SI: um exemplo de sistema distribuido com padroes de arquitetura atuais e populares

Samuel de O. Silva[1], **Bernardo F. Costa**[1]

[1]Instituto de Matemática e Estatística – Universidade Estadual do Rio de Janeiro (UERJ)
Campus Maracanã – 20.550-900 – Rio de Janeiro – RJ – Brasil

samuelsilvapd61@gmail.com, bernardofcosta@ime.uerj.br

Resumo. *Este artigo apresenta um exemplo de sistema que foi construído utilizando-se de padrões populares de arquitetura de software em nossa atualidade. Nosso objetivo é oferecermos uma breve revisão dos padrões atuais popularmente utilizados em arquitetura de software e ilustrar como os diversos componentes do sistema interagem entre si neste caso. Além disso, compartilhamos os resultados dos testes e análises realizados para avaliar o sistema em questão. Trata-se de uma versão resumida de um Trabalho de Conclusão de Curso em Ciência da Computação.*

1. Introdução

Na atualidade, os sistemas em geral são desenvolvidos em camadas, de forma a se distribuir a responsabilidade e carga de trabalho entre vários entes, tendo cada um deles um conjunto de responsabilidades em particular. Para que este conjunto de entes funcione coletivamente de forma satisfatória, são utilizados padrões de comunicação e formas de organização da divisão de seu trabalho. Um padrão popular nos dias atuais de comunicação entre estas partes é o REST [WEBBER et al. 2010], que nos permite a organização dos sistemas em um conjunto de partes com responsabilidades específicas. Essas partes que se comunicam, por sua vez, também possuem organizações internas, arquiteturas que modelam a forma de realizar suas tarefas.

Este artigo tem como objetivo descrever a arquitetura de um sistema simples que ilustre o funcionamento desta arquitetura assim como de padrões que se tornaram populares entre os desenvolvedores nos últimos anos. Para isto, utilizaremos como tema o desenvolvimento de um software gratuito que ajuda pequenos negócios a gerenciarem seus inventários de forma mais fácil. Sistemas de gerenciamento do inventário de produtos [FERREIRA et al. 2022, Faustino et al. 2020] são fundamentais para o sucesso de qualquer negócio, mesmo aqueles pequenos como comércios individuais.

A seguir, trataremos dos seguintes assuntos na ordem listada. Na seção 2, será detalhada a Revisão Bibliográfica dos temas importantes a este trabalho. Na seção 3, detalharemos o projeto em si desenvolvido. Na seção 4, discorreremos a respeito dos resultados alcançados com este projeto. Por fim, apresentaremos as conclusões deste trabalho

Cadernos do IME - Série Informática
e-ISSN: 2317-2193 (online)
DOI: 10.12957/cadinf.2024.84436

e as referências bibliográficas usadas. Ao final do documento se encontram os apêndices onde estão detalhados alguns assuntos específicos. Este trabalho foi apresentado como TCC (Trabalho de Conclusão de Curso) na UERJ e possui apêndices e anexos que detalham alguns assuntos específicos. Estes serão omitidos deste documento por conta do limite de páginas deste formato. Caso o leitor tenha interesse em conhecê-los, estes poderão ser consultados em sua versão completa [SILVA 2023].

2. Revisão Bibliográfica

A seguir, abordaremos os principais temas da literatura científica relacionados a este trabalho. Primeiro será abordada a arquitetura de sistemas, que estabelece a existência de um ou mais softwares e delega as funções ou responsabilidades de cada um. Em seguida será abordada a arquitetura de software, que modela as diversas partes que compõem o software, delegando os papéis de cada uma. Após isso, serão abordados temas referentes à comunicação entre os sistemas na rede. Depois serão vistos dois exemplos de software de gerenciamento de inventário.

2.1. Arquitetura de sistemas: cliente-servidor e desenvolvimentos front-end e back-end

Segundo Kumar [KUMAR 2019], cliente-servidor é um modelo de arquitetura de sistemas dividido em duas partes, sistema cliente e sistema servidor, que se comunicam numa rede de computadores. A aplicação cliente-servidor é uma forma de compartilhar a carga de trabalho entre os sistemas. Em um modelo de arquitetura Cliente-Servidor, um dispositivo denominado cliente se conecta periodicamente a outro dispositivo denominado servidor para conseguir informações ou utilizar serviços. O cliente é responsável por disponibilizar interfaces e possibilitar ao usuário formas de interação com o sistema. O servidor é responsável por armazenar as informações e fornecê-las ao cliente sempre que necessário.

Segundo Kumar [KUMAR 2019] e Oluwatosin [OLUWATOSIN 2014], existem quatro classificações de arquitetura cliente-servidor, que são as arquiteturas: nível um, nível dois, nível três e nível N. Nesta classificação, o tamanho ou ordem do nível determina a quantidade de camadas de dispositivos em que o sistema foi dividido. Na arquitetura nível um só existe um software, que fica num único dispositivo, concentrando as três camadas: apresentação, aplicação e banco de dados. Na arquitetura nível dois, o sistema é dividido em duas partes, uma possui as camadas de apresentação e aplicação, enquanto a outra possui a camada de banco de dados, ambas com apenas um dispositivo cada. Já a arquitetura nível N é uma evolução da arquitetura nível três, com a diferença de que a camada de aplicação não está mais em apenas um dispositivo, mas está dividida em diferentes dispositivos, no objetivo de separar as tarefas realizadas pela camada de aplicação.

É importante entender a arquitetura nível três, pois é a arquitetura utilizada neste projeto. A arquitetura cliente-servidor é classificada como nível três quando ela está dividida em três camadas, sendo elas: camada de apresentação, camada de aplicação e camada de banco de dados. A camada de apresentação é o sistema cliente, que cuida de fornecer tudo o que o usuário é capaz de ver, interagir e participar. A camada de aplicação é a responsável por processar os dados, aplicando as regras de negócio, ou seja, fazer validações

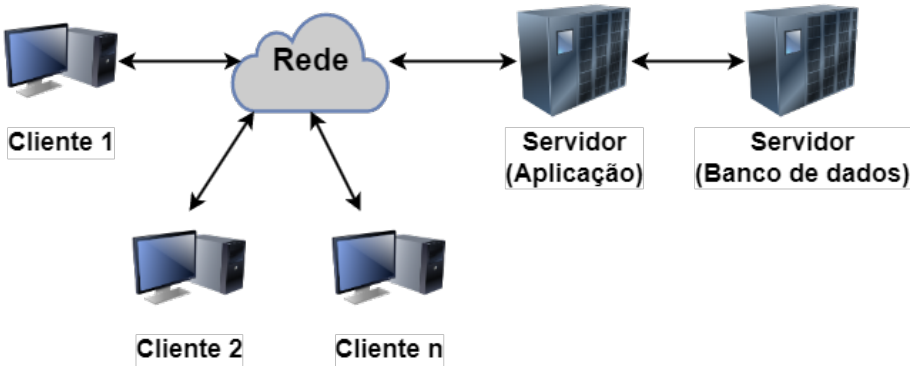


Figura 1. Arquitetura nível três

específicas sobre os dados recebidos, adicionar dados que necessitam ser gerados em tempo real e tomar decisões sobre quais ações realizar com base nos dados em mãos. A camada de banco de dados é onde existe um banco de dados, tendo a responsabilidade de armazenar as informações com segurança.

