**INSTITUIÇÃO ANHEMBI MORUMBI DE ENSINO SUPERIOR DO GRUPO ÂNIMA EDUCAÇÃO**

**LUCAS SIMÕES CARVALHO DA SILVA, GIOVANNI LOPES CAMPOS, MARCELO HENRIQUE DA SILVA VENTURA, ANDY HYONG TAE CHOI YOUN, FELIPE DAURA LANZONE FERREIRA**

**TRABALHO DE SISTEMAS DISTRIBUÍDOS E MOBILE: SITE CSGOBRASIL**

SÃO PAULO

2023

|  |  |  |  |
| --- | --- | --- | --- |
| CURSO(S): | ADS, Ciência da Computação | TURMA(S): |  |
| CAMPUS: | Mooca | SEMESTRE(S): | 3º |

|  |  |  |  |
| --- | --- | --- | --- |
| NOME: | Marcelo Henrique da Silva Ventura | RA: | 12522128126 |
| NOME: | Felipe Daura Lanzone Ferreira | RA: | 1252214120 |
| NOME: | Giovanni Lopes Campos | RA: | 12522149685 |
| NOME: | Andy Hyong Tae Choi Youn | RA: | 12522142446 |
| NOME: | Lucas Simões Carvalho da Silva | RA: | 12522161179 |
| NOME: |  | RA: |  |
| NOME: |  | RA: |  |

**RESUMO**

Este trabalho tem como objetivo apresentar um sistema distribuído criado com orientação objeto na linguagem Java. Desta maneira, apresentando suas características, protocolos utilizados e demais tecnologias relacionadas. Além de apresentar a motivação para o tema do sistema. Este projeto utilizará a linguagem Java para o back-end, MySql para banco de dados e CSS, HTML e JavaScript para o front-end.

Palavras-chave: Sistemas distribuídos. Java. JavaScript.

Sumário

[1. INTRODUÇÃO 5](#_Toc137311106)

[2. OBJETIVOS 6](#_Toc137311107)

[2.1 OBJETIVOS GERAIS 6](#_Toc137311108)

[2.2 OBJETIVOS ESPECÍFICOS 6](#_Toc137311109)

[3. JUSTIFICATIVA 6](#_Toc137311110)

[4. IMPLEMENTAÇÃO DO CÓDIGO 8](#_Toc137311111)

[4.1 DIAGRAMA DE CLASSES 8](#_Toc137311112)

[4.2 BANCO DE DADOS 10](#_Toc137311113)

[4.3 SPRING BOOT 11](#_Toc137311114)

[4.4 TECNOLOGIAS E PROTOCOLO 12](#_Toc137311115)

[4.5 DESCRIÇÃO DO FUNCIONAMENTO BÁSICO DO SISTEMA 12](#_Toc137311116)

[5. PRINCIPAIS FUNCIONALIDADES 15](#_Toc137311117)

[6. FRONT-END 15](#_Toc137311118)

[7. CONCLUSÃO 16](#_Toc137311119)

[8. REFERÊNCIAS 17](#_Toc137311120)

# 1. INTRODUÇÃO

Um sistema distribuído é aquele no qual computadores interligados em rede se comunicam e coordenam ações. Neste sentido, o envio de mensagens entre as máquinas é essencial para que a comunicação aconteça. Tal definição leva à algumas características importantes como: concorrência de componentes, falta de um relógio global e falhas de componentes independentes.

Este trabalho tem como ideal apresentar um sistema distribuído que terá as funcionalidades de um site de compra e venda de *skins* (diferentes tipos de padrões de cores para um item em um jogo) para o jogo “Counter-Strike: Global Offensive”. O sistema em questão foi criado a partir da linguagem Java, utilizando-se da tecnologia de *Web service REST*, a qual será explicada no decorrer deste documento, para a comunicação do sistema distribuído.

O documento em questão irá abordar o tema do sistema e sua importância no atual contexto do mundo e como a tecnologia escolhida para a comunicação entre as entidades do sistema distribuído é a mais adequada para a arquitetura do sistema apresentado.

