IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Andrey Camargo Lacerda	SP3013049
Fabricio Ernesto dos Santos	SP3013171
Guilherme Oliveira de Souza Leão	SP3013243
Luis Antonio Gonçalves Novaes Angelim	SP301309X
Vitor Urdilai	SP3013111

DowJonesSkins

São Paulo - SP - Brasil

22 de Novembro de 2019

IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Andrey Camargo Lacerda	SP3013049
Fabricio Ernesto dos Santos	SP3013171
Guilherme Oliveira de Souza Leão	SP3013243
Luis Antonio Gonçalves Novaes Angelim	SP301309X
Vitor Urdilai	SP3013111

DowJonesSkins

Monografia do Projeto de DW2A4, MTPA4 e ESA4A orientadas pelos professores Luis Fernando Aires Branco Menegueti (DW2A4 e MTPA4) e Daniel Marques Gomes de Morais(ES4A4).

Professor: Luis Fernando Aires Branco Menegueti

Professor: Daniel Marques Gomes de Morais

IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Tecnologia em Análise e Desenvolvimento de Sistemas

DW2A4 - Desenvolvimento Web II

São Paulo - SP - Brasil

22 de Novembro de 2019

Resumo

Este trabalho de pesquisa visa a implementação de uma aplicação web que simule um mercado financeiro para skins presentes no jogo eletrônico "Counter-Strike: Global Offensive", funcionando como uma bolsa de valores e permitindo a modalidade de negócio "Day Trading". Neste sistema, o usuário irá logar via Steam Account e depositará suas skins no site, permitindo assim o investimento de suas skins, assim como a compra de outras skins em investimento. O desenvolvimento e planejamento do projeto beberá dos métodos ágeis, aplicando uma engenharia de software próxima ao que reside na metodologia XP. Para isso a ferramenta Pivotal Tracker será utilizada para administração do projeto, onde todas as Histórias de Usuário serão organizadas e discutidas. Partindo para a parte mais técnica, o sistema será implementado utilizando "HTML5"para o front-end e "Node.Js"para o back-end. No back-end, o framework "express"será responsável por instanciar e administrar o servidores web. Para testes, os frameworks "Jest"e "Cucumber"serão utilizados, enquanto para deploy e controle de versão, o Heroku e o Github serão utilizados, respectivamente.

Palavras-chaves:day trading de skins. bolsa de valores de skins. sistema de troca de skins. aplicação web.

Abstract

This research work aims at the implementation of a web application that simulates a financial market for skins in the electronic game 'Counter-Strike: Global Offensive ', functioning as a stock exchange and allowing the modality of Day Trading business. In this system, the user will login via Steam Account and deposit your skins on the site, allowing you to invest your skins as well as buying other investment skins. The development and planning of the project will drink from the applicable methods by applying software engineering which lies in the XP methodology. For this the Pivotal Tracker tool will be used for project administration, where all the 'User Stories' will be organized and discussed. Starting with the most technical part, the system will be implemented using HTML5 for the front end and Node.Js for the back end. In the backend, the framework Express will be responsible for instantiating and administering the web server. For testing, the 'Jest' and 'Cucumber' frameworks will be used, while deploying and controlling version, Heroku and Github will be used respectively.

Key-words: skins day trading. skins stock exchange. skins exchange system. web application.

Lista de ilustrações

Sumário

1 Introdução

Atualmente, o mercado de jogos eletrônicos movimenta acima de 130 bilhões de dolares ao redor do planeta, como pode ser visto em pesquisas publicadas pela NewZoo, empresa especializada em coleta e estudo de dados do mercado de jogos digitais. Dentro do mercado de jogos, existe um nicho de destaque, que consiste no mercado de Skins, que são equipamentos ou customizações que podem ser compradas com dinheiro físico e utilizadas dentro do jogo especifico que as detêm.

Mesmo sendo um mercado forte, movimentando mais 10 bilhões de dólares por ano no jogo 'Counter-Strike: Global Offensive', como dito na máteria 'Mercado de skins de CS:GO pode movimentar até 10 bilhões de dólares por ano' do portal 'The Enemy', as aplicações criadas para este nicho exploraram poucas vertentes de mercado até então, sendo baseadas em simples sistemas de mercados de Skins, em que um vendedor oferece sua mercadoria para que algum comprador interessado faça negócio, ou sendo baseadas em jogos de azar.

Como este mercado gera muito capital, como dito anteriormente, uma grande quantidade de aplicações, principalmente web, foram implementadas e estão no ar, o que saturou os softwares que seguem as vertentes citadas anteriormente. Com isso, como seria possível criar uma aplicação web para este mercado de skins de forma que tal sistema não caia em saturação?

Com objetivo de resolver este problema, este trabalho de pesquisa consistirá no desenvolvimento de um software web para venda de skins de jogos eletrônicos, neste caso vendo em conta skins de 'Counter-Strike: Global Offensive', que terá como funcionamento um sistema de bolsa de valores e Day Trading, saindo do simples mercado já bem explorado.

1.1 Questão de Pesquisa

O tema deste projeto de pesquisa consiste no desenvolvimento de aplicações web para venda de skins de jogos eletrônicos.

A problemática do projeto de pesquisa reside na implementação de uma aplicação de venda de skins do jogo 'Counter-Strike: Global Offensive' que funcione como um sistema de bolsa de valores e Day Trading, simulando um mercado financeiro de skins, saindo da visão saturada de mercado comum, onde um vendedor simplesmente anuncia sua mercadoria, e um comprar a compra diretamente.

1.2 Objetivos

Em linhas gerais, o objetivo deste projeto é desenvolver uma solução inovadora no mercado de venda de skins de CS:GO que implementará um sistema de bolsa de valores e Day Trading dessas skins, algo não visto no mercado até então.

