeto adotadas.

4.5.1 Tecnologia

Ao contrário do aplicativo móvel apresentado na seção 4.4, o ambiente de administração do sistema não possui necessariamente uma plataforma para ser escolhida, haja vista que ela deverá funcionar a partir de um navegador de internet tradicional.

Portanto, seguindo o uso das tecnologias aplicadas no desenvolvimento do aplicativo Android e dos serviços de *backend*, o ambiente de administração também foi implementado fazendo uso do *Kotlin Multiplatform*, reaproveitando boa parte da implementação já realizada nas etapas anteriores deste trabalho.

Assim, as tecnologias citadas na seção 4.4.1, sobre o desenvolvimento do aplicativo Android, foram também utilizadas para a implementação do ambiente de administração, com a única exceção sendo o *Jetpack Compose*, que no caso do ambiente de administração foi substituído pela biblioteca *Compose Multiplatform*, com a mesma aplicabilidade do anterior.

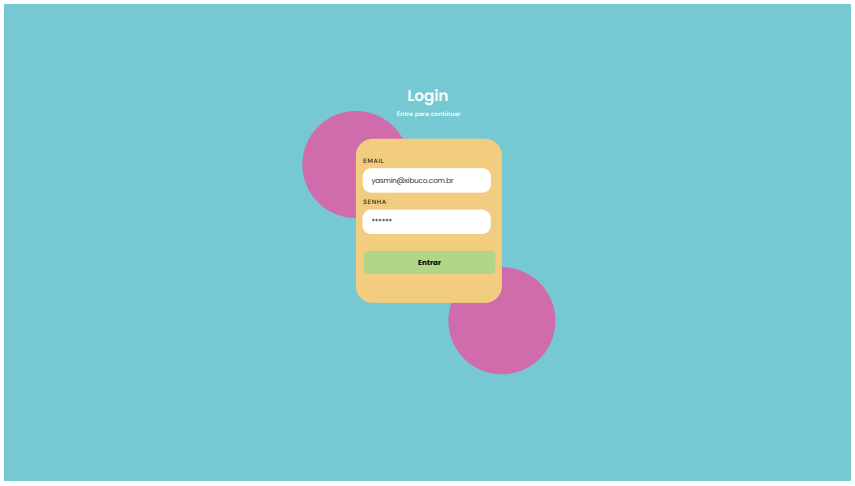
4.5.2 Protótipos e interfaces

Com base nos requisitos do funcionário e nos requisitos do administrador do sistema, conforme descrito nas seções 4.1.1.2 e 4.1.1.3 deste trabalho, pode-se extrapolar a implementação do sistema para 4 telas de interação com o usuário. Cada uma dessas telas está apresentada em detalhes nas seções seguintes.

4.5.2.1 Tela de autenticação do colaborador

Esta é a primeira tela vista pelo usuário ao acessar o sistema. Ela é responsável por coletar os dados de autenticação do usuário e redirecioná-lo para a tela da lista de inventários descrita na sequência. Importante destacar que esta tela atende o requisito do funcionário RF.F.001. Veja a Figura 20 a seguir com o protótipo da tela.

Figura 20 – Protótipo da tela de autenticação do colaborador



Fonte: Elaborado pelo autor

4.5.2.2 Tela da lista de inventários para avaliação

Após a autenticação do colaborador, ele é imediatamente direcionado para esta interface, podendo iniciar a execução do fluxo de avaliação de um inventário, conforme descrito na seção 4.2.2.2 deste trabalho. A Figura 21 abaixo mostra os detalhes desta tela.

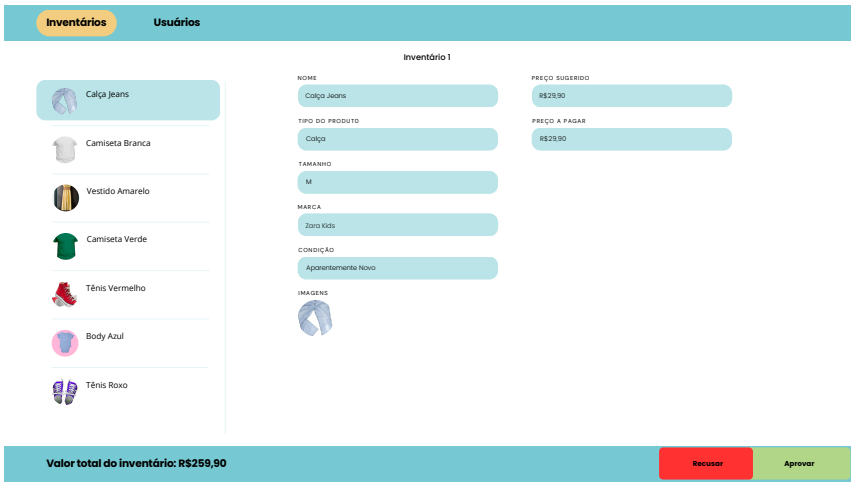
Figura 21 – Protótipo da lista de inventários para avaliação

Fonte: Elaborado pelo autor

4.5.2.3 Tela da lista de produtos para avaliação

Esta interface apresenta a lista de produtos cadastrada em um determinado inventário do sistema, sendo utilizada para executar os casos de uso relacionados à avaliação destes produtos. Veja a Figura 22 abaixo com os detalhes do protótipo.