As camadas de aplicação e de banco de dados são consideradas como sistema servidor, pois elas juntas cumprem o papel de armazenar, gerenciar e fornecer os dados ao sistema cliente. Na arquitetura nível três, a camada de apresentação é composta por vários dispositivos, enquanto a camada de aplicação e de banco de dados são ambas compostas por um dispositivo cada. As camadas de aplicação e banco de dados podem estar localizadas no mesmo dispositivo, caso seja desejado, mas é importante entender que cada camada é um software diferente, ou seja, o computador possuirá dois softwares instalados, um sendo a camada de aplicação e o outro a de banco de dados. A Figura 1 apresenta um diagrama da arquitetura nível três utilizada neste projeto.

Segundo Da Rocha *et al.* [da Rocha et al. 2019] front-end e back-end são duas áreas distintas no processo de criação de um software ou site. O front-end é responsável pela parte visual e interativa que os usuários podem ver e utilizar. Por outro lado, o back-end é responsável por toda a lógica e processamento dos dados. Dessa forma, comumente é dito que o front-end é o sistema cliente e o back-end é o sistema servidor. Essa definição cabe bem ao projeto que será apresentado neste documento.

2.2. Arquitetura de software: arquitetura MVVM (cliente), Arquitetura em Camadas (servidor) e Banco de dados

Lou [LOU 2016] e Batista e Bastos [BATISTA 2017] abordam o conceito de arquitetura de software, que se refere ao código de cada software do sistema. Ela existe para organizar o código-fonte, no objetivo de separar as responsabilidades internas. A arquitetura de software é a estrutura que define como um programa é organizado e como seus diferentes componentes se comunicam. Ela descreve as decisões de design do código e como as partes do programa trabalham juntas para atender às necessidades do sistema. As arquiteturas aqui abordadas serão a arquitetura MVVM e a Arquitetura em Camadas, utilizadas no sistema cliente e no sistema servidor, respectivamente.

2.2.1. Arquitetura MVVM

MVVM é o acrônimo de Model-View-ViewModel, que são três partes constituintes do sistema front-end. O site DhiWise [DhiWise PVT. LTD 2023], plataforma especializada em desenvolvimento de software, explica que essa é uma arquitetura moderna e recomendada pela Google. Ela surgiu com o objetivo de substituir a clássica e antiga arquitetura MVC [LOU 2016] ainda muito usada, que significa Model-View-Controler, ou Modelo-Visão-Controle. A arquitetura MVVM é comumente utilizada no desenvolvimento de aplicativos para dispositivos Android. Lou [LOU 2016] explica que a Google lançou o projeto chamado Android Architecture Blue Print, um projeto para demonstrar novas arquiteturas. Nesse projeto foram desenvolvidos dois programas de exemplo, utilizando duas novas arquiteturas, sendo uma delas a MVVM, com intenção de substituir a MVC.

Como mencionado anteriormente, a arquitetura é dividida em três componentes, ou camadas: Visão, VisãoModelo e Modelo. Uma camada é uma parte do programa que é responsável por realizar suas tarefas específicas. A camada Visão é responsável por exibir a interface do usuário da aplicação. Modelo representa os dados em formatos especiais chamados de objetos. Além disso, também pode fazer acesso a banco de dados ou se comunicar com outros sistemas, para armazenar e buscar os dados ou solicitar serviços. Segundo Lou [LOU 2016], a camada VisãoModelo tem o papel de gerenciar o estado do componente Visão. Ela recebe dados da Visão, aplica regras de negócio referentes à interação do usuário com o sistema, envia ou busca dados do Modelo e ativa gatilhos da Visão que fazem com que a interface do usuário se altere, exibindo na tela o que for necessário. A Figura 2 apresenta um diagrama da arquitetura MVVM.

Figura 2. Arquitetura MVVM

A seguir é apresentada a arquitetura utilizada pelo servidor.

2.2.2. Arquitetura em Camadas

A Arquitetura em Camadas é uma arquitetura popular no desenvolvimento de sistemas servidores na linguagem Java, especialmente quando é utilizada uma ferramenta em conjunto chamada Spring Boot [Broadcom Inc. 2023], por isso, alguns chamam essa arquitetura de Arquitetura Spring Boot. Batista e Bastos [BATISTA 2017] dizem que essa ferramenta, em resumo, é um apoio aos desenvolvedores de software, facilitando o uso de padrões e protocolos que inclusive serão abordados nos próximos capítulos, fazendo com que o trabalho de construir o código seja extremamente mais fácil e rápido.

A Arquitetura em Camadas é dividida em três componentes ou camadas. São elas: Controle (Apresentação ou Recurso), Serviço e Repositório, ou no original em inglês, Controler (Presentation ou Resource), Service e Repository. Segundo Batista e Bastos [BATISTA 2017] a Arquitetura em Camadas organiza as funcionalidades do sistema em

camadas, de forma que cada camada só depende dos recursos disponíveis na camada imediatamente inferior a ela.

A camada Controle é a responsável por receber as solicitações, chamadas comumente pelos programadores de requisições, de serviços de outros sistemas e verificar se são válidas e encaminhar os dados extraídos das solicitações para a camada Serviço. Além disso, é ela que devolve as informações processadas ao sistema que fez a solicitação. A camada Serviço é responsável por fazer validações mais complexas sobre os dados recebidos do Controle, aplicar as regras de negócio do servidor, tomando decisões sobre armazenar, modificar ou buscar dados, retornando em seguida ao Controle os dados processados. A camada Repositório obedece a tomada de decisão do Serviço e é responsável por fazer o acesso direto ao banco de dados, adicionando, buscando ou alterando as informações armazenadas no banco de dados. A Figura 3 apresenta um diagrama da Arquitetura em Camadas.

Figura 3. Arquitetura em Camadas

2.2.3. Banco de dados

Segundo Date [DATE 2004] um banco de dados é um sistema de manutenção de registros, podendo ser considerado como um armário de arquivamento eletrônico, ou seja, o banco de dados é um recipiente para dados computadorizados. Existem duas classificações que são as mais comuns e amplamente utilizadas. Uma delas é o banco de dados SQL, abreviação de Structured Query Language, ou em português, Linguagem de Consulta Estruturada. De acordo com Martins Filho [MARTINS FILHO 2015], em um banco de dados SQL os dados são estruturados e percebidos pelo usuário em forma de tabela. O outro tipo é o NoSQL, abreviação de Not Only SQL, ou em português, Não Somente SQL. Martins Filho [MARTINS FILHO 2015] explica que NoSQL, não é uma única forma de banco de dados, mas sim uma grande variedade de formas e funcionalidades. A diferença mais visível entre os tipos de banco SQL e NoSQL é na estrutura de organização dos dados. Martins Filho [MARTINS FILHO 2015] diz que enquanto o banco relacional organiza os dados em tabelas, o NoSQL é como são classificados os outros tipos de banco de dados que utilizam outras estruturas, como chave-valor, documentos, grafos e família de colunas, por exemplo. Neste projeto, optamos por utilizar um banco de dados SQL, por vezes também chamado de banco de dados relacional.

Para representar os dados em um banco de dados SQL, os atributos de uma tabela seguem formatos específicos que existem para garantir a integridade e consistência do banco de dados. Beaulieu [BEAULIEU 2019] explica que os tipos de atributos mais comuns são em formato de caractere, texto, data ou data e hora, número inteiro ou real e verdadeiro ou falso, mas existem ainda outros tipos de dados específicos.

2.3. Padrões de comunicação pela rede

Nesta subseção, abordaremos como é feita a comunicação entre os sistemas cliente e servidor. Para que seja possível a comunicação, é necessário que os sistemas sigam padrões específicos de comunicação. Nesta subseção, cabe detalharmos melhor alguns padrões de comunicação usualmente seguidos: o protocolo HTTP, o formato JSON de dados e a arquitetura REST.