1.2.1 Objetivos Primários

Os objetivos principais giram em torno de elaborar e desenvolver um novo sistema de venda de skins de jogos eletrônicos, inovando este cenário atual. Para isso, será necessário conhecer as soluções existentes no mercado quando falamos em aplicações de venda de skins de jogos e entender seu funcionamento como um todo, para que identificar os pontos fortes e fracos deste mercado atualmente.

1.2.2 Objetivos Secundários

Como objetivos secundários, o projeto visa promover estudo e capacitação sobre tecnologias e meios utilizados atualmente para a implementação de aplicações web, além de promover um melhor entendimento sobre os ideais de mercado financeiro de ações e o mecanismo de Day Trading.

1.3 Justificativa

As principais razões que se levam à execução deste projeto e, por consequência, ao desenvolvimento de aplicações web para venda de skins de jogos eletrônicos são:

- Inovação do mercado, pois a implementação de um mercado de ações e um Day Trade de skins de CSGO consiste em algo novo para o cenário desenvolvido até então, que permeia sites de mercado comum e aposta;
- Exploração de novas tecnologias que farão parte do desenvolvimento do sistema como Handlebars, por exemplo, possibilitando estudo prático para os membros do projeto;
- Viabilidade comercial, pois se pode lucrar muito com taxas de trocas, como pode ser vista no artigo de pesquisa da NewZoo sobre o lucro do mercado de jogos em 2018, que passou de 130 bilhões de dólares ao redor do planeta, sendo que o mercado de skins de CSGO movimenta por ano mais de 10 bilhões de dólares, segundo a matéria da 'The Enemy'.

2 Modelo Teórico e Pressupostos (ou Hipóteses) da Pesquisa

No mercado em geral, principalmente no Brasil, não existem sites com a proposta da valorização de skins, muito menos aplicações inspiradas no mundo financeiro de bolsa de valores e Day Trading, então o desenvolvimento do sistema proposto neste documento preencherá esta lacuna fazendo possível um usuário além de fazer as suas trocas, ter uma experiência de bolsa de valores com seus itens do jogo.

Tendo em vista que grande parte dos jogadores utilizam suas skins também para uso comercial e conseguir dinheiro, foi levantada a hipótese de que o site de valorização tem grande possibilidade de obter popularidade e o agrado do público do jogo envolvido.

3 Metodologia do Projeto

Buscando a implementação de aplicação web proposta, este projeto utilizará dos métodos ágeis, principalmente seguindo a figura da metodologia XP. Para auxiliar na execução destes métodos, a ferramenta Pivotal Tracker será utilizada para administrar as histórias de usuários e promover um ambiente para organizá-las e discuti-las.

Como parte dos métodos ágeis, o sistema utilizará duas ferramentas de testes, o Jest e o Cucumber, realizando testes unitários, de integração e testes End-to-End, além do fato de que o Cucumber fará os testes direto nos cenários criados pelas histórias de usuário.

Em questão técnica, a aplicação será feita em cima de um back-end em Node.Js, enquanto o front-end utilizará do HTML e do Handlebars. Para controle de versão, o Git será utilizando, portanto o projeto contém um repositório no GitHub onde cada progresso será salvo, controlando o ciclo de vida do software. Para deploy, o Heroku será utilizado. Com isso, uma aplicação foi criada no Heroku, onde o sistema será posto no ar toda vez que uma mudança for 'commitada' no GitHub.

Para estudo de domínio, informações com players ativos de Counter-Strike: Global Offensive serão coletadas, além de informações sobre sites de grande importância no meio de mercado de skins. Como não é uma área documentada, será necessário fazer uma abordagem mais informal e retirar informações direto de clientes e de experiencias próprias dos integrantes deste trabalho.

Pesquisas com alguns jogadores para verificar a viabilidade desta aplicação proposta no projeto serão realizadas.

4 Desenvolvimento do Projeto

4.1 User Stories

Como descrição das necessidades dos usuários desta aplicação, servindo como casos de uso + requisitos funcionais, as seguintes User Storie foram discutidas e, ao longo do projeto, implementadas:

- Eu, enquanto usuário, devo conseguir cadastrar/logar via Steam Account para usufruir das funcionalidades do site.
- Eu, enquanto usuário, devo conseguir depositar minhas Skins de CSGO do inventário da Steam no inventário do Site, a fim de investir nelas no site.
- Eu, enquanto usuário, devo conseguir depositar apenas as Skins permitidas pelo site, já que o site limita as Skins aceitas.
- Eu, enquanto usuário, devo conseguir colocar na bolsa uma Skins do meu inventário do site, para que ela possa valorizar/desvalorizar com o tempo.
- Eu, enquanto usuário, não posso "sacar"as Skins em investimento no site, apenas as que estão no meu inventário.
- Eu, enquanto usuário, só poderei retornar uma Skins do investimento para o inventário após 1 mês de investimento.
- Eu, enquanto usuário, devo conseguir "sacar"as Skins que estão no inventário do site para o meu inventário Steam, para que possa utilizá-las no jogo e na Steam.
- Eu, enquanto usuário, devo conseguir ver meu histórico de vendas/compras no site, visualizando o item comprado/vendido, o preço e a data da compra.
- Eu, enquanto usuário, poderei ver a valorização/desvalorização das Skins investidas no site, para que eu possa saber a hora de comprar e vender.
- Eu, enquanto usuário, devo conseguir comprar Skins investidas no site com dinheiro.

4.2 1° Sprint

O primeiro sprint consistiu na preparação do repositório do projeto, com a instalação dos seguintes FrameWorks: Code Climate, Cucumber, Jest, Travis CI. Além da

configuração dessas ferramentas, foi desenvolvido o primeiro esboço do projeto do projeto para inserção no Heroku e teste da configuração. Neste esboço, foi criado a Home do site e o arquivo node.js principal. Para a parte de ES4A4, foi criado o projeto no PivotalTracker, a fim de inserirmos as User Stories.