Figura 22 – Protótipo da tela da lista de produtos para avaliação

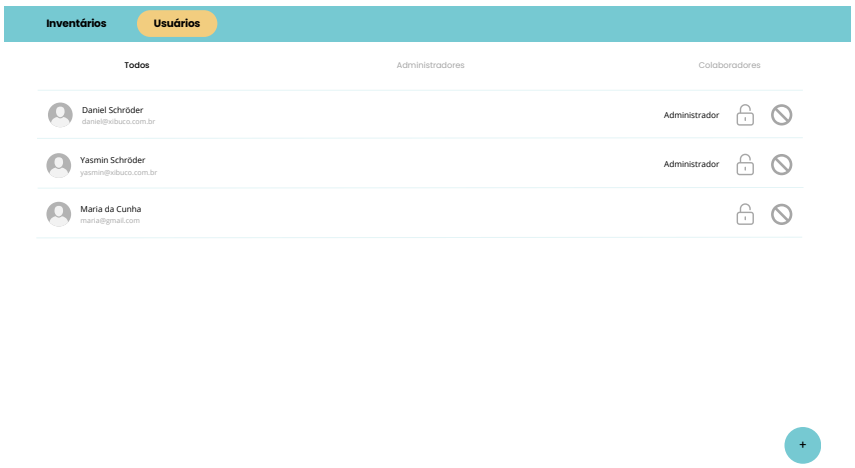


Fonte: Elaborado pelo autor

4.5.2.4 Tela da lista de usuários do sistema

Esta tela é de uso restrito aos administradores do sistema. Nela é possível executar os casos de uso descritos na seção 4.1.1.3 deste trabalho, como por exemplo, criar usuários, alterar o seu tipo de conta, recuperar a senha de um colaborador, entre outros. Veja a Figura 23 abaixo com os detalhes do protótipo.

Figura 23 – Protótipo da tela da lista de usuários do sistema



Fonte: Elaborado pelo autor

4.5.3 Arquitetura

Dado o grande reaproveitamento da implementação do aplicativo Android para o ambiente de administração do sistema, através do uso das mesmas tecnologias e do *Kotlin Multiplatform*, pode-se verificar os seguintes fatos:

a) A arquitetura geral dos componentes do aplicativo Android foi completamente reutilizada para a implementação do ambiente de administração. Ou seja, a Figura 16, apresentada na seção 4.4.3 é também válida para o ambiente de administração do sistema;

b) Os componentes com as funcionalidades dos usuários, dos inventários e dos produtos, apresentados respectivamente nas seções 4.4.3.2, 4.4.3.3 e 4.4.3.4, não sofreram alterações diretas nas suas camadas da logica de negócio em código preexistente. É importante destacar que as camadas de apresentação destes componentes foram reimplementadas em função da diversidade das telas entre as duas aplicações;

c) Os casos de uso do administrador, que não estavam presentes no aplicativo Android, foram facilmente incluídos na camada da lógica de negócios da funcionalidade dos usuários.

Com a implementação da camada de persistência e dos serviços de backend, apresentados na seção 4.3, o desenvolvimento do aplicativo móvel, visto na seção 4.4 anterior, e a implementação deste ambiente administrativo, a versão preliminar do sistema de gestão de compras para a empresa Xibuco Comércio Infantil Online está concluída e o seu uso pode ser iniciado em processo de avaliação.

Infelizmente, durante o desenvolvimento deste trabalho a empresa foi dissolvida em função dos seus altos custos operacionais. Contudo, é importante destacar que o aprendizado obtido com o desenvolvimento deste sistema, além de não se perder, pode ser utilizado em um domínio mais amplo, inclusive com a possibilidade de o sistema ser reestilizado e repassado para outras empresas do mesmo segmento.

5 VALIDAÇÃO

Este capítulo está intrinsecamente ligado à seção 4.1, sobre a análise de requisitos do sistema, apresentados no capítulo de desenvolvimento do trabalho, imediatamente anterior. A seção 5.1.3, apresentará a correlação entre os requisitos funcionais descritos no capítulo anterior, e o seu atendimento pelo sistema implementado. Além disso, a seção 5.2, apresentará uma avaliação direta sobre o atendimento dos requisitos não funcionais especificados anteriormente. É importante ressaltar que, como a empresa Xibuco Comércio Infantil Online encerrou as suas atividades antes da conclusão deste trabalho, não foi possível avaliar os resultados completos em seu ambiente esperado. Contudo, pode-se extrapolar os resultados pelos objetivos atingidos.

Além disso, em virtude da natureza modesta dos casos de uso especificados na seção 4.1, não se fez necessária a criação e a implementação de testes unitários e de integração para a validação do funcionamento do sistema desenvolvido.

5.1 REQUISITOS FUNCIONAIS

Assim como na seção 4.1, apresentada em 3 subseções sobre os requisitos funcionais dos clientes, dos funcionários e dos administradores do sistema, esta seção de validação dos requisitos também está assim subdividida.

5.1.1 Requisitos funcionais do cliente

Segue abaixo a relação completa de requisitos funcionais do cliente, com a sua respectiva apresentação de como o requisito foi atendido no desenvolvimento do sistema:

a) **RF.C.001:** Apresentado no item 4.4.2.1, da seção de protótipos e interfaces do aplicativo móvel, pode-se observar que o requisito deste item está atendido;

b) **RF.C.002:** Apresentado no item 4.4.2.2, da seção de protótipos e interfaces do aplicativo móvel, pode-se observar que o usuário que já possui uma conta no sistema pode realizar o login através do seu email e senha;

c) **RF.C.003:** No item 4.4.2.3, da seção de protótipos e interfaces do aplicativo móvel, pode-se observar que o requisito deste item está atendido e apresenta a lista de inventários do cliente. Acessando-se cada um dos inventários, as opções para enviar e/ou excluir um inventário se revelam;

d) **RF.C.004:** Também através da tela apresentada em 4.4.2.3, no canto inferior direito, pode-se observar um botão azul para a inclusão de novos inventários no sistema;

e) **RF.C.005:** Na tela apresentada no item 4.4.2.3, ao se acessar um inventário clicando sobre o seu elemento na lista, pode-se editar o inventário em questão se, e somente