2.3.1. Protocolo HTTP

HTTP[1] é um protocolo de aplicação em requisição-resposta que permite que sistemas clientes e servidores consigam trocar informações pela rede. Esse protocolo esta descrito em detalhes por Fielding *et al* [FIELDING 1999]. O HTTP estabelece que as requisições são feitas por URLs. URL[2] é um endereço eletrônico que permite que um serviço seja encontrado na rede, como os links de sites populares acessados todos os dias. O HTTP define um conjunto de métodos, que informam a ação a ser realizada. Com isso um sistema servidor recebe uma requisição e consegue identificar a tarefa que o cliente quer que ele realize. Existem diversos métodos. A Tabela 1 lista os métodos mais comuns.

Tabela 1. Métodos HTTP comuns

Método	Função	Exemplo
GET	Solicitar dados	Quando acessamos um site digitamos uma URL em um navegador, então é enviada uma requisição GET, o servidor então recebe a requisição e retorna a página da web do site, que é exibida na tela.
POST	Enviar dados	Quando o usuário preenche um formulário e envia ao servidor, que fará algum processamento ou armazenará os dados.
PUT	Atualizar completamente os dados	Quando o usuário preenche um formulário com alguns dados, fazendo com que os dados restantes sejam apagados.
PATCH	Atualizar parcialmente os dados	Quando o usuário preenche um formulário só com os dados que deseja alterar, mantendo os outros inalterados.
DELETE	Remover dados	Quando o usuário deseja excluir seu cadastro.

Por fim, o protocolo HTTP também estabelece códigos de retorno, chamados popularmente de status. Esses status indicam o resultado da solicitação feita pelo sistema cliente. Esses status são devolvidos do servidor e recebidos pelo cliente, e dessa forma, o cliente consegue lidar com o resultado da requisição.

[1] HyperText Transfer Protocol em inglês ou Protocolo de Transferência em Hipertexto.
[2] Uniform Resource Locator em inglês ou Localizador Uniforme de Recursos.

2.3.2. Formato JSON de dados

Bray [BRAY 2014] diz que JSON[3] é um formato amplamente utilizado para troca de dados na rede. É um derivado da sintaxe da linguagem JavaScript. É uma estrutura simples e legível que permite representar informações de maneira organizada e eficiente. JSON é o formato que a arquitetura REST utiliza para troca de informações. Smith [SMITH 2015] explica que a estrutura do JSON é baseada no modelo de pares de chave-valor. Cada par tem uma chave, que é um nome representando o atributo, e o valor do atributo. Esses pares são organizados por uma sintaxe específica, formando uma estrutura fácil de entender e manipular.

2.3.3. Arquitetura REST

Webber [WEBBER et al. 2010] diz que REST[4] é um estilo arquitetural popular para projetar e implementar serviços web, ou seja, serviços que fazem comunicação pela rede. A arquitetura REST estabelece alguns princípios a serem seguidos, para tornar os serviços escaláveis, flexíveis, de fácil manutenção e altamente interoperáveis. A comunicação deve ser feita através de URLs. As operações são realizadas através dos métodos HTTP que foram listados na Tabela 1. As requisições devem ser ausentes de estado, ou seja, uma requisição deve conter todas as informações necessárias para seja possível compreender e processá-la, não dependendo do resultado de alguma requisição anterior. Os dados são representados em um formato específico, geralmente JSON ou XML[5]. De forma breve, XML, proposto por Rose [Rose 1999], é outro formato de representação de dados, muito utilizado no desenvolvimento de interfaces, mas que também serve para transferência de dados. Por fim, o REST define que será seguida a arquitetura cliente-servidor. Se um sistema segue o estilo arquitetural REST, ele é chamado de RESTful. Nesse caso, portanto, é correto dizer que o sistema servidor do projeto que irá ser demonstrado é uma API web RESTful.

2.3.4. Sistemas de gerenciamento de inventário

Por fim, antes de abordar o projeto realizado, é interessante ilustrar alguns sistemas existentes de gerenciamento de inventário e quais funcionalidades eles oferecem, contemplando que tarefas o usuário pode realizar com esses sistemas. Listamos a seguir dois exemplos de ferramentas conhecidas como referência de software utilizado neste setor.

Sortly® O Sortly® [Sortly Inc. 2023] é um software pago de gerenciamento de inventário, amplamente utilizado por mais de 10 mil empresas. Ele é uma ferramenta poderosa, disponível tanto em versão para computador quanto para dispositivos móveis. Uma das principais funcionalidades é o cadastro de produtos, que permite adicionar imagens e diversas informações relevantes sobre cada item. Além disso, o software oferece a opção de consultar e editar essas informações conforme necessário. O software permite o

[3] JavaScript Object Notation em inglês ou Notação de Objeto em JavaScript
[4] Representational State Transfer em inglês ou Transferência de Estado Representacional
[5] Extensible Markup Language em inglês ou Linguagem Extensível de Marcação

cadastro de diferentes inventários, possibilitando a organização dos produtos de forma individual em cada um deles. Isso facilita a consulta de produtos em inventários específicos. Para utilizar o Sortly® [Sortly Inc. 2023], é necessário fazer login. Os usuários podem escolher entre um cadastro local no software ou optar por realizar o login através de suas contas Google ou Apple, tornando o acesso mais prático e seguro.

Como pontos positivos, podem ser citados: possui versão para desktop e dispositivos móveis, disponibiliza personalização de filtros para as consultas, possui leitura de QR Code, disponibiliza configuração de alertas personalizados, exige login para aumento de segurança e disponibiliza formas de auxílio na gestão das vendas, como geração de relatórios e exportação em arquivos. Como pontos negativos, podemos citar o custo financeiro do software. A título de exemplo, em um plano de assinatura, para se poder ter até 5 logins usados por funcionários, o custo varia entre $29,00 USD a $59,00 USD mensais, já para possuir mais do que 5 logins é necessário contatar a empresa do software e negociar o valor, que será maior que os mencionados.

Contagem Estoque - Contestoque® Contagem Estoque - Contestoque® [ELV Dev 2023] é uma ferramenta simples, que oferece as condições mínimas para o gerenciamento das informações de inventário. É um software gratuito para dispositivos Android, disponível na Google Play Store. Com esse software, é possível cadastrar, consultar e editar informações sobre os produtos de forma prática e rápida. Além disso, o aplicativo também suporta a leitura de códigos de barras, o que torna o cadastro e a consulta dos produtos ainda mais ágil e conveniente. Para garantir a segurança dos dados e facilitar o processo de cadastro, o Contagem de Estoque possibilita a exportação e importação dos dados armazenados em arquivos de texto, permitindo que o usuário faça backups dos registros.

Como pontos positivos, podem ser citados: o software é gratuito, possui opções simples de filtragem de consulta, possui leitura de código de barras e possui uma forma de backup dos dados. Como pontos negativos, podem ser citados: tem poucas informações a serem cadastradas sobre os produtos, não exige login, os dados serem salvos no próprio dispositivo não é seguro, além disso faz com que não seja possível o acesso compartilhado dos dados por outros dispositivos, caso exista mais de um funcionário.

3. A Ferramenta SI: Samuca Inventários

Agora, entendido todos os conceitos sobre arquitetura e observadas as funcionalidades disponibilizadas por sistemas existentes, iremos aqui detalhar o projeto SI. Trata-se de um sistema simples de gerenciamento de inventário desenvolvido utilizando-se alguns padrões populares em sistemas da atualidade como Arquitetura em Camadas entre outros.