Ademais, serão apresentados a estrutura do código do sistema e as motivações que levaram a escolha do tema do sistema e as tecnologias utilizadas no mesmo. Também será apresentado as principais funcionalidades do sistema e a integração do código com o *front-end* que foi criado como tema principal para a UC (unidade curricular) de Usabilidade, Desenvolvimento Web, Mobile e Jogos.

# 2. OBJETIVOS

## 2.1 OBJETIVOS GERAIS

Este projeto tem como objetivo geral demonstrar de maneira sistemática o funcionamento de um sistema distribuído com no mínimo 5 funcionalidades bem definidas. Desta maneira, serão mostradas imagens, justificativas, explicações e códigos para o sistema desenvolvido durante o andamento da UC de Sistemas distribuídos.

## 2.2 OBJETIVOS ESPECÍFICOS

Dentre os objetivos específicos do trabalho estão:

* Elencar as funcionalidades do sistema;
* Demonstrar a importância do sistema na contemporaneidade;
* Apresentar as tecnologias e protocolos seguidos e utilizados para o desenvolvimento do *software*;
* Explicar a estrutura em código que sustenta a aplicação;
* Ilustrar brevemente o front-end do projeto.

# 3. JUSTIFICATIVA

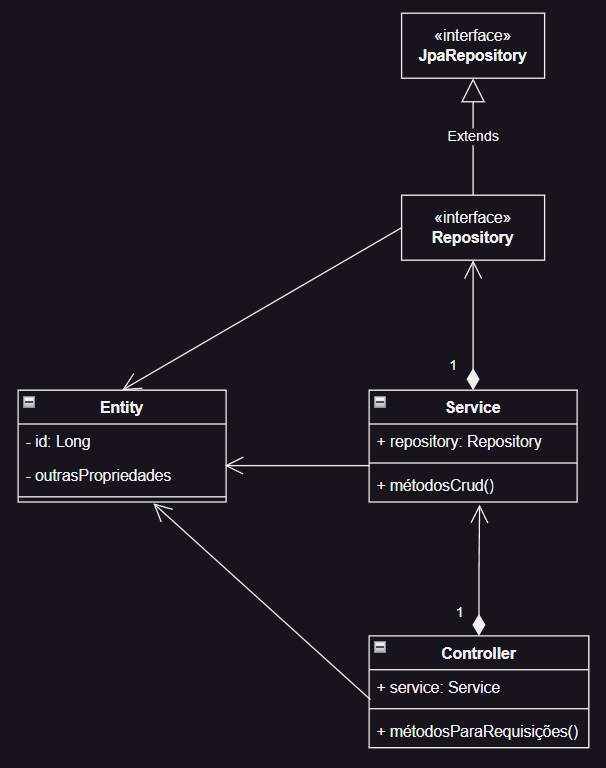
O jogo para computador “Counter-Strike: Global Offensive” é um sucesso global que apresentou este ano um recorde histórico tendo no dia 25 de março um total de 1.507.212 jogadores simultâneos nos seus servidores, segundo a base de dados da empresa Steam. Além disso, o jogo também traz grande rendimento para a empresa desenvolvedora, Valve, que lucra em base 650 milhões de dólares por ano com as caixas que a empresa vende para o público dentro do jogo, segundo a própria empresa.

Com base em tais dados, foi decidido dentro do grupo a criação de um sistema de compra e venda de *skins* do jogo, que movimenta milhões de dólares por ano, segundo a empresa desenvolvedora do jogo.

Dentre a tecnologia utilizada, tem-se a integração REST com *web service*, pois este é o método mais simples e prático de realizar a comunicação entre um *front-end* e um *back-end*. Através de requisições utilizando URLs (Uniform Resource Locator) se torna extremamente simples e rápido de se enviar dados entre as duas partes da aplicação.