se, este ainda não foi enviado para a avaliação. Esta edição é realizada através da tela de edição de produtos no inventário, apresentada no item 4.4.2.4;

f) **RF.C.006:** Na tela apresentada no item 4.4.2.4, ao se interagir com o lápis no canto superior direito, revela-se uma opção para excluir o inventário atual, atendendo ao requisito especificando neste elemento;

g) **RF.C.007:** Conforme apresentado anteriormente, a lista de produtos de um determinado inventário pode ser observada na tela apresentada no item 4.4.2.4;

h) **RF.C.008:** No canto inferior direito da tela apresentada no item 4.4.2.4, com a lista de produtos do inventário, pode-se observar um botão azul para a inclusão de novos produtos no inventário selecionado. A tela com o formulário completo para a inclusão do produto está apresentada no item 4.4.2.5, atendendo ao requisito especificado;

i) **RF.C.009:** Ao acessar qualquer produto na tela apresentada no item 4.4.2.4, com a lista de produtos, o cliente será levado ao formulário de edição daquele produto, apresentado no item 4.4.2.5, podendo editá-lo diretamente neste local;

j) **RF.C.010:** Assim como para o atendimento do requisito anterior, ao acessar um produto na lista de produtos apresentada em 4.4.2.4, o cliente poderá também excluí-lo, se assim desejar, na tela do produto, apresentada em 4.4.2.5;

k) **RF.C.011:** Na parte inferior da tela apresentada no item 4.4.2.5, com o formulário do produto, existe um botão azul para a inclusão de imagens no respectivo produto;

l) **RF.C.012:** Na parte inferior da tela apresentada no item 4.4.2.5, com o formulário do produto, ao se manter pressionada uma imagem previamente adicionada ao produto, revela-se a opção para excluir a imagem;

m) **RF.C.013:** A tela com a lista de produtos do inventário, apresentada no item 4.4.2.4, apresenta as informações necessárias do inventário e atua como um resumo para o cliente antes do envio do inventário para a avaliação;

n) **RF.C.014:** Na parte inferior da tela com a lista de produtos do inventário, apresentada no item 4.4.2.4, existe um botão verde para submeter o inventário para a avaliação da Xibuco.

5.1.2 Requisitos funcionais do funcionário

Segue abaixo a relação completa de requisitos funcionais do funcionário, com a sua respectiva apresentação de como o requisito foi atendido no desenvolvimento do sistema:

a) **RF.F.001:** Apresentado no item 4.5.2.1, da seção de protótipos e interfaces do componente de administração do sistema, pode-se observar que o requisito deste item está atendido;

b) **RF.F.002:** Conforme apresentado no item 4.5.2.2, que mostra a lista de inventários para avaliação no componente de administração do sistema, pode-se observar que o requisito deste item está atendido;

c) **RF.F.003:** Este requisito está atendido no sistema através da tela apresentada no item 4.5.2.3, que exibe os produtos de um determinado inventário, bem como o formulário para a sua avaliação;

d) **RF.F.004:** Apresentado no item 4.5.2.3, na parte inferior do formulário do produto, é possível observar as imagens adicionadas pelo cliente ao produto selecionado, atendendo o requisito especificado para o sistema;

e) **RF.F.005:** No canto inferior direito da tela apresentada no item 4.5.2.3, com a lista de produtos de um inventário em avaliação, é possível observar um botão verde. Este botão pode ser utilizado pelo colaborador da Xibuco para marcar um inventário como avaliado no sistema;

f) **RF.F.006:** A tela apresentada no item 4.5.2.2, com a lista de inventários para avaliação, exibe na lateral direita o estado atual de cada inventário no sistema. Ao acessar o inventário, o colaborador poderá seguir com o restante do processo de interação com o cliente;

g) **RF.F.007:** Após marcar o inventário como avaliado no botão verde disponível no canto inferior direito da tela apresentada no item 4.5.2.3, abre-se um formulário ao colaborador para que se confirme o envio da proposta de compra ao cliente;

h) **RF.F.008:** Apresentado na tela descrita no item 4.5.2.3, no canto inferior direito existe um botão vermelho que, quando utilizado pelo colaborador, iniciará o fluxo para a comunicação da recusa de compra de um inventário ao cliente.

5.1.3 Requisitos funcionais do administrador

Segue abaixo a relação completa de requisitos funcionais do administrador, com a sua respectiva apresentação de como o requisito foi atendido no desenvolvimento do sistema:

a) **RF.A.001:** No canto inferior direito da tela apresentada no item 4.5.2.4 existe um botão azul. Este botão pode ser utilizado pelos administradores do sistema para a inclusão de novos usuários, atendendo ao requisito especificado do sistema;

b) **RF.A.002:** A listagem de usuários especificada neste requisito pode ser observada na tela apresentada no item 4.5.2.4;

c) **RF.A.003:** Na listagem de usuários do sistema, apresentada na tela do item 4.5.2.4, ao acessar um determinado usuário, revela-se a opção para alterar o tipo de conta do usuário;

d) **RF.A.004:** Na listagem de usuários do sistema, apresentada na tela do item 4.5.2.4, ao pressionar o primeiro botão da direita para a esquerda no elemento do usuário, a conta deste usuário será desativada, caso ela esteja ativa;

e) **RF.A.005:** Na listagem de usuários do sistema, apresentada na tela do item 4.5.2.4, ao pressionar o primeiro botão da direita para a esquerda no elemento do usuário, a conta deste usuário será reativada, caso ela esteja desativada;

f) **RF.A.006:** Na listagem de usuários do sistema, apresentada na tela do item 4.5.2.4, ao pressionar o segundo botão da direita para a esquerda no elemento do usuário, o administrador do sistema poderá alterar a senha de um determinado usuário.