3.1. Definições iniciais

Listaremos a seguir um conjunto de sete funcionalidades para este sistema. Iremos classificá-los como itens necessários ou desejáveis. A diferença entre ambos é que estes últimos não são estritamente necessários para ilustrarmos os padrões de desenvolvimento a que o sistema se propõe a ilustrar numa primeira versão do mesmo, diferentemente dos primeiros. As funcionalidades e requisitos classificados como necessários são:

1. Possibilidade de cadastrar, consultar, alterar e remover informações de forma rápida e precisa;
2. O acesso deve ser feito de forma remota pela rede local;
3. O acesso deve ser possível por mais de uma pessoa ao mesmo tempo.

As funcionalidades ou requisitos classificados como desejáveis são:

1. O acesso deve ser possível pela Internet;
2. O acesso deve ser permitido apenas após realização de login do usuário;
3. O sistema deve gerar algum tipo de relatório, podendo ser exportado como arquivo;
4. O sistema deve registrar as ações realizadas pelos usuários e possibilitar alguma forma de consulta a esses registros.

Para atender esses requisitos, foi desenvolvido um sistema que se divide entre cliente e servidor, utilizando a arquitetura cliente-servidor nível três apresentada na Revisão Bibliográfica. O cliente utilizado para esta prova de conceito é um aplicativo para dispositivos móveis feito para o sistema Android. Esse é o sistema pelo qual o usuário pode interagir, visualizar os produtos e realizar suas tarefas. O servidor é um sistema feito para ser instalado em um computador desktop.

Para a instalação do aplicativo, é necessário que a versão mínima do sistema Android seja a 7.0. O aplicativo, após ser instalado, ocupa 18,60 MB de espaço interno do dispositivo móvel. Para a instalação do servidor, é necessário instalar a JDK 17.0.6, o banco de dados MySQL 8.0.34.0, a ferramenta Workbench, o arquivo do servidor e seus executáveis para ligar e desligar o servidor e o sistema para gerar senhas criptografadas dos usuários, juntamente com seu executável. Para a instalação funcionar é recomendado pelo menos 1,5 GB de espaço disponível no computador desktop, mas é fortemente recomendado reservar uma quantidade maior de espaço, para que o banco de dados tenha espaço para armazenar os dados.

3.2. Instalação dos sistemas

Caso seja do seu interesse obter os códigos dos sistemas cliente e servidor, você pode fazer o download deles acessando os links abaixo:

- Cliente – https://github.com/samuelsilvapd61/ProductClient
- Servidor - https://github.com/samuelsilvapd61/ProductServerApi

Caso você deseje fazer o download dos pacotes de instalação dos sistemas cliente e servidor, faça o download das pastas cliente e servidor, disponíveis no link a seguir: https://1drv.ms/f/s!AvNtyddEP78cgthMbu-Y4na1tEhFtg?e=09q3xk. Para a instalação, é recomendado assistir o vídeo onde é demonstrada a instalação completa de ambos os sistemas, além de uma breve demonstração de uso do cliente ao fim. O vídeo se encontra no link a seguir: https://youtu.be/Lz0X02Az3FU. Os links para download dos pacotes de instalação dos sistemas e do vídeo demonstrativo de instalação também se encontram na documentação dos códigos dos sistemas.

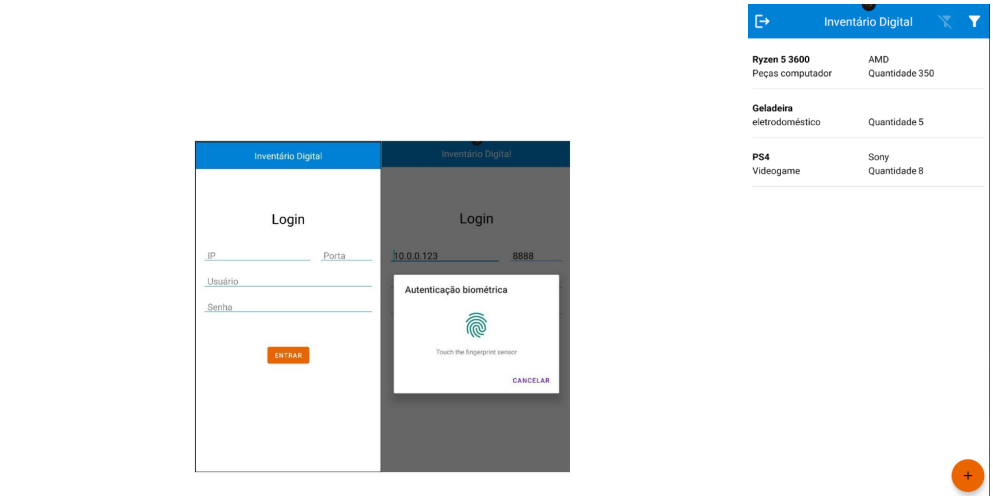
3.3. Implementação dos sistemas

Nesta seção, serão abordados os detalhes da implementação do cliente, servidor e banco de dados.

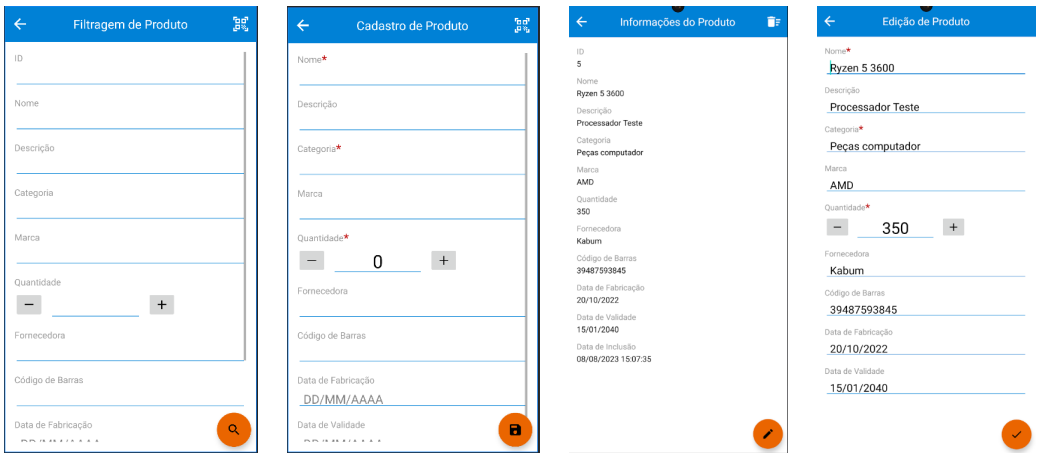
3.3.1. Implementação do cliente

O sistema cliente foi feito utilizando a linguagem Kotlin e seguindo a arquitetura MVVM, ele possui algumas interfaces, cada uma com um propósito bem definido.

Tela de login A Tela de login é por onde o usuário faz login, informando usuário e senha. Caso o login aconteça com sucesso, o usuário é levado para a Tela de consulta. O login realizado permanece válido por 24 horas. A Figura 4a apresenta à esquerda a Tela de login solicitando os dados e à direita a tela de login solicitando a digital do usuário.



(a) Login **(b)** Consulta



(c) Filtragem **(d)** Cadastro **(e)** Informações **(f)** Edição

Figura 4. Telas da Interface Gráfica

Tela de consulta A Tela de consulta é a tela onde o usuário visualiza os produtos cadastrados no servidor. No canto superior direito existem dois botões, um deles leva o usuário para a Tela de filtragem de consulta, o outro desfaz a filtragem de consulta configurada. Ao tocar no botão inferior direito, o usuário é levado para a Tela de cadastro de produto. A Figura 4b apresenta a Tela de consulta com três produtos cadastrados.