# 4. IMPLEMENTAÇÃO DO CÓDIGO

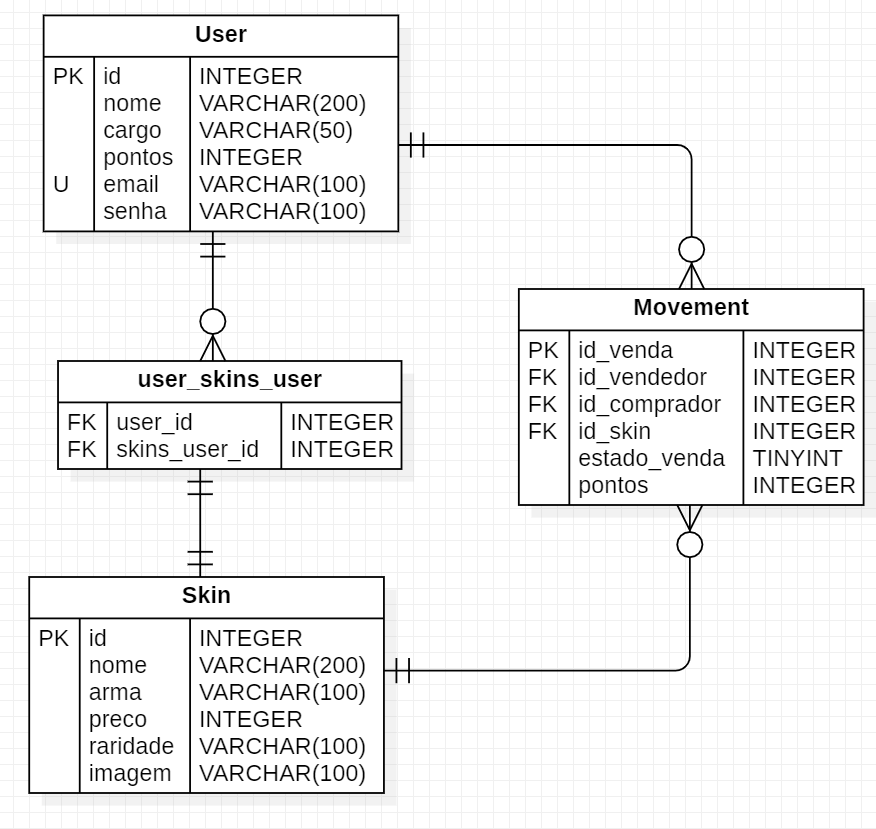
## 4.1 DIAGRAMA DE CLASSES



O diagrama anterior mostra de forma genérica e simples como as classes que representam os itens fundamentais da aplicação se relacionam. Em base, todas as entidades da aplicação funcionam da maneira apresentada, cada uma possui seu repositório que herda métodos funcionais da interface *JpaRepository*. Já a classe *Service* pega os métodos do repositório que ele necessita e então aplica uma lógica de negócio para modificação de dados caso necessário. Então, por fim, a classe *Controller* coloca os métodos da classe de serviço dentro de um método próprio que poderá receber solicitações HTTP (*Hypertext Transfer Protocol*) especificas para aquele controle.

É importante ressaltar, que alguns serviços se utilizam de serviços de outras entidades para retornar os valores necessários para o controle.

## 4.2 BANCO DE DADOS



O banco de dados utilizado para o projeto foi o MySQL e este foi escolhido apenas por maior familiaridade, já que há a possibilidade de ser substituído por qualquer banco de dados relacional.

A estrutura de tabelas para o sistema é dividida em quatro tabelas como é possível ver na imagem anterior. A tabela “User” representa o usuário do sistema que poderá comprar ou vender *skins* e a coluna “cargo” foi criada para distinguir entre um usuário administrador ou um usuário comum. Outra tabela primordial no sistema é a “Skin” que representa a entidade skin que cada usuário poderá possuir várias e tal relação entre usuário e *skin* é representada na tabela intermediária “user\_skins\_user” que possuirá diversos “Ids” de usuários com um determinado Id de *skin*, sendo importante ressaltar que um Id de usuário pode aparecer diversas vezes nesta tabela, porém o Id de uma *skin* só aparecerá uma vez.