5.2 REQUISITOS NÃO FUNCIONAIS

Em paralelo aos requisitos não funcionais apresentados na seção 4.1.2 do desenvolvimento deste trabalho, segue abaixo a lista destes requisitos, bem como uma avaliação direta do seu atendimento pela implementação do sistema:

a) **RNF.001 - Desempenho:** Para o caso do aplicativo móvel, todos os procedimentos de sincronização de informações entre o usuário e a camada de serviços ocorre em segundo plano no aplicativo. Ou seja, as informações são manipuladas localmente para posteriormente serem trafegadas entre os diferentes componentes do sistema sem interação do usuário. Além disso, para o caso do ambiente de administração, as informações são carregadas de forma incremental, fazendo com que a percepção do usuário seja a de que o sistema está sempre disponível e que este opera de maneira eficiente;

b) **RNF.002 - Segurança:** Os dados trafegados e persistidos estão criptografados diretamente no nível do ambiente em que o sistema opera. Ou seja, os sistemas de armazenamento possuem criptografia em repouso e toda transmissão de dados acontece através do uso do protocolo SSL (*Secure Sockets Layer*);

c) **RNF.003 - Usabilidade:** As interfaces de usuário foram implementadas através das bibliotecas de desenvolvimento das próprias plataformas sobre as quais as aplicações operam. Assim, as funcionalidades de acessibilidade são atingidas através dos próprios recursos disponíveis nos sistemas operacionais destas plataformas;

d) **RNF.004 - Confiabilidade:** Em virtude de o sistema não ter sido avaliado em ambiente produtivo, não foi possível metrificar e/ou avaliar a confiabilidade do sistema desenvolvido;

e) **RNF.005 - Escalabilidade:** Assim como para o caso da confiabilidade, este requisito não funcional não pode ser avaliado. Contudo, em se tratando de um número bastante limitado de clientes, os recursos computacionais necessários para garantir o nível desejado de atendimento não é elevado;

f) **RNF.006 - Manutenibilidade:** O código-fonte do sistema foi implementado em conformidade com boas práticas de desenvolvimento. Foi respeitado o padrão estrutural MVVM (*Model View Viewmodel*) para o desenvolvimento dos componentes, em conjunto com as definições da *Clean Architecture*, apresentada na seção 4.4.3 deste trabalho;

g) **RNF.007 - Compatibilidade:** Esta compatibilidade se atinge através do uso das bibliotecas *Ktor*, para comunicação entre os diferentes componentes do sistema via REST (*Representational State Transfer*) e no uso das bibliotecas do *Compose* para as interfaces de usuário. Em ambos os casos, os requisitos são atendidos;

h) **RNF.008 - Regulamentação:** Como o sistema não foi disponibilizado para os usuários, este requisito não funcional não pode ser avaliado diretamente. Contudo, as ações regulatórias determinadas pela LGPD podem ser atingidas através do ambiente de administração do sistema;

i) **RNF.009 - Disponibilidade:** Requisito não avaliado pois o sistema não foi disponibilizado aos usuários;

j) **RNF.010 - Interoperabilidade:** Todas as APIs (*Application Programming Interface*) do sistema podem ser diretamente utilizadas para realizar a integração com serviços de terceiros.

Assim, com base na avaliação direta dos requisitos funcionais e não funcionais do sistema, pode-se afirmar que o software desenvolvido atende suficientemente a especificação apresentada na seção 4.1 com a análise de requisitos do sistema.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Conforme apresentado no capítulo 5, sobre a validação do sistema implementado frente à especificação dos requisitos realizada na seção 4.1, pode-se observar que o sistema atende de maneira satisfatória as definições propostas. Contudo, em se tratando de um projeto de desenvolvimento de software, pode-se apontar também uma coleção de melhorias, evoluções e inclusões de novos componentes para deixar o produto ainda melhor.

Algumas destas possíveis evoluções estão descritas abaixo:

a) **Aplicativo iOS:** A estrutura de código compartilhado utilizado para a implementação do aplicativo móvel Android, apresentado na seção 4.4, pode ser reaproveitada para a criação de um aplicativo direcionado para a plataforma iOS da Apple. Isto aumentaria a área de cobertura da gama de clientes da empresa e permitiria que os usuários desta plataforma também se beneficiassem com o sistema de compras;

b) **Aplicação *desktop*:** Assim como o código desenvolvido para o aplicativo móvel pode ser reaproveitado para uma segunda plataforma, o código do ambiente de administração do sistema também pode ser reutilizado para a criação de uma versão *desktop* do sistema, mais eficiente e sem a necessidade de conexão constante com a internet;

c) **Autenticação centralizada:** A versão implementada do sistema possui um serviço de autenticação de usuários que integra a camada de serviços da própria aplicação. Este módulo de autenticação pode ser externalizado e implementado com o uso de serviços de terceiros, aumentando a segurança do sistema e disponibilizando novas funcionalidades como a autenticação centralizada ou mesmo a autenticação através de contas de redes sociais;

d) **Testes unitários e de integração:** Em virtude da natureza dos casos de uso especificados para o sistema, não se fez necessária a implementação de testes unitários e de integração. Contudo, o desenvolvimento de uma plataforma robusta com essa funcionalidade é imprescindível para o crescimento ordenado do sistema;

e) **Integração com *e-commerce*:** Pode-se implementar uma integração direta entre o sistema de compras desenvolvido no trabalho e o *e-commerce* da empresa, tanto para os clientes, que poderiam utilizar uma plataforma unificada para comprar e vender os seus produtos, quanto para os colaboradores, que poderiam gerenciar todos os processos em um único sistema centralizado.