Tela de filtragem da consulta A Tela de filtragem é onde o usuário pode personalizar a busca feita pela Tela de consulta. O usuário chega nessa tela ao tocar no botão de filtragem que se localiza no canto superior direito da Tela de consulta. Os campos nome, descrição, categoria, marca, fornecedora e código de barras podem ser informados parcialmente, ou seja, não é necessário informar nome completo, por exemplo. Os demais campos precisam ser informados corretamente, com exceção do campo data de inclusão, sendo necessário apenas data, sem hora, minuto e segundo, como é salvo no banco de dados. Ao tocar no botão QR Code no canto superior direito, é possível fazer leitura de código, preenchendo os campos automaticamente. Ao tocar no botão inferior direito, a filtragem é configurada com base nos campos preenchidos e o usuário retorna para a Tela de consulta. A Figura 4c apresenta a tela de filtragem.

Tela de cadastro de produto A Tela de cadastro é onde o usuário adiciona um novo produto. O usuário chega nessa tela ao tocar no botão de cadastro localizado no canto inferior direito da Tela de consulta. Na Tela de cadastro, o usuário pode cadastrar um produto informando os dados nos respectivos campos, podendo informar apenas os campos obrigatórios ou, além deles, os demais campos. Ao tocar no botão QR Code no canto superior direito, é possível fazer leitura de código, preenchendo os campos automaticamente. Ao tocar no botão inferior direito, um novo produto é armazenado no servidor, com os dados informados. A Figura 4d apresenta a tela de cadastro.

Tela de informações do produto A Tela de informações do produto é onde são exibidas todas as informações de um produto específico. O usuário chega nessa tela ao tocar em um dos produtos que estão listados na Tela de consulta. Na Tela de informações do produto, ao tocar no botão de edição, localizado no canto inferior direito, o usuário é levado para a tela de edição de produto. Ao tocar no botão de deleção, localizado no canto superior direito, o usuário pode deletar permanentemente um produto cadastrado, após confirmar que tem certeza de que deseja realizar tal ação. A Figura 4e apresenta a Tela de informações exibindo os dados de um dos produtos cadastrados.

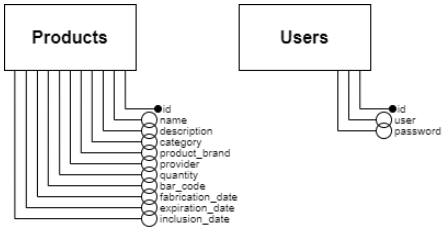
Tela de edição de produto A Tela de edição é onde o usuário pode alterar os dados do produto que estava sendo exibido pela Tela de informações do produto, produto este que já está adicionado no servidor. O usuário chega nessa tela ao tocar no botão de edição localizado no canto inferior direito da Tela de informações do produto. Na tela de Edição, ao tocar no botão inferior direito, o produto em questão tem suas informações alteradas (apenas os campos informados têm seus dados alterados). A Figura 4f apresenta a tela de edição.

3.3.2. Implementação do servidor - aplicação

O sistema servidor foi feito utilizando a linguagem Java e seguindo a Arquitetura em Camadas. O servidor possui 4 rotas de acesso para manipulação de produtos. Cada endpoint é configurado para atender solicitações de uma URL específica, com um método HTTP específico. No projeto em questão, cada endpoint aceita um método HTTP diferente, sendo eles: POST, GET, PATCH e DELETE. Para enviar solicitações de serviço de produtos nesses endpoints, é exigido na URL a adição do caminho "/products". Para a manipulação de produtos, os métodos realizam as ações de adição de produto, busca de produtos, edição de produtos e remoção de produtos, respectivamente. Além disso, existe mais um endpoint com método HTTP POST, que nesse caso não é para adicionar um registro, mas sim para realização de login. Para enviar solicitações de login, é exigido na URL a adição do caminho "/login".

3.3.3. Implementação do servidor - banco de dados

No banco de dados foi estabelecido o modelo de produto de usuário, ou seja, as tabelas que armazenam os dados dos produtos e usuários, configurando os dados, seus tipos, definindo tamanho máximo, chave primária e outras características. Os campos da tabela produtos são: id, nome, descrição, categoria, marca, fornecedora, quantidade, código de barras, data de fabricação, data de validade e data de inclusão. Os campos da tabela usuários são: id, usuário e senha. Os tipos de dados utilizados para o armazenamento são: inteiro (BIGINT), texto (VARCHAR), data sem horário (DATE) e data com horário (TIMESTAMP). Os atributos possuem marcações de chave primária (Primary Key), não nulo (Not Null) e auto incrementável (Auto Increment). A Figura 5a apresenta o Diagrama Entidade Relacionamento do banco de dados. A Figura 5c apresenta a estrutura da tabela Users, para usuários no banco de dados. A Figura 5b apresenta a estrutura da tabela Products, para produtos no banco de dados. A Figura 5d apresenta um exemplo de dois usuários presentes na tabela usuários, sendo possível constatar que a senha está criptografada.



(a) Tabelas

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
name	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
description	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
category	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
product_brand	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
provider	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
quantity	BIGINT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
bar_code	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
fabrication_date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
expiration_date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
inclusion_date	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

(b) Atributos de Products

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
user	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
password	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	

(c) Atributos de Users

id	user	password
1	joao	$2a$10$Fid9fCQsYIM3vBpnyhNmHO.dDXYMrvb41YlcYNQByJ.KUUkc43/My
2	convidado	$2a$10$8x2EOYB48mzKExl6Cas3Iun/qfVDRn86UrnvO449OVlFPGWbVCm8W

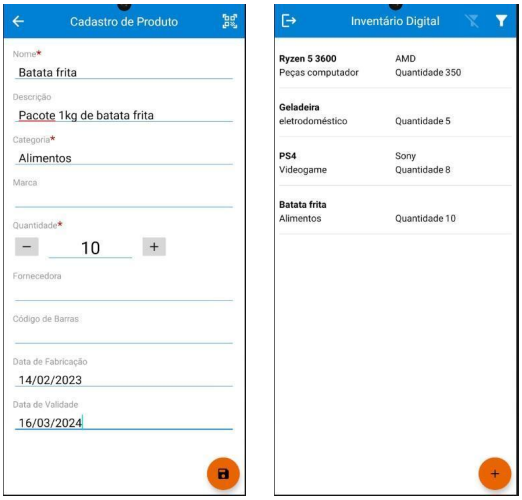
(d) Senha criptografada

Figura 5. Modelagem da Base de Dados

3.3.4. Demonstração do fluxo de cadastro pelas camadas da arquitetura dos sistemas

A título de ilustração do fluxo de dados e ações pelas camadas da arquitetura cliente-servidor, vamos seguir um exemplo em que o usuário faz cadastro de um produto novo. O usuário deseja cadastrar informações sobre um pacote de batata frita, com dados sobre nome, descrição, categoria, quantidade, data de fabricação e data de validade. Será demonstrado o fluxo pelo cliente, em seguida o fluxo pelo servidor, incluindo a camada de aplicação e banco de dados, e por fim o resultado exibido no aplicativo.

Fluxo no sistema cliente O usuário está na tela de cadastro e preenche os campos com as informações que nela constam. A camada Visão providencia a interface para que o usuário possa preencher os campos. A Figura 6a apresenta a tela com os dados informados.