4.2.1 Criação do Repositório no Git

O primeiro passo tomado pela equipe foi a criação do repositório no Git. Para isso, o integrantes Guilherme Leão criou o repositório no GitHub, e posteriormente adicionou todos os membros da equipe e o professor de ES4A4, Daniel Morais.

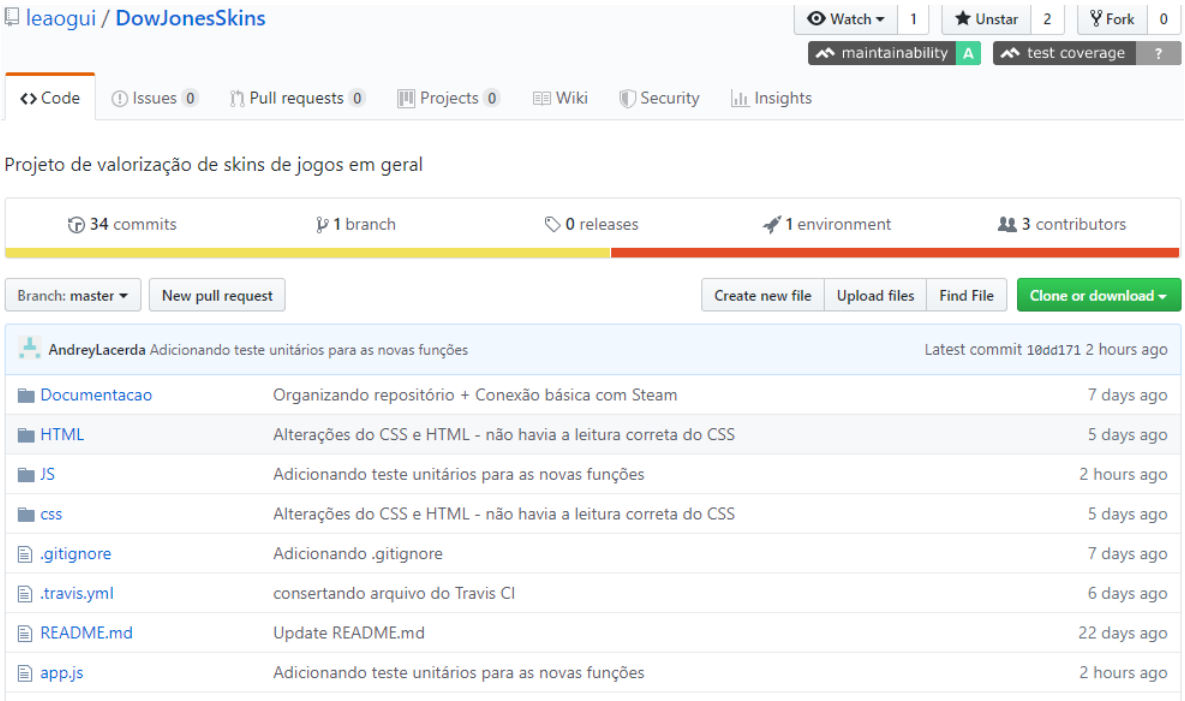

Figura 1 – Repositório do Projeto no GitHub

4.2.2 Instalação dos Frameworks

Dentro da pasta raíz do projeto, o comando 'npm init' foi executado via CMD. Para isso, o Node.js foi instalado nas máquinas dos integrantes do projeto. Este comando criou três arquivos: node_modules, que consiste no diretório onde todos os módulos e bibliotecas utilizadas são salvas; package.json, que consiste em um 'arquivo de configuração', possuindo informações sobre a execução do project; package-lock.json, que consiste em um arquivo que salva informações de todas as biliotecas e dependencias instaladas no projeto.

Com esses três arquivos criados, o Jest foi instalado, a partir do comando 'npm install jest', e o Cucumber, a partir do comando 'npm install cucumber'. Ambos os

frameworks são utilizados para testes.

Após a instalação dos dois frameworks para o node.js, os dois frameworks foram instalados para o próprio repositório do GitHub. O primeiro foi o CodeClimate. Para isso, o dono do repositório instalou a extensão CodeClimate no navegador, e posteriormente entrou no git do projeto. Dentro do GitHub do projeto, uma opção apareceu para adicionar o projeto ao CodeClimate. Após clicar na opção, o repositório foi lido e adicionado ao CodeClimate, que agora é o plugin de avaliação de código do repositório.

Após isso, o dono do repositório instalou o Travis CI pelo próprio 'GitHub Market-Place'.

4.2.3 Criação e configuração do Heroku

Para a criação do app no Heroku, o Heroku CLI foi instalado. Com ele instalado, o comando 'heroku create' foi executado via CMD, criando o app no Heroku. Após criar o Heroku App, o deploy do Heroku foi configurado para que ele seja sincronizado com o 'push' para o repositório no GitHub.