7 CONCLUSÃO

A conclusão deste trabalho acadêmico representa um esforço na busca por soluções abrangentes e inovadoras para otimizar o sistema de gestão de compras da empresa Xibuco. A elaboração de um modelo de persistência robusto oferece a base sólida necessária para assegurar a integridade e a acessibilidade dos dados, contribuindo significativamente para a eficiência operacional do sistema.

A criação e implementação dos serviços de backend constituem um passo fundamental na materialização deste modelo de persistência, proporcionando uma infraestrutura coesa e eficaz para a interação contínua com os dados do sistema. Essa abordagem, aliada à especificação do aplicativo móvel destinado aos clientes da empresa, reflete o resultado positivo e garante a facilidade de uso do sistema, permitindo uma experiência ágil e intuitiva na gestão da compra de produtos para a Xibuco.

Além disso, a especificação dos requisitos e da arquitetura do ambiente de administração do sistema para os colaboradores representa um componente vital para a eficácia operacional e o controle interno. Ao fornecer uma estrutura clara e direta, esse ambiente visa capacitar os colaboradores da empresa, possibilitando uma gestão eficiente e eficaz do sistema de compras.

Dessa forma, a combinação desses elementos: o modelo de persistência, os serviços de backend, o aplicativo móvel e o ambiente de administração, culmina em um ecossistema integrado e dinâmico, alinhado com as necessidades da Xibuco. Este trabalho atende aos objetivos propostos e deixa espaço para a contínua modernização e otimização dos processos de gestão de compras, promovendo a competitividade e a excelência operacional da empresa.

O desenvolvimento deste trabalho revelou-se desafiador, envolvendo uma variedade de temas abordados ao longo do curso. Explorar estes desafios do desenvolvimento de um sistema mobile, desde a concepção até a implementação, proporcionou uma compreensão mais profunda das complexidades envolvidas neste processo. Ao concluir este projeto, obteve-se uma versão inicial do sistema mobile, uma conquista que demandará testes mais aprofundados e refinamentos futuros.

Esta experiência foi uma rica fonte de aprendizado, permitindo a aplicação prática de diversos conceitos estudados ao longo do curso. A construção e percepção das interconexões entre as diferentes áreas da Ciência da Computação foram especialmente destacadas durante o processo. Este trabalho não apenas representou um marco no enfrentamento de desafios, mas também contribuiu significativamente para a consolidação e aplicação dos conhecimentos adquiridos ao longo deste curso.

REFERÊNCIAS

ALECRIM, Emerson. **O que é Tecnologia da Informação?** [*S.l.*: *s.n.*], mar. 2013. `https://www.infowester.com/ti.php`. Acesso em 23/09/2023.

APPLE. **iOS 17: Apple**. [*S.l.*: *s.n.*], set. 2023. `https://www.apple.com/ios/ios-17`. Acesso em 23/09/2023.

BASSETT, Lindsay. **Introduction to JavaScript Object Notation: A To-The-Point Guide to Json**. [*S.l.*]: O'Reilly Media, set. 2015.

CAMARGO, Adriano. **Como será o futuro do mercado mobile?** [*S.l.*: *s.n.*], jan. 2023. `https://olhardigital.com.br/2023/01/10/colunistas/como-sera-o-futuro-do-mercado-mobile`. Acesso em 23/09/2023.

CHUGH, Anupam. **Android MVVM Design Pattern**. [*S.l.*: *s.n.*], ago. 2022. `https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern`. Acesso em 10/11/2023.

FELKE-MORRIS, Terry. **Web Development and Design Foundations with HTML5**. [*S.l.*]: Pearson, fev. 2016.

FOWLER, Martin. **UML Distilled: A Brief Guide to the Standard Object Modeling Language**. [*S.l.*]: Addison-Wesley Professional, set. 2003.

GAMMA, Erich *et al.* **Design Patterns: Elements of Reusable Object-Oriented Software**. [*S.l.*]: Addison-Wesley Professional, out. 1994.

GOOGLE. **Android: Do More With Google on Android Phones and Devices**. [*S.l.*: *s.n.*], set. 2023. `https://www.android.com`. Acesso em 23/09/2023.

HAVERBEKE, Marijn. **Eloquent Javascript, 3rd Edition: A Modern Introduction to Programming**. [*S.l.*]: No Starch Press, dez. 2018.

JACOBSON, Daniel; BRAIL, Greg; WOODS, Dan. **APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces**. [*S.l.*]: O'Reilly Media, jan. 2012.

JEMEROV, Dmitry; ISAKOVA, Svetlana. **Kotlin in Action**. [*S.l.*]: Manning, fev. 2017.

KLEPPMANN, Martin. **Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems**. [*S.l.*]: O'Reilly Media, mai. 2017.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering: Processes and Techniques**. [*S.l.*]: Wiley, set. 1998.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações Móveis. Arquitetura, Projeto e Desenvolvimento**. [*S.l.*]: Pearson, jan. 2005.

MARTIN, Robert C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. [*S.l.*]: Pearson, set. 2017.

MOZILLA. **An overview of HTTP**. [*S.l.*: *s.n.*], mai. 2023. `https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview`. Acesso em 15/09/2023.

PORTOGENTE. **Por que a indústria têxtil é uma das mais poluentes?** [*S.l.*: *s.n.*], fev. 2020. `https://www.portogente.com.br/noticias-corporativas/111101`. Acesso em 08/09/2023.

PRESSMAN, Roger S.; MAXIM, Bruce. **Software Engineering: A Practitioner's Approach**. [*S.l.*]: McGraw Hill, jan. 2014.

RICHARDSON, Leonard; AMUNDSEN, Mike; RUBY, Sam. **RESTful Web APIs: Services for a Changing World**. [*S.l.*]: O'Reilly Media, set. 2013.