(a) Cadastro **(b)** Consulta

id	name	description	category	product_brand	provider	quantity	bar_code	fabrication_da	expiration_da	inclusion_date
5	Ryzen 5 3600	Processador Teste	Peças computador	AMD	Kabum	350	39487593845	2022-10-20	2040-01-15	2023-08-08 15:07:35
6	Geladeira	NULL	eletrodoméstico	NULL	NULL	5	NULL	NULL	NULL	2023-08-08 15:10:31
7	PS4	Playtation 4	Videogame	Sony	Terabyte	8	NULL	2021-11-15	NULL	2023-08-08 22:37:49
8	Batata frita	Pacote 1kg de batata frita	Alimentos	NULL	NULL	10	NULL	2023-02-14	2024-03-16	2023-08-09 20:24:20

(c) Produto incluído

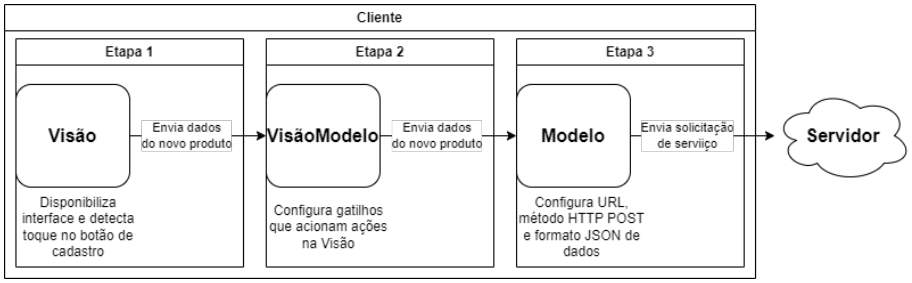
Figura 6. Inserção e consulta na Base de Dados

Após preencher as informações o usuário toca no botão de cadastro. O fluxo entre as camadas da arquitetura do cliente pode ser separado em três etapas:

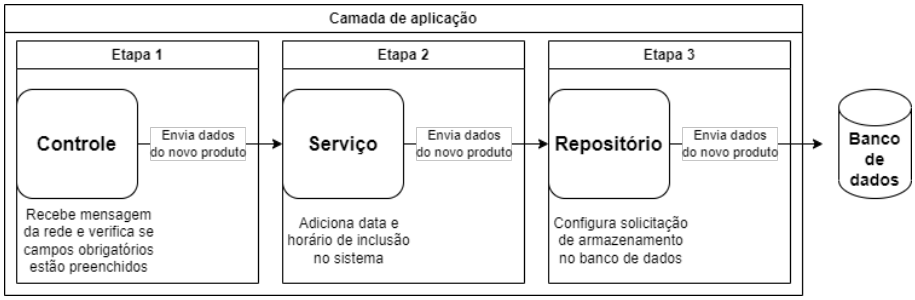
1. A camada Visão detecta o toque no botão de cadastro, então resgata os valores preenchidos nos campos da interface e os envia para a camada VisãoModelo.
2. A camada VisãoModelo, constata que a ação é um cadastro, configura gatilhos que irão acionar ações correspondentes na camada Visão, caso o cadastro resulte em sucesso ou falha, e depois envia os dados do produto para a camada Modelo.

3. A camada Modelo configura a mensagem que será enviada para o servidor. Ela estabelece a URL de destino, o método HTTP POST e que os dados do produto estão representados no formato JSON. Após montada a mensagem, o Modelo a envia pela rede com destino ao servidor.

A Figura 7a apresenta o fluxo entre as camadas do sistema cliente.



(a) Cliente



(b) Aplicação

Figura 7. Fluxo entre camadas

Fluxo no sistema servidor: camada de aplicação Continuando o exemplo do fluxo de cadastro, o sistema servidor aguarda mensagens vindas da rede para realizar ações. O fluxo entre as camadas da arquitetura do servidor pode ser separado em três etapas listadas a seguir:

1. A camada Controle estabelece que serão recebidas chamadas referentes a manipulação de produtos, configurando a URL pelo qual receberá as chamadas. Dentre os endpoints do servidor, um deles está configurado para receber requisições com método HTTP POST, que é por onde a requisição enviada pelo cliente chega, dando início ao fluxo do servidor. O Controle então recebe os dados do sistema cliente, entende que é uma solicitação de inclusão de um novo produto, constata que os atributos obrigatórios estão preenchidos e envia os dados do produto para a camada Serviço. Por fim, nesse endpoint está estabelecido que, caso o cadastro do produto resulte em sucesso, o servidor responderá de volta ao cliente o status 201 - Criado.

2. A camada Serviço inclui no conjunto de dados a data e horário de inclusão do produto no sistema. Após isso ela envia os dados para o Repositório.
3. A camada Repositório monta a mensagem de armazenamento dos dados e os envia para o banco de dados, informando que é uma ação de adição de novo registro na tabela de produtos.

A Figura 7b apresenta o fluxo entre as camadas da aplicação.

Fluxo no sistema servidor: camada de banco de dados Seguindo o fluxo do exemplo de cadastro, ao receber da aplicação os dados do novo produto, o banco de dados os adiciona na tabela de produtos. Finalmente, o fluxo de ida terminou e é possível agora dizer que os dados foram armazenados no servidor. A Figura 6c apresenta os produtos com seus dados preenchidos, incluindo o produto que acabou de ser adicionado pelo usuário.

Conclusão do fluxo Após os dados serem armazenados corretamente, a informação de sucesso parte do banco de dados, volta pelas camadas do servidor até a camada Controle, que envia de volta ao cliente o status de sucesso. O cliente ao receber o status de sucesso, tem um gatilho correspondente ativado, que faz a interface retornar para a tela de consulta, exibindo os produtos cadastrados, sendo possível ser constatado pelo usuário, que o produto foi cadastrado corretamente. Além disso, também é exibida uma mensagem de sucesso de cadastro. A Figura 6b apresenta a tela de consulta do aplicativo com os dados sendo exibidos.

3.3.5. Fluxo de outras funcionalidades

Agora serão explicadas, de forma mais resumida, o fluxo de edição, deleção e consulta.

Fluxo de edição Para a edição de produto, o usuário se encontra na Tela de edição de produto e, após preencher os campos da forma que desejar, toca no botão de confirmação (reveja a Figura 4f). O fluxo no sistema cliente é extremamente parecido, a diferença principal acontece na montagem da mensagem que é enviada para o servidor. Nesse caso é enviada uma mensagem com a mesma URL "/products", mas usando o método HTTP PATCH.

O servidor recebe a mensagem no endpoint específico para método HTTP PATCH. A camada Controle envia os dados do produto para a camada Serviço. A camada Serviço verifica se o produto em questão existe no banco de dados. Se não existe, gera uma mensagem de erro que é enviada pelo Controle de volta ao cliente. Se existir, a camada Serviço faz o trabalho de moldar um novo produto, unindo os dados informados pelo usuário na edição, com os dados armazenados no banco que não serão modificados, dessa forma sobrescrevendo apenas os dados que o usuário informou. Após isso, os dados são enviados para a camada Repositório, que monta uma mensagem de edição de linha e a envia para o banco de dados. O banco de dados recebe a mensagem e sobrescreve

completamente a linha onde estava armazenado o produto, com os dados recebidos. Ele então, retorna à aplicação o sucesso da ação. Assim como no cadastro, a confirmação retorna por cada uma das camadas do servidor e cliente, até que o usuário é levado para a Tela de consulta, onde vê uma mensagem de sucesso de edição e pode constatar que as informações do produto foram modificadas.