A última entidade no banco de dados do sistema é a tabela “Movement”, a qual representa os dados de transações e movimentações de *skins* entre usuários. Ela possui os Ids do comprador e vendedor, da *skin* que está sendo negociada, os pontos pelo qual a *skin* foi anunciada e a coluna “estado\_venda” representa se a *skin* já foi vendida ou não. Ademais, é interessante notar que o tipo da coluna “estado\_venda” é um número inteiro, já que em MySQL uma das melhores formas de representar o tipo *boolean* é por meio de um número, então a representação de verdadeiro ou falso é dada por 1 ou 0, respectivamente.

## 4.3 SPRING BOOT

Neste projeto, o *back-end* foi desenvolvido com o auxílio do *framework* *Spring Boot*. A sua utilização foi devido a facilidade que o framework trás para diversos aspectos, como gerenciamento e injeção de dependências, conectividade com banco de dados, facilidade para a criação de aplicações web, entre outros.

O *Spring Boot Framework* é um dos *frameworks* mais populares do mundo quando se está falando de Java, segundo uma pesquisa feita pela empresa Alura em 2022, e também é um dos mais aprovados e testados pela comunidade o que o tornou altamente confiável e amplamente usado por grandes empresas como Netflix, Amazon e Google.

Neste sistema, o *Spring Boot* foi majoritariamente usado para a conexão com o banco de dados, injeção de dependências utilizando suas anotações, criação de um controle para recebimento de requisições http e gerenciamento de outras dependências.

## 4.4 TECNOLOGIAS E PROTOCOLO

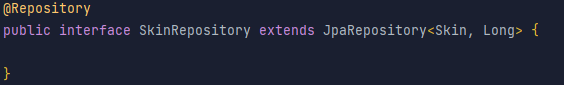
Este projeto conta para o desenvolvimento *back-end* tecnologias como o *Spring Boot Framework* como foi mencionado anteriormente e esta tecnologia tem grande participação no desenvolvimento da arquitetura REST no trabalho. Como uma aplicação *web*, este *software* utiliza de métodos do protocolo HTTP para a operações em recursos, os quais são identificados por URLs.

Desta forma, tem-se a utilização do protocolo HTTP, pois este é o protocolo mais simples, amplo e utilizado de toda a internet para a comunicação cliente-servidor. Através de seus métodos próprios (como GET, PUT, DELETE, POST) e de uma URL para identificar o recurso que sofrerá a ação do método, o HTTP permite uma comunicação simples e rápida entre o cliente e o servidor através de conexões TCP/IP.

Outra capacidade do protocolo que faz com que ele se torne mais atrativo para seu uso em aplicações é a sua capacidade de trocar dados entre o cliente e o servidor em diferentes formatos como JSON, XML, texto, HTML, entre outros.

Portanto, fica claro que a utilização do protocolo HTTP com uma arquitetura REST é a melhor opção para a criação e controle de uma aplicação *web* devido à sua simplicidade, escalabilidade, suporte a praticamente todos os navegadores e adequação ao modelo cliente-servidor.

## 4.5 DESCRIÇÃO DO FUNCIONAMENTO BÁSICO DO SISTEMA



Nesta primeira imagem é apresentado o repositório da classe *Skin*, é perceptível que ela está herdando as propriedades da interface *JpaRepository*, que possui como parâmetros a própria classe *Skin* e o objeto *Long* que representa o tipo do id que a classe *Skin* possui. É nesta classe que outros métodos mais complexos de pesquisas SQL seriam realizados caso necessário.



Agora, é possível ver a classe de serviço da classe *Skin*, nesta classe haveria qualquer lógica de negócio necessária para o tratamento de dados recebidos ou enviados pelo sistema. Ademais, nota-se a presença da anotação *Autowired* para o repositório da classe *Skin*, esta anotação de propriedade do *Spring Boot Framework* gerencia a injeção da dependência do repositório de *skins* neste serviço.