ROBBINS, Jennifer. **Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript and Web Graphics**. [*S.l.*]: O'Reilly Media, set. 2012.

RODRIGUES, Iara. **State of Mobile 2023: estudo mostra tendências do mercado de apps e comportamento dos usuários**. [*S.l.*: *s.n.*], jan. 2023. `https://g4educacao.com/portal/state-of-mobile-2023`. Acesso em 23/09/2023.

SEBESTA, Robert W. **Concepts of Programming Languages**. [*S.l.*]: Pearson, fev. 2018.

SEBRAE. **Relatório de Inteligência: Mercado de Segunda Mão**. [*S.l.*: *s.n.*], abr. 2019. `https://atendimento.sebrae-sc.com.br/inteligencia/relatorio-de-inteligencia/mercado-de-segunda-mao`. Acesso em 08/09/2023.

SIVAN, Vishnu. **Introduction to Ktor**. [*S.l.*: *s.n.*], mai. 2023. `https://dev.to/codemaker2015/introduction-to-ktor-571d`. Acesso em 15/09/2023.

SOMMERVILLE, Ian. **Software Engineering**. [*S.l.*]: Pearson, mar. 2015.

STATISTA, Research Department. **Market share of mobile operating systems worldwide 2009-2023**. [*S.l.*: *s.n.*], ago. 2023.

`https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009`. Acesso em 23/09/2023.

W3C. **World Wide Web Consortium: About us, History**. [*S.l.*: *s.n.*], jan. 2023. `https://www.w3.org/about/history`. Acesso em 15/09/2023.

ZOOK, Chris; ALLEN, James. **O crescimento mora ao lado**. [*S.l.*]: Harvard Business Review, dez. 2003.

APÊNDICE A – ARTIGO EM FORMATO DA SBC

Sistema de Gestão de Compras para a Empresa Xibuco

Daniel F. Schröder[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina - (UFSC)

daniel@schroder.cloud

Abstract. *This work aims to develop a purchasing management system for the company Xibuco Comércio Infantil Online, covering new and second-hand clothing, footwear, and accessories. The project established specific goals: creating a data model, implementing backend services, developing a mobile application for customers, and defining the architecture for an internal administrative environment. Upon achieving these objectives, the company can choose to implement the system, improving efficiency in purchasing management for both customers and employees.*

Resumo. *Este trabalho visa desenvolver um sistema de gestão de compras para a empresa Xibuco Comércio Infantil Online, abrangendo roupas, calçados e acessórios novos e seminovos. O projeto estabeleceu metas específicas: criar um modelo de dados, implementar serviços de backend, desenvolver um aplicativo móvel para os clientes e definir a arquitetura para um ambiente administrativo interno. Ao alcançar esses objetivos, a empresa pode optar pela implementação do sistema, melhorando a eficiência na gestão de compras para clientes e colaboradores.*

1. Introdução

A indústria têxtil ocupa o segundo lugar no ranking das atividades que mais prejudicam o meio ambiente global. Isso ocorre principalmente devido à sua grande contribuição para as emissões de dióxido de carbono (CO2), que representam cerca de 10% das emissões totais. Além disso, essa indústria produz uma enorme quantidade de resíduos ao longo de sua cadeia de produção e consome quantidades significativas de água na fabricação de roupas e acessórios. [Portogente 2020]

Neste contexto encontra-se a Xibuco, uma microempresa localizada na região da Grande Florianópolis. A Xibuco atua como um e-commerce varejista especializado em roupas, calçados e acessórios infantis, oferecendo tanto produtos novos quanto seminovos desde setembro de 2018. Seu principal objetivo é reduzir o consumo em massa de roupas infantis, promovendo uma abordagem mais consciente para a compra desses itens. A empresa tem a ambição de se fortalecer no mercado e se tornar um dos maiores brechós infantis do Brasil. Para alcançar esse objetivo, é essencial que a empresa evolua os seus processos internos, reduzindo custos e aumentando a agilidade no tratamento dos seus produtos. Portanto, o foco deste trabalho é criar e implementar uma versão preliminar de um sistema de gestão de compras para a empresa Xibuco.

2. O projeto Xibuco

Os itens comercializados pela empresa chegam até o brechó através das mães que desejam repassar as peças que não servem mais em seus filhos para outras pessoas. O contato inicial destas mães para a venda dos produtos para a Xibuco ocorre através dos canais públicos de comunicação da empresa, como o *WhatsApp* ou o *Instagram*, por exemplo. A partir deste ponto, o processo para a aquisição de um inventário, que é o nome dado ao conjunto de peças que um cliente deseja vender para a Xibuco, segue uma coleção de regras internas para a pré-classificação e avaliação da viabilidade de compra deste inventário por parte da empresa. Por fim, este inventário segue um roteiro de negociação, transporte e curadoria até que seja de fato adquirido, processado e disponibilizado na loja virtual.

Com esta breve descrição, relata-se também um conjunto de dificuldades técnicas e pontos de melhoria a serem adotados pela empresa. Por exemplo: o atendimento não padronizado e altamente dependente do colaborador; a operação de curadoria com custo elevado; o tempo de resposta longo entre as interações com o cliente; entre outros.

3. Análise de requisitos

Para atingir o objetivo deste trabalho, foram identificados 3 atores principais no sistema. São eles: os clientes, consumidores finais da empresa Xibuco e principais usuários do sistema; os funcionários da empresa, que utilizaram o sistema para interagir com os clientes; e por fim, os administradores, que são também funcionários da empresa Xibuco cujo papel de gestor lhes dá direitos administrativos sobre as operações do sistema.

É importante destacar que, dentre todos os casos de uso levantados para a completa implementação da solução, dois fluxos de operação principais se destacam. São eles: o fluxo de operações para o envio de um inventário para avaliação, apresentado na figura 1 abaixo; e o fluxo de operações para a avaliação de um inventário, apresentado na figura 2, também abaixo.