Fluxo de remoção Na remoção, o usuário encontra-se na Tela de informações do produto e toca no botão de remoção (reveja a Figura 4e). O fluxo no cliente é muito parecido com cadastro e edição. A diferença acontece na montagem da mensagem que é enviada ao servidor, agora com método HTTP DELETE e enviando apenas o campo id do produto.

O servidor recebe, no endpoint específico para método HTTP DELETE, o id do produto e o envia para o Serviço. O Serviço verifica, com base no id, se o produto em questão existe. Se não existir, gera uma mensagem de erro que é enviada pelo Controle de volta ao cliente. Se existir, o Serviço envia o id para o Repositório que por sua vez, monta uma mensagem de deleção de produto e a envia para o banco de dados. O banco de dados recebe a mensagem da aplicação e apaga o produto em questão. Após isso retorna uma confirmação de sucesso. O fluxo retorna pelas camadas dos sistemas até que o usuário é levado para a Tela de consulta, onde vê uma mensagem de sucesso de deleção e pode constatar que o produto não está mais listado.

Fluxo de consulta O fluxo de consulta acontece automaticamente, sempre que o usuário abre a Tela de consulta, seja após realizar o login, ou retornar para a consulta, vindo de alguma outra tela (reveja a Figura 4b). O sistema cliente envia ao servidor uma mensagem HTTP GET, solicitando ao servidor que retorne uma lista com os 20 primeiros produtos, ordenado crescentemente por id, de acordo com a filtragem configurada no momento.

O servidor, no endpoint específico para requisições HTTP GET, recebe a solicitação. A camada Controle repassa os dados para o Serviço, que repassa para o Repositório. A camada Repositório monta uma mensagem de busca de dados, moldando a mensagem especificamente para atender à filtragem configurada pelo usuário. A mensagem é enviada ao banco de dados. O banco de dados executa a mensagem e retorna os 20 produtos, ou menos, caso ainda não existam 20 produtos. O fluxo retorna pelas camadas dos sistemas até que a Tela de consulta exibe a listagem com os produtos retornados do servidor.

Esse fluxo acontece repetidamente caso o usuário arraste a tela, descendo até o fim da lista, no objetivo de encontrar um produto que não foi um dos 20 listados. O sistema cliente envia outra solicitação, recebe mais outros 20 produtos e adiciona ao fim da lista atual, fazendo com que ela tenha agora 40 produtos, e assim por diante, conforme o usuário desce indefinidamente na listagem.

4. Resultados

Neste capítulo faremos uma avaliação da implementação alcançada para esta versão do sistema SI (1.0) e mostraremos os resultados alcançados por meio dos seguintes eixos: métricas de testes de carga e avaliação de usuários a respeito da ferramenta. As subseções

seguintes detalham como foram obtidas as avaliações realizadas assim como o resultado obtido.

4.1. Teste de carga

Neste projeto, é desejado que o servidor consiga responder requisições simultâneas de pelo menos 20 usuários, com um tempo médio de resposta de no máximo 1 segundo. A ideia desse teste é simular um momento de pico de uso do sistema de uma pequena empresa, ou seja, uma situação em que todos os funcionários estão utilizando o software em algum momento do dia de trabalho.

Para realização do teste de carga, foi utilizado o software Apache JMeter [Apache Software Foundation 2023]. Para medir o desempenho do sistema, o teste foi configurado para ter diferentes quantidades de usuários, todos fazendo requisições simultâneas, cada um fazendo mil requisições sequenciais. O teste dura o tempo necessário para que todos os usuários executem suas mil requisições cada um. As requisições são feitas via Internet. A funcionalidade escolhida foi a consulta de produtos, ou seja, o método HTTP GET, já que é o mais custoso para o sistema, porque manipula uma quantidade muito maior de informações do que as outras funcionalidades. Para saber se as respostas estão sendo retornadas com sucesso, foi configurado no teste que ele verifique se as respostas estão retornando com o código de resposta 200, que é o código esperado para uma consulta bem-sucedida. A Tabela 2 mostra os resultados obtidos com os testes aplicados.

Tabela 2. Resultados do teste de carga

Usuários	**Requisições por usuário**	**Total de requisições**	**Tempo médio de resposta (ms)**	**Tempo mínimo de resposta (ms)**	**Tempo máximo de resposta (ms)**	**Quantidade de respostas com erro (%)**	**Tempo total de teste (s)**
20	1000	20000	5	< 1	44	0%	6
50	1000	50000	14	< 1	101	0%	15
100	1000	1000	36	< 1	2228	0%	38

Os resultados obtidos mostram que as medidas alcançadas para tempo de resposta do servidor e carga de trabalho neste cenário testado estão condizentes com os requisitos mínimos postos para o sistema. O servidor começa a apresentar gargalos consideráveis a partir de uma quantidade de usuários simultâneos muito maior que a quantidade de usuários esperada para uma pequena empresa.

Vale destacar, porém, que os resultados dependem bastante da potência do computador desktop sendo utilizado para rodar o servidor e banco de dados. Além disso, caso as solicitações estejam sendo feitas via Internet, o tempo de resposta também é afetado pela velocidade de conexão de Internet entre o sistema cliente e o sistema servidor.

4.2. Teste de uso com pessoas

Foram feitos dois testes, um para instalação e outro para uso. O teste de instalação foi realizado com duas pessoas, primeiro com uma pessoa envolvida com TI, mas não sendo da área de desenvolvimento de software, e depois outra não envolvida com TI. Foi disponibilizado para elas um documento que lista o passo a passo de instalação e o vídeo tutorial

de instalação. O primeiro participante disse que a instalação foi fácil e que a finalizou em 15 minutos. O segundo participante disse não ter dificuldades e finalizou a instalação em 40 minutos. O segundo participante disse que os arquivos estavam bem organizados e que o vídeo tutorial foi um bom apoio demonstrativo. Ambos os participantes disseram que, pelo menos a princípio, não acham que existe uma grande necessidade de facilitar mais a instalação de alguma forma.

Para o teste de uso, foi solicitado apenas que os participantes instalassem o aplicativo cliente nos seus dispositivos celulares. Elas utilizaram o aplicativo acessando um único computador onde foi previamente instalado o servidor. Após o teste, foi solicitado que as pessoas respondessem um questionário online, informando suas opiniões sobre o sistema e seu uso. A ideia com o teste de uso é deixar que as pessoas utilizem o sistema e, após isso, compartilhem suas sinceras opiniões.

O teste foi realizado com 20 pessoas. Suas idades são estimadas entre 22 e 50 anos, sendo a maioria mais jovem, entre 22 e 35 anos. A Tabela 3 apresenta a distribuição dos participantes em relação à escolaridade e envolvimento com a área de TI[6], seja pelo ensino ou pelo cargo que exercem, não necessariamente desenvolvimento de software.

Tabela 3. Perfil dos participantes do teste de uso

	Médio completo	**Superior incompleto**	**Superior completo**	**Total pessoas**
Envolvido com TI	0	14	3	17
Não envolvido com TI	2	0	1	3
Total pessoas	2	14	4	20

No questionário foram feitas 15 perguntas, abordando alguns assuntos relacionados ao uso dos sistemas. Dessas perguntas, 13 delas são perguntas objetivas e seus resultados podem ser expressos em gráficos. A Figura 8 exibe o resultado das 13 perguntas objetivas. O detalhamento das perguntas em forma textual assim como as respostas não objetivas estão disponíveis na versão completa [SILVA 2023] deste trabalho.