Nesta classe há diferentes métodos que estão recebendo e enviando dados para o repositório de *skins*, o qual se conectará com o banco de dados, assim pode-se dizer, que o *service* é uma classe intermediária entre o repositório e o controle do *software*.



Por fim, tem-se o controlador da classe *Skin*. Este é aquele que recebe solicitações HTTP e retorna uma entidade de resposta com um status HTTP. É notável nesta classe a presença da anotação *Autowired*, novamente, para a injeção da dependência do serviço da aplicação.

Além disso, é importante ressaltar a importância das anotações corretas para os métodos dessa classe, já que é nestas anotações que são definidos o tipo de método HTTP e a URL que ele estará escutando chamadas.

# 5. PRINCIPAIS FUNCIONALIDADES

Este sistema possui diversas funcionalidades, que satisfazem a necessidade de qualquer chamada do seu respectivo *front-end*. Dentre elas há:

* Criação e atualização para todas as principais entidades do sistema;
* Capacidade de leitura de todos os dados do banco de dados, de maneira específica ou genérica;
* Capacidade de receber solicitações HTTP e enviar respostas congruentes;
* Criação de relatórios com informações úteis sobre as movimentações de *skins*;
* Filtro para a procura de dados dos usuários;
* Capacidade de checar e validar senha, nome de login e o email de um usuário.

# 6. FRONT-END

O *front-end* do sistema tem como característica mais importante para o *back-end* o envio de requisições HTTP. Através da *Fetch* API, utilizando apenas Javascript é possível enviar uma solicitação HTTP com corpo e até mesmo parâmetros na URL se necessário.

Através do uso das URLs corretas, foi possível conectar o *back-end* e o *front-end* deste sistema de maneira fácil e rápida, permitindo assim que o *front-end* receba informações do banco de dados sem acessá-lo diretamente. Desta forma, mantendo os dados do usuário em segurança.

# 7. CONCLUSÃO

Em conclusão, este documento explorou os sistemas distribuídos com foco na arquitetura REST e a utilização do protocolo HTTP de um *front-end* em JavaScript, utilizando a arquitetura REST.

Ao longo da dissertação, foi apresentado os principais conceitos e benefícios dos sistemas distribuídos, destacando a capacidade de escalabilidade, disponibilidade e tolerância a falhas que eles proporcionam.

Utilizando a arquitetura REST, foi possível projetar uma API REST para um *site* de *skins* do jogo “Counter-Strike: Global Offensive”, que segue princípios como a utilização adequada dos métodos HTTP, a definição de URIs consistentes e a transferência de dados no formato JSON.

Além disso, é perceptível que o uso do *Spring Boot* facilitou e aprimorou o desenvolvimento do back-end, fornecendo recursos como injeção de dependência, controle de recebimento de requisições HTTP e integração com o banco de dados.

Também é notável que o uso do padrão REST, juntamente do protocolo HTTP permitiu uma comunicação eficiente, clara e consistente.

Em resumo, a junção das tecnologias usadas neste sistema oferece uma base sólida para o desenvolvimento de sistemas distribuídos, principalmente para aqueles voltados para aplicações de *web service*.

# 8. REFERÊNCIAS

<https://www.alura.com.br/artigos/spring-conheca-esse-framework-java>;

<https://www.yworks.com/yed-live/>;

**Sistemas Distribuídos George Coulouris, Jean Dollimore, Tim Kindb;**

<https://blog.counter-strike.net>;

<https://steamdb.info>;

AUTORIA [RAMÍREZ PÉREZ, SANTIAGO](https://ebuah.uah.es/dspace/browse?authority=21b13914-c985-4054-a871-82aa33911003&type=author). Estudio del framework Spring, Spring Boot y microservicios. Disponível em: <https://ebuah.uah.es/dspace/handle/10017/45107>. Acesso em: 01/06/2023;

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP>;

https://www.csgodatabase.com/skins;