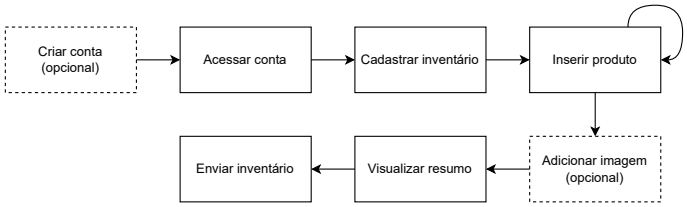


Figure 1. Fluxo de operações para o envio de inventário para avaliação

4. Arquitetura geral

Considerando-se o levantamento de requisitos realizado no escopo deste trabalho e apresentado na seção 3, podemos identificar 3 grandes componentes no sistema de gestão de compras da Xibuco. Estes componentes atuam de forma completamente independente no ponto de vista dos requisitos do sistema, contudo, para a otimização do processo de desenvolvimento e para aumentar a manutenibilidade a médio e longo prazo, é de suma importância ressaltar que eles compartilham o domínio, a lógica e, em virtude da tecnologia adotada neste trabalho, boa parte da implementação e do seu código fonte. Assim, os

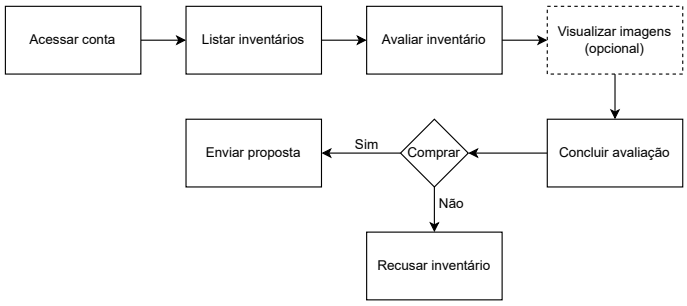


Figure 2. Fluxo de operações para avaliação de inventário

3 componentes identificados são: a camada de serviços e persistência; o aplicativo móvel Android; e o ambiente de administração do sistema.

Cada um destes componentes é apresentado em maiores detalhes nas próximas seções deste trabalho e podem também ser visualizadas de forma ampla no figura 3 abaixo:

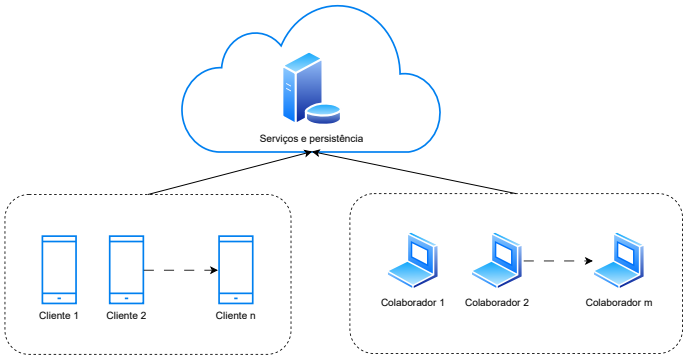


Figure 3. Arquitetura geral dos componentes do sistema

4.1. Serviços e persistência

Estes serviços, também chamados de *backend*, são serviços web que permitem a comunicação e a interação entre os demais componentes do sistema através da internet ou por meio de uma rede local. Além disso, eles são responsáveis por armazenar, tratar e abastecer os demais componentes com informações corretas e confiáveis, mantendo a integridade e a aderência às regras de negócio previamente especificadas na seção 3 e arbitrando em casos de divergência entre quaisquer outros componentes do sistema.

Em virtude da natureza das informações que se necessita armazenar, a primeira decisão de projeto foi a adoção de um banco de dados relacional que atenderá às necessidades deste projeto sem maiores preocupações.

Além disso, outra decisão tecnológica relacionada ao componente dos serviços e da persistência de dados foi a escolha da linguagem de programação a ser utilizada para a implementação dos serviços propriamente ditos. Nesse ponto, a opção adotada foi a linguagem Kotlin em conjunto com o framework de desenvolvimento de sistemas web *Ktor*.

Por fim, vale ressaltar a adoção de bibliotecas menores, utilizadas para garantir a implementação correta e segura de soluções como o acesso ao banco de dados relacional, através do SqlDelight, por exemplo, e a serialização de objetos para transferência via internet, através da própria solução de serialização de objetos da própria linguagem de desenvolvimento Kotlin.

A estrutura geral dos serviços foi dividida em duas grandes camadas: a camada de serviços e a camada de persistência. A camada de serviços foi subdividida em 3 módulos, com uma API de serviços para cada domínio do problema, que são: usuários, inventários e produtos. Por outro lado, a camada de persistência mantém em isolamento o sistema de gerenciamento do banco de dados, cujo modelo de armazenamento de informações está apresentado na figura 4 à seguir.

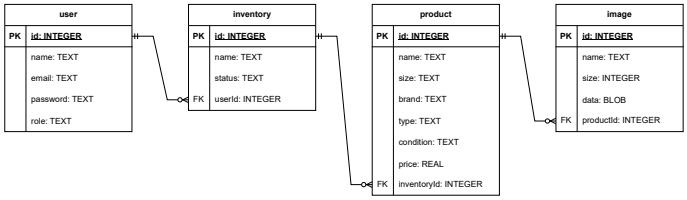


Figure 4. Modelo de dados da camada de persistência

4.2. Aplicativo móvel

O aplicativo móvel é a porta de entrada do usuário mais importante deste sistema, o cliente final da Xibuco. O aplicativo móvel é responsável por apresentar para este cliente as suas opções de forma fácil, clara e rápida, estimulando uma boa interação e garantindo que os processos da empresa sejam realizados de forma correta e segura.