Pelos gráficos apresentados é possível concluir que a maior parte das pessoas que participaram do teste de uso do sistema, de uma forma geral, enxergaram o sistema de uma forma positiva e gostariam de utilizar o sistema caso estivessem na situação proposta pelo projeto. Também é possível concluir com base nas respostas das questões não objetivas que, apesar das reações positivas, foram feitos diversos apontamentos de melhorias e novas funcionalidades que as pessoas gostariam de encontrar no sistema.

Conclusão

Este projeto teve como objetivo revisar a literatura sobre desenvolvimento de software usando uma arquitetura cliente-servidor junto com protocolos de comunicação atualmente populares como o REST e padrões de arquitetura de software. Foi feita também uma revisão de vários conceitos e tecnologias atualmente populares em desenvolvimento de soft-

[6]Tecnologia da Informação

ware que foram explicadas de maneira mais extensa no corpo deste trabalho. Junto a isto, oferecemos também uma opção de software gratuito para gerenciamento de inventário.

A construção do sistema foi bem-sucedida. Foi possível implementar todas as funcionalidades consideradas necessárias, e dentre as desejáveis foi possível implementar o login e a possibilidade de acesso via Internet. Sobre as outras funcionalidades desejáveis, ou seja, a geração de relatório e o registro de ações dos usuários, essas não foram implementadas por falta de conhecimento e tempo para desenvolvimento. Existem também as melhorias e novas funcionalidades que foram sugeridas pelas pessoas que participaram do teste de uso do sistema, sugestões essas relacionadas a vários aspectos do sistema, que envolvem tanto o sistema cliente quanto o servidor. As sugestões feitas envolvem login, consulta, cadastro, filtragem, organização de informações, fluxo de telas e o caractere "/" que em alguns modelos de celulares não aparece no teclado numérico ao ser acionado pelo aplicativo. Todas as funcionalidades desejadas e melhorias apontadas que forem consideradas relevantes podem ser implementadas futuramente para uma nova versão superior do sistema.

Referências

Apache Software Foundation (2023). Apache jmeter — apache jmter™.

BATISTA, Mateus Alfredo; BASTOS, W. C. (2017). *Sistemas de Informação*. Pedra Branca.

BEAULIEU, A. (2019). *Aprendendo SQL: Dominando os Fundamentos de SQL*. Novatec Editora.

BRAY, T. (2014). Rfc7159: The javascript object notation (json) data interchange format.

Broadcom Inc. (2023). Spring boot.

da Rocha, L. C. B., Calazans, V. C., Tolentino, V. C. C., and Villela, H. F. (2019). Índice de popularidade das linguagens de programação e frameworks front-end e back-end nas fábricas de software da região de belo horizonte. *Computação & Sociedade*, 1(1). Artigos Curso Ciência da Computação.

DATE, C. J. (2004). *Introdução a Sistemas de Bancos de Dados*. Elsevier Brasil.

DhiWise PVT. LTD (2023). Google recommendation for android application development architecture.

ELV Dev (2023). Contagem estoque – contestoque — apps no google play.

Faustino, C. R., Luciano, É. L., Oliveira, J. F. V. d., Alves, L. S. R., Alvareli, L. V. G., and Ribeiro, R. B. (2020). Use of the radio frequency identification system in stock management in the automotive sector. *Research, Society and Development*, 9(7):e102973929.

FERREIRA, C. V. S. d. C., CORRÊA, J. P. B., COELHO, Lucas Martins FERNANDES, L. C. D. C., CONCEIÇÃO, P. d., ALMEIDA, R. R. d., and PEREIRA, V. A. d. S. (2022). Sistema de gerenciamento de materiais.

FIELDING, R. e. a. (1999). *RFC2616: Hypertext Transfer Protocol--HTTP/1.1*. RFC Editor.

KUMAR, S. (2019). A review on client-server based applications and research opportunity. *International Journal of Recent Scientific Research*, 10(7):33857–3386.

LOU, T. e. a. (2016). A comparison of android native app architecture: Mvc, mvp, and mvvm. Master's thesis, Eindhoven University of Technology.

MARTINS FILHO, M. A. P. (2015). Sql x nosql: Análise de desempenho do uso do mongodb em relação ao uso do postgresql.

OLUWATOSIN, H. S. (2014). Client-server model. *IOSR International Journal of Recent Scientific Research*, 16(1):67–71.

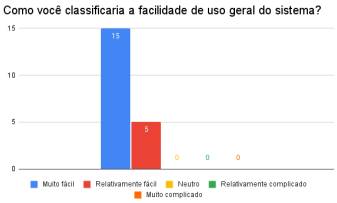
Rose, D. M. T. (1999). Writing I-Ds and RFCs using XML. RFC 2629.

SILVA, S. d. O. (2023). Si: Samuca inventários.

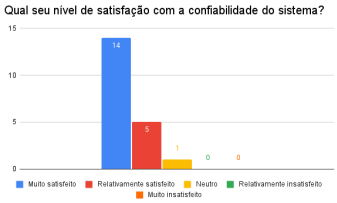
SMITH, B. (2015). *Beginning JSON*. Apress.

Sortly Inc. (2023). Sortly: Inventory simplified.

WEBBER, J., PARASTATIDIS, S., and ROBINSON, I. (2010). *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly Media, Inc.

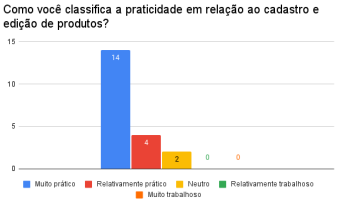


(a) Questão 1

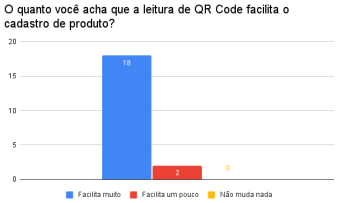


(b) Questão 2

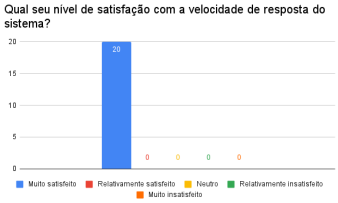
(c) Questão 3



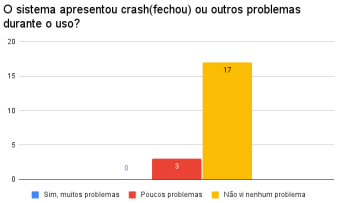
(d) Questão 4



(e) Questão 5

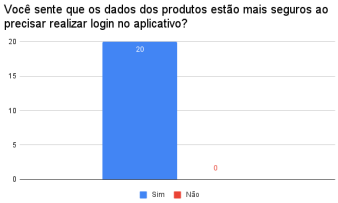


(f) Questão 6

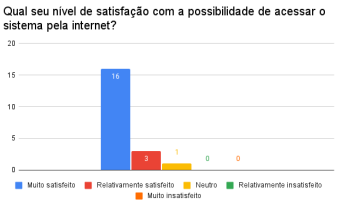


(g) Questão 7

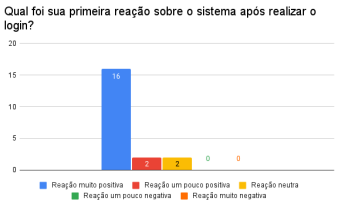
(h) Questão 8



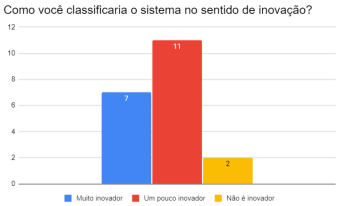
(i) Questão 9



(j) Questão 10



(k) Questão 11



(l) Questão 12

(m) Questão 13

Figura 8. Avaliação do usuário