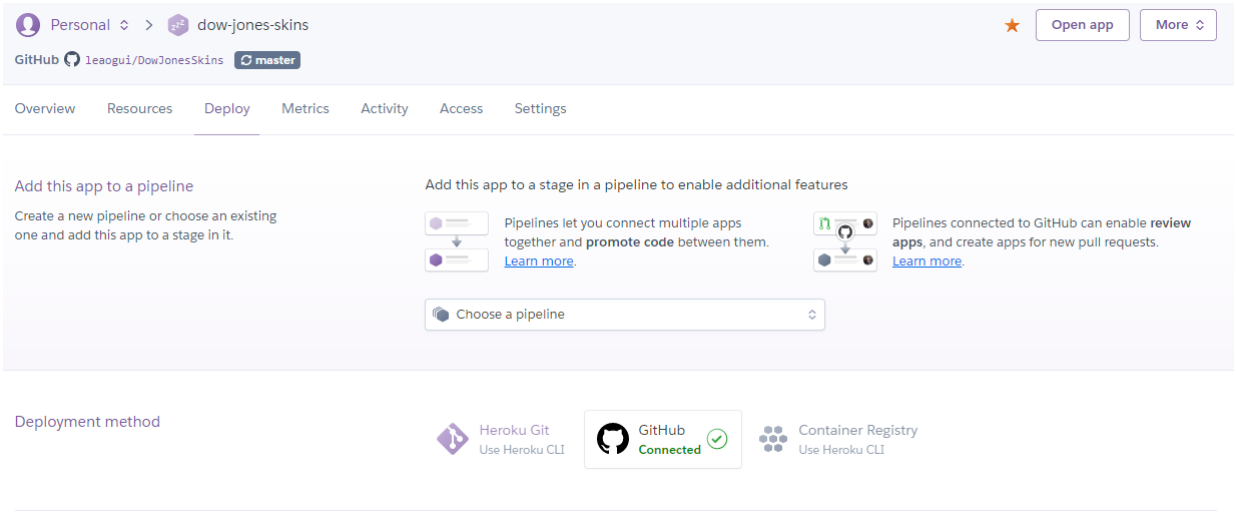

Figura 2 – App do Projeto no Heroku

4.2.4 Home.html + app.js

Para criar o esboço da Home do projeto, o HTML5 foi utilizado. Para estilização, o Bulma foi explorando, que consiste num FrameWork de CSS. Cada arquivo foi separado em pastas baseadas em suas extensões. Diante disso, uma pasta HTMl, uma CSS e uma JS foram criadas. Na pasta CSS, o .min.css do Bulma foi salvo, para que fosse possível utilizá-lo.

Com a Home criada, o próximo passo foi o primeiro contato com o login via Steam. Para isso, a API da Steam para Node.JS foi utilizada. No arquivos app.js, que consiste no arquivo 'main', todas as rotas foram criadas e o framework "express"foi utilizado para instanciação do servidor web. Neste arquivo, as rotas da API Steam foram implementadas, configurando o middleware, informando a API Key e domínio. Por conta da execução ser tanto localmente quanto no Heroku, vários arquivos JS com functions fororam modularizados para realizar essa troca de domínio, porta e API Key, já que tais keys mudam para cada domínio (local e Heroku).

Para essas funções foram criadas testes unitários utilizando o Jest. Esses testes estão na subpasta 'tests' dentro da pasta 'JS'.

Figura 3 – Esboço da Home pronto

4.2.5 Engenharia de Software

No PivotalTracker criado pelo professor Daniel, as primeiras User Stories foram inseridas. Essas Stories consistem nas principais funcionalidades de usuários identificadas pelo grupo e divididas até então. Após a inserção, Uma pontuação foi atribuída para cada uma, a partir de um debate, onde o integrante que deu a menor nota defendia seu argumento junto ao integrante de deu a maior nota. O melhor argumento decide a nota.

Após isso, a prioridade e dependencias de cada User Story foi organizada, além de lançar algumas Users Stories no IceBox

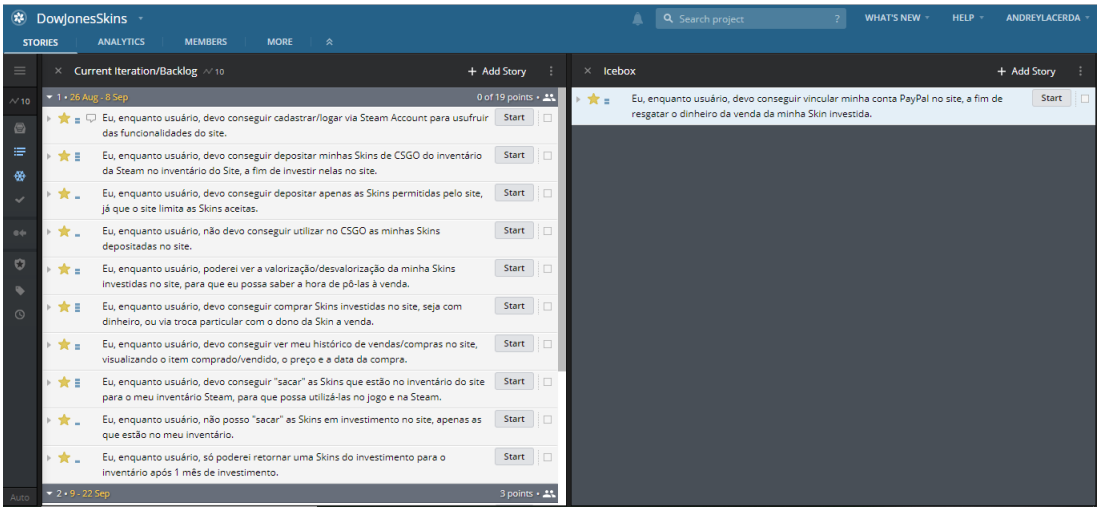

Figura 4 – PivotalTracker no 1° Sprint

4.3 2° Sprint

4.3.1 Descrevendo as features com Cucumber

Com as User Stories definidas, o passo seguinte foi esboçar todas as features utilizando o ideal do BDD e utilizando o Cucumber para isso. Com isso, os possíveis cenários de todas as User Stories foram descritos até então, seguindo como base o projeto exemplo mostrado pelo professor Daniel, mais a própria documentação do Cucumber.

Como Dito anteriormente, para isso a sintaxe do Cucumber foi utilizada, e cada feature foi separada em um arquivo diferente, salvas em uma pasta separada, chamada 'Features'. Como sintaxe, cada User Storie foi salva como Feature, com suas descrições, Backgrounds, e assim Cenários.

4.3.2 Testando cadastro/login com PostgreSQL do Heroku

Para começar o sistema de cadastro/login, o add-ons do PostGre foi adicionado ao Heroku via Heroku CLI. Para isso, o comando 'npm install pg' foi utilizado para instalar o Postgres do projeto. Após isso, o add-on do Postgre foi instalado no Heroku via CMD, com comando 'heroku addons:create heroku-postgresql:hobby-dev'. Após isso, os passos que estão na documentação do Heroku Postgre foram seguidos, encontrados em 'https://devcenter.heroku.com/articles/heroku-postgresql'.

Após isso, as tabelas foram criadas via CMD, usando comando 'heroku psql' e adicionando todas as linhas de comando sql. Tendo a conexão estável e as tabelas criadas, uma classe chamada 'UserCRUD' foi desenvolvida, e nela o JSON enviado pela Steam após conexão foi utilizado para salvar um usuário novo no banco de dados.

Nessa classe, a função signUp() recebe o JSON da Steam, conecta ao banco de

dados e verifica se este usuário já foi cadastro. Se sim, mais nada é feito, mas caso contrário, o usuário é inserido no banco de dados. Tendo isso feito, a primeira parte do sistema de cadastro/login foi finalizada.

4.3.3 Estabelecendo Sessão

A fim de estabelecer o sistema de login, a utilização de sessão para bloquear o acesso de usuário a determinadas páginas do site foi a solução mais viável. Para isso, a constante Session da biblioteca 'Express' foi utilizada.

O comando app.use() foi utilizado para usar o Session. Neste etapa, um id paraa sessão foi inserido dentro do parâmetro de inicialização 'secret', e false foi posto nos outros dois itens de inicialização, saveUninitialized e resave, que, após pesquisar, foi decidido que não fariam sentido algum.

Após isso, dentro da página '/verify' foi colocada a criação da sessão de acordo com o login do usuário na Steam. Com o usuário autenticado via Steam, o comando 'req.session.user' foi itilizado e o JSON fornecido pela Steam foi atribuído à essa constante.

Com isso, uma sessão é criada para cada usuário logado, e após isso, para bloquear o acesso à páginas sem que o user esteja logado, um 'IF' foi inserido em cada get do servidor (ignorando a /index e a /login). Dentro do IF, '!req.session.user' foi inserido, mostrando que caso não exista ression, o usuário deveria ser redirecionado para a tela de login da Steam, finalizando assim o ideal de sessão do site.

4.3.4 Visualizando o inventário Steam do usário

Para obter acesso ao inventário Steam do usário logado, a biblioteca 'steam-inventory-api' foi utilizada. O serviço da API é inicializado quando o servidor sobe. Após isso, ao se loga na Steam, o usuário é redirecionado para o '/verify' onde, além de ter a sessão instanciada, os itens de seu inventário são visualizados.

Para isso, a steamId do usuário de dentro do JSON fornecido pela Steam foi recuperada e inserida como um dos diversos parâmetros da API de inventário. Constantes foram utilizadas para facilitar o preenchimentos dos outros parâmetros de inicialização, seguindo como base o exemplo de uso da API fornecido em 'https://www.npmjs.com/package/steam-inventory-api'.

Com isso, ao logar no Steam, o sistema consegue capturar todos os itens trocáveis do inventário do usário, além de filtrar para o jogo 'CSGO'.

4.4 3° Sprint

4.4.1 Personalização via Cookie

A fim personalizar a tela do usuário logado, cookies foram utilizados para exibir seu avatar e seu username. Para isso, o JSON disponibilizado pela steam logo após a autenticação do usuário é salvo no cookie e posteriormente resgatado por duas funções JavaScript encarregadas de resgatar o avatar e o username do user.

Com isso, o avatar do usuário foi printado na navbar de todas as telas, enquanto logado, enquanto seu nome aparece na página 'Minha Conta'.

4.4.2 Template Engine Handlebars

Conforme o projeto foi evoluindo, sua complexidade foi aparecendo. Sabendo que seria necessário o retorno de muita informação do banco de dados para a tela da aplicação, além de muitas regras que resultariam em manipulação do banco, tornou-se necessária a utilização de um Template Engine, tornando a aplicação inteiramente back-end.

Diante disso, o 'Handlebars' foi instalado no sistema e marcado como template engine. Após isso, todos os HTMls desenvolvidos até então foram trocados para Handlebars, e as devidas alterações no back-end foram realizadas.

O HBS foi a escolha por conta de sua natureza totalmente virada para o Node.js, o qual foi utilizado para o back-end inteiro, e por conta de seu fácil aprendizado e resultado que solucionaria os problemas do projeto.

4.5 4° Sprint

4.5.1 Mecânica de Depósito e Saque de Skins

A fim de criar a mecânica de deposito e saque de skins, a API 'steam-inventory' foi utilizada, junto a algumas funções próprias.

Ao acessar o próprio inventário do DJS, a API 'steam-inventory' é utilizada, retornando todas as skins trocáveis de Counter-Strike: Global Offensive. Com essa informação, o nome de tais skins são printadas na tela do usuário em uma coluna a esquerda da tela, enquanto apenas as aceitadas pelo site são printadas na coluna do meio. Na coluna a direita, por sua vez, foi utilizada para printar as skins já depositadas no site pelo usuário.

Com todas as informações na tela, botões foram adicionados. Na coluna do meio, botões que chamam uma função para investir a skin clicada foram adicionados, enquanto

na coluna a direita, botões para sacar skin e investir na skin foram adicionados. A coluna a esquerda ficou apenas com nomes de skins.

Os botões chamam funções, que por sua vez, realizam as devias validações e manipulações no banco de dados, possibilitando assim o mecanismo de deposito e saque.

Devido a burocracias com a Steam, não foi possível criar uma conta para realizar as trocas reais de skins. Com isso, ao depositar uma skin, tal skin não some do inventário real do usuário, apenas é mapeado para dentro do inventário do DJS. A mesma coisa para o saque, em que a skin é realmente enviada para a Steam, mas sim apenas retirada do inventário do DJS, simulando tal troca.

4.5.2 Mecânica de Investimento de Skins

A fim de investir na skin, mecanismo core da aplicação, o botão na 'Investir' foi criado para cada skin no inventário DJS do usuário. Ao clicar no botão, uma função é invocada, que realiza a confirmação do investimento da skin, marcando a data do investimento. Com a data marcada, o usuário não pode retirar o investimento até bater 31 dias, que consiste em outro User Story do projeto.

A fim de simular a flutuação de valores das skins do mercado, ao investir em uma skin, caso a skin investida é a primeira no sistema inteiro, seu valor aumenta 15%. Caso contrário, cai 3%.

4.6 5° Sprint

4.6.1 Mecânica de Carteira

Após a implementação do deposito, saque e investimento de skins, a mecânica de carteira foi desenvolvida para possibilitar a compra de skins. Para isso, foi criado uma sessão 'Carteira', e uma coluna 'saldo' na tabela 'usuario' do Postgresql.

Ao clicar em 'Depositar' na sessão de 'Carteira', automaticamente R$15 creditados em seu 'saldo', realizando um UPDATE no banco de dados. Ao informar uma quantia a sacar e clicar em 'Retirar', tal quantia é validada com o saldo. Se o saldo for maior ou igual a quantia, tal valor é debitado do saldo, realizando outro UPDATE no banco de dados.

4.6.2 Mecânica de Compra de Skins e Histórico de Vendas

Para o mecanismo de compra, foi criado a sessão 'DayTrade', em que o usuário acessará para ver todas as skins investidas no site e seu determinado preço, além de ver

seus próprios investimentos, possibilitando o cancelamento de algum deles, caso tenham mais de 31 dias.

Com isso, ao encontrar um skin que deseja, o usuário clica em 'Comprar', e uma função é chamada para validar a compra. Se a compra for válida, o usuário comprador recebe a skin do usuário "vendedor" que está a mais tempo com o investimento aberto. Com isso, do saldo do comprador é debitado o valor da skin, enquanto do saldo do vendedor é creditado tal valor.

Além disso, um histórico de vendas é criado, marcando assim o nome do vendedor/comprador, as informações da skin comprada e a data da compra. Tal histórico também pode ser visto na tela 'DayTrade'.

4.6.3 Testes

Após todos os desenvolvimentos serem concluídos e testados pela equipe, o desenvolvimento de teste automatizados foram realizados. Para isso, o framework de testes Jest foi utilizado para o testes unitários e de integração, enquanto o Puppeteer e o Cucumber foram utilizados para os testes End-To-End.

Ao total, dezesseis testes foram implementados, testando as funcionalidades principais do sistema, evitando qualquer má manipulação de banco de dados e interferência no mecanismo de valorização de skins do site.

Além desses testes, uma ferramenta de análise estática foi utilizada no projeto. Tal ferramenta foi o 'jshint', e serviu como "cleaner" de "code smells". A análise estática de 100% ao final das refatorações requisitadas pela ferramenta.

5 Diagramas

Durante o desenvolvimento do projeto, alguns diagramas foram elaborados para sincronização de pensamento da equipe e entendimento coletivo.

Dentre os diagramas construídos, dois consistem no banco de dados, para entendimento da conexão entre as classes (Figura 5 e 6), enquanto sete consistem na aplicação (Figura 7, 8, 9, 10, 11, 12 e 13), para entendimento de suas mecânicas, da interação do usuário com o sistema, e do funcionamento interno. Os diagramas utilizados para a aplicação foram o casos de uso e diagrama de atividades.

Os diagramas elaborados foram:

5.0.1 Banco de Dados

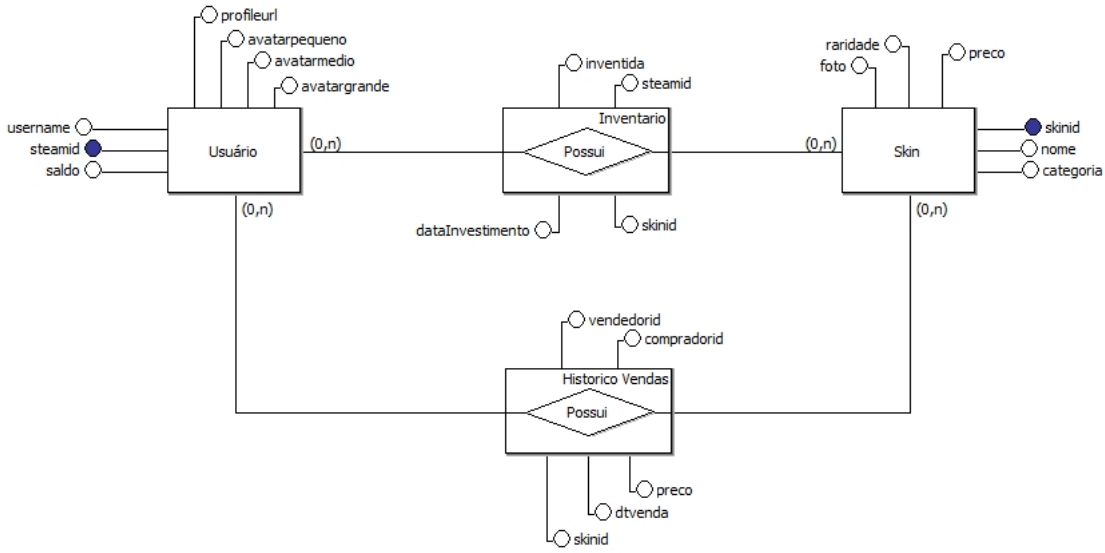

Figura 5 – Banco de Dados - Diagrama Entidade Relacionamento

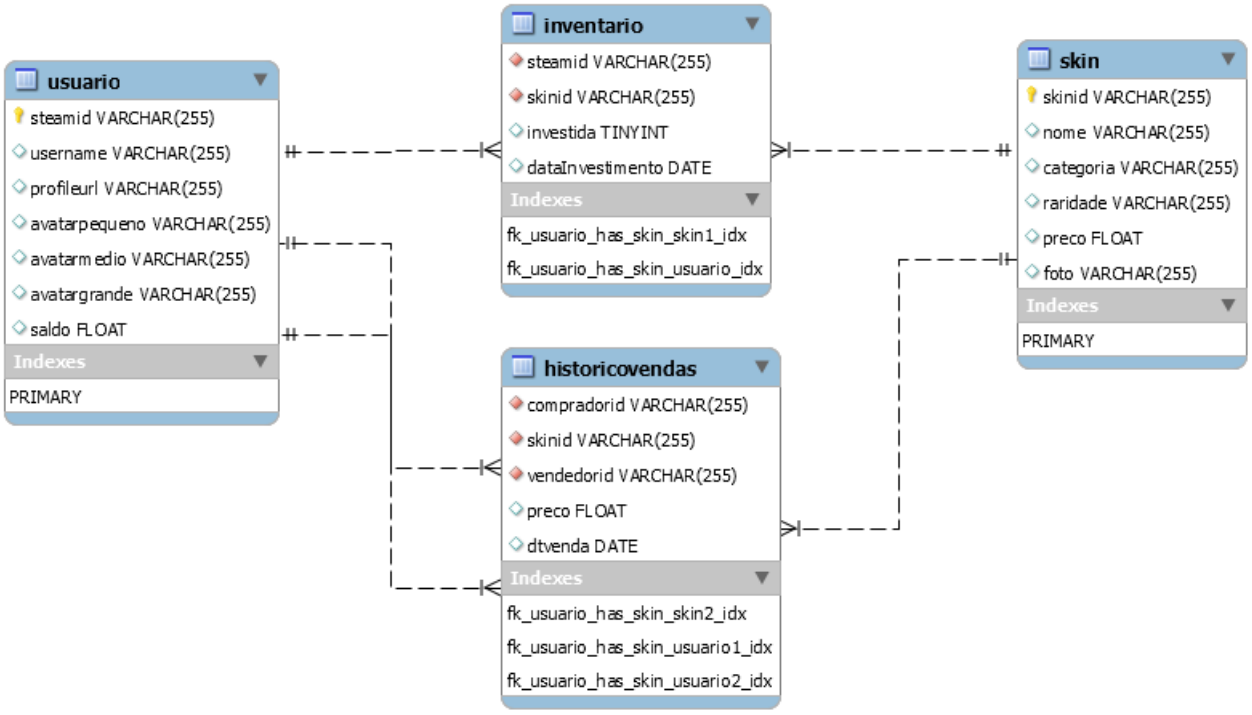

Figura 6 – Banco de Dados - Modelo Lógico

5.0.2 Casos de uso

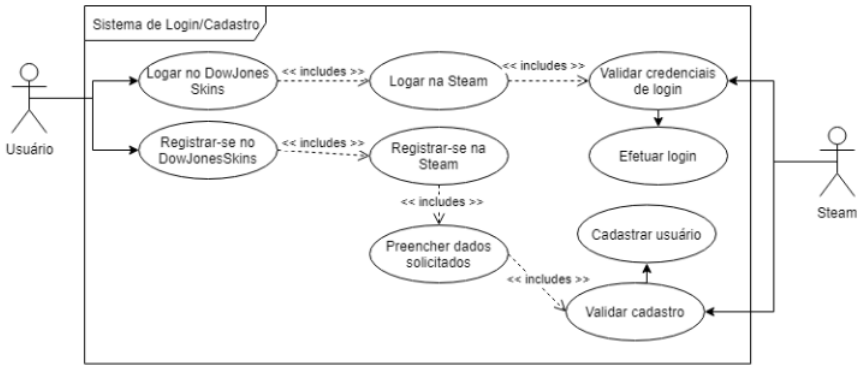

Figura 7 – Caso de Uso - Login do usuário

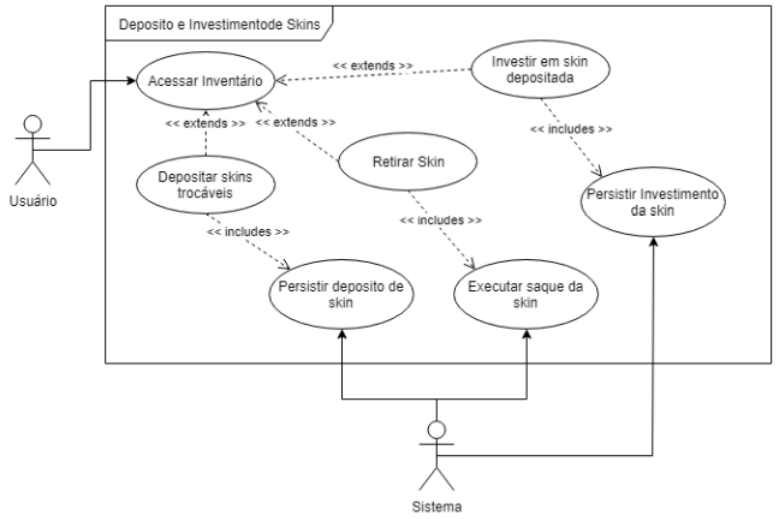

Figura 8 – Caso de Uso - Depósito de Skins

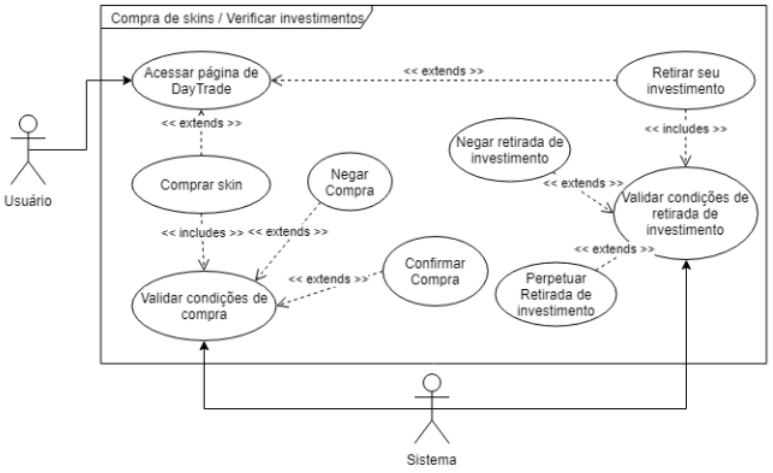

Figura 9 – Caso de Uso - Compra de Skins

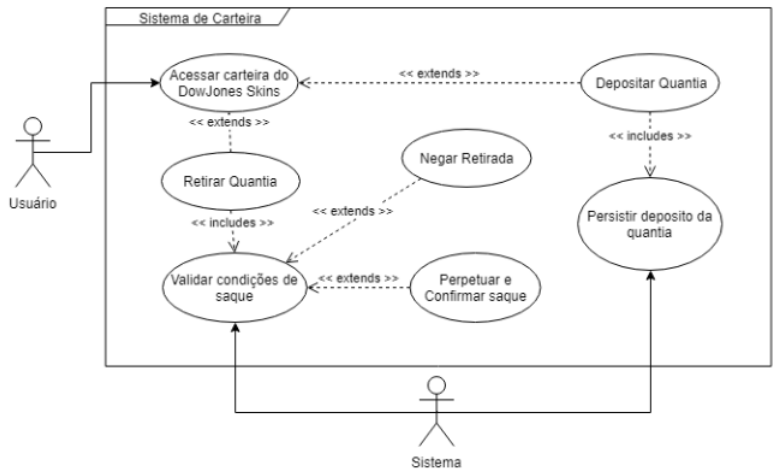

Figura 10 – Caso de Uso - Carteira do usuário

5.0.3 Diagramas de Atividade

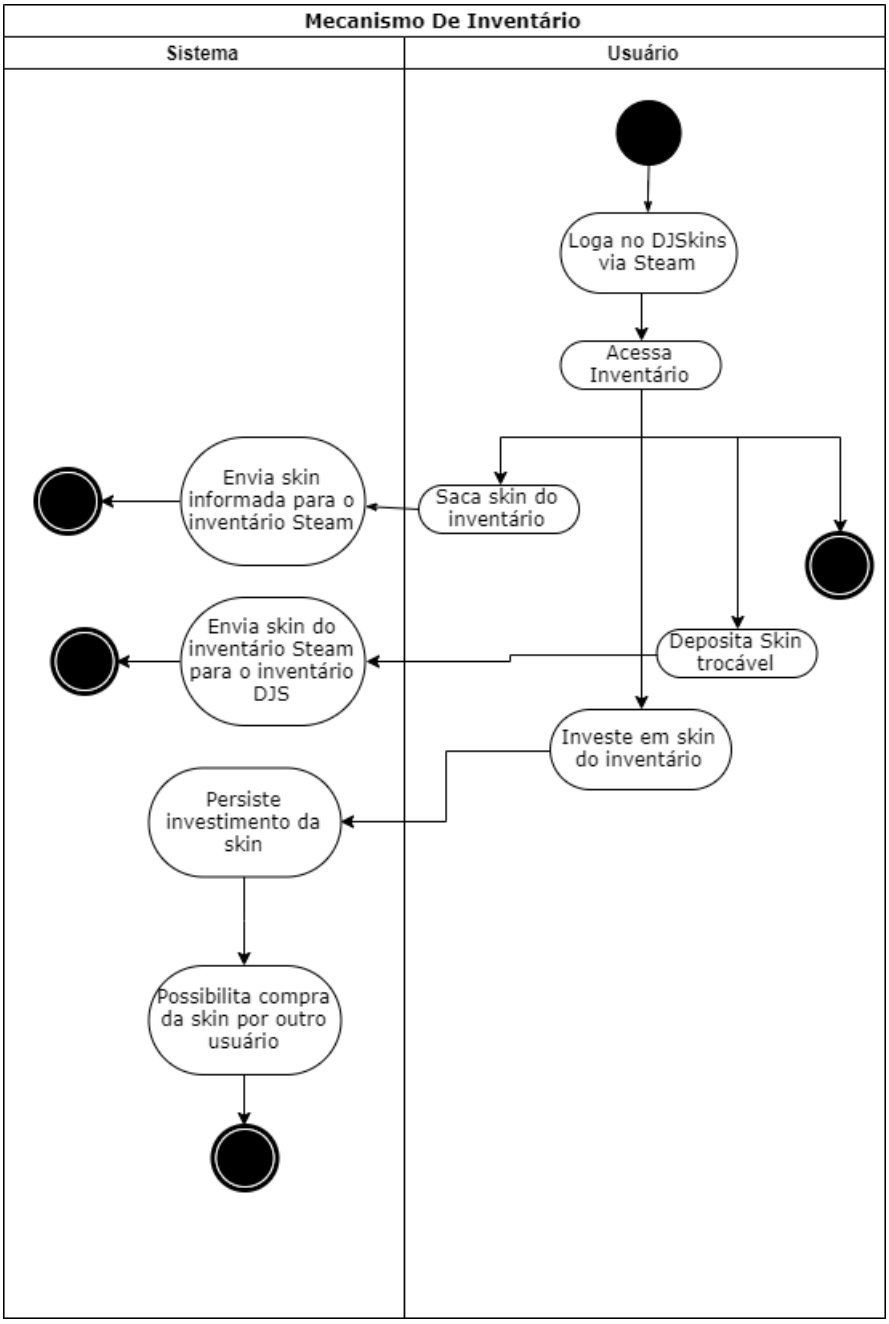

Figura 11 – Diagrama de Atividades - Mecanismo de Inventário