É importante ressaltar que, como a estrutura do sistema como um todo foi implementado fazendo uso do *Kotlin Multiplatform*, boa parte de uma eventual implementação futura para a plataforma iOS, da Apple, já está encaminhada e pode ser reutilizada a partir da implementação existente. Assim, além desta tecnologia, foram adotadas também: o *Ktor*, framework de desenvolvimento de sistemas web e utilizado para auxiliar na implementação de toda a camada de sincronização do aplicativo com os serviços de backend; o *SqlDelight*, utilizada para implementar uma camada de abstração do banco de dados; e o Coroutines, biblioteca do próprio ecossistema Kotlin que oferece suporte à programação assíncrona e concorrente.

A arquitetura escolhida para a implementação do aplicativo Android segue o padrão de desenvolvimento MVVM (*Model View Viewmodel*), adotando também as boas práticas definidas pelo autor Robert C. Martin em seu livro *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. [Martin 2017]

A estrutura geral dos componentes do aplicativo Android foi dividida em 6 grandes componentes: o primeiro sendo a base da aplicação, também chamado de *core*; um segundo chamado de *dependency injection*, responsável por conectar os componentes internos de forma dinâmica; um outro componente responsável pelas definições de interface, chamado de *ui theme* e, por fim, um componente para cada grande funcionalidade do sistema, que são: usuários, inventários e produtos. Mais detalhes dessa estrutura podem ser compreendidos com a Figura 5 a seguir.

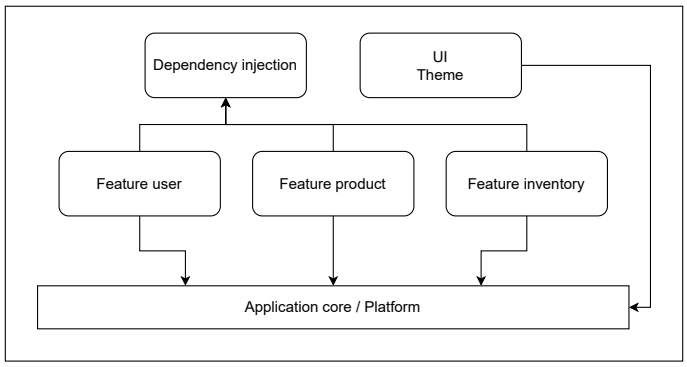


Figure 5. Arquitetura geral dos componentes do aplicativo Android

4.3. Ambiente de administração

O ambiente de administração do sistema, será utilizado pelos colaboradores da Xibuco para realizar todas as ações necessárias para operar o sistema e interagir com os clientes da empresa.

Ao contrário do aplicativo móvel apresentado na seção anterior, o ambiente de administração do sistema não possui necessariamente uma plataforma para ser escolhida, haja vista que ela deverá funcionar a partir de um navegador de internet tradicional.

Portanto, seguindo o uso das tecnologias aplicadas no desenvolvimento do aplicativo Android e dos serviços de *backend*, o ambiente de administração também foi implementado fazendo uso do *Kotlin Multiplatform*, reaproveitando boa parte da implementação já realizada nas etapas anteriores deste trabalho.

Dado o grande reaproveitamento da implementação do aplicativo Android para o ambiente de administração do sistema, através do uso das mesmas tecnologias e do *Kotlin Multiplatform*, pode-se verificar os seguintes fatos: a arquitetura geral dos componentes do aplicativo Android foi completamente reutilizada para a implementação do ambiente de administração; os componentes com as funcionalidades dos usuários, dos inventários e dos produtos, não sofreram alterações diretas nas suas camadas da logica de negócio em código preexistente; e por fim, os casos de uso do administrador, que não estavam presentes no aplicativo Android, foram facilmente incluídos na camada da lógica de negócios da funcionalidade dos usuários.

5. Conclusão

Infelizmente, durante a execução deste trabalho, a empresa Xibuco Comércio Infantil Online encerrou as suas atividades, de tal forma que não foi possível avaliar os resultados completos em seu ambiente esperado. Contudo, pode-se extrapolar os resultados através dos objetivos atingidos e, dada a natureza modesta dos casos de uso especificados, não se fez necessária a criação e a implementação de testes unitários e de integração para a validação do funcionamento do sistema desenvolvido.

Assim, a elaboração de um modelo de persistência robusto oferece a base sólida necessária para assegurar a integridade e a acessibilidade dos dados, contribuindo significativamente para a eficiência operacional do sistema. A criação e implementação dos serviços de backend constituem um passo fundamental na materialização deste modelo de

persistência, proporcionando uma infraestrutura coesa e eficaz para a interação contínua com os dados do sistema. Essa abordagem, aliada à especificação do aplicativo móvel destinado aos clientes da empresa, reflete o resultado positivo e garante a facilidade de uso do sistema, permitindo uma experiência ágil e intuitiva na gestão da compra de produtos para a Xibuco.

Além disso, a especificação dos requisitos e da arquitetura do ambiente de administração do sistema para os colaboradores representa um componente vital para a eficácia operacional e o controle interno. Ao fornecer uma estrutura clara e direta, esse ambiente visa capacitar os colaboradores da empresa, possibilitando uma gestão eficiente e eficaz do sistema de compras.

Dessa forma, a combinação desses elementos: o modelo de persistência, os serviços de backend, o aplicativo móvel e o ambiente de administração, culmina em um ecossistema integrado e dinâmico, alinhado com as necessidades da Xibuco.

References

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.

Portogente (2020). Por que a indústria têxtil é uma das mais poluentes? `https://www.portogente.com.br/noticias-corporativas/111101`. Acesso em 08/09/2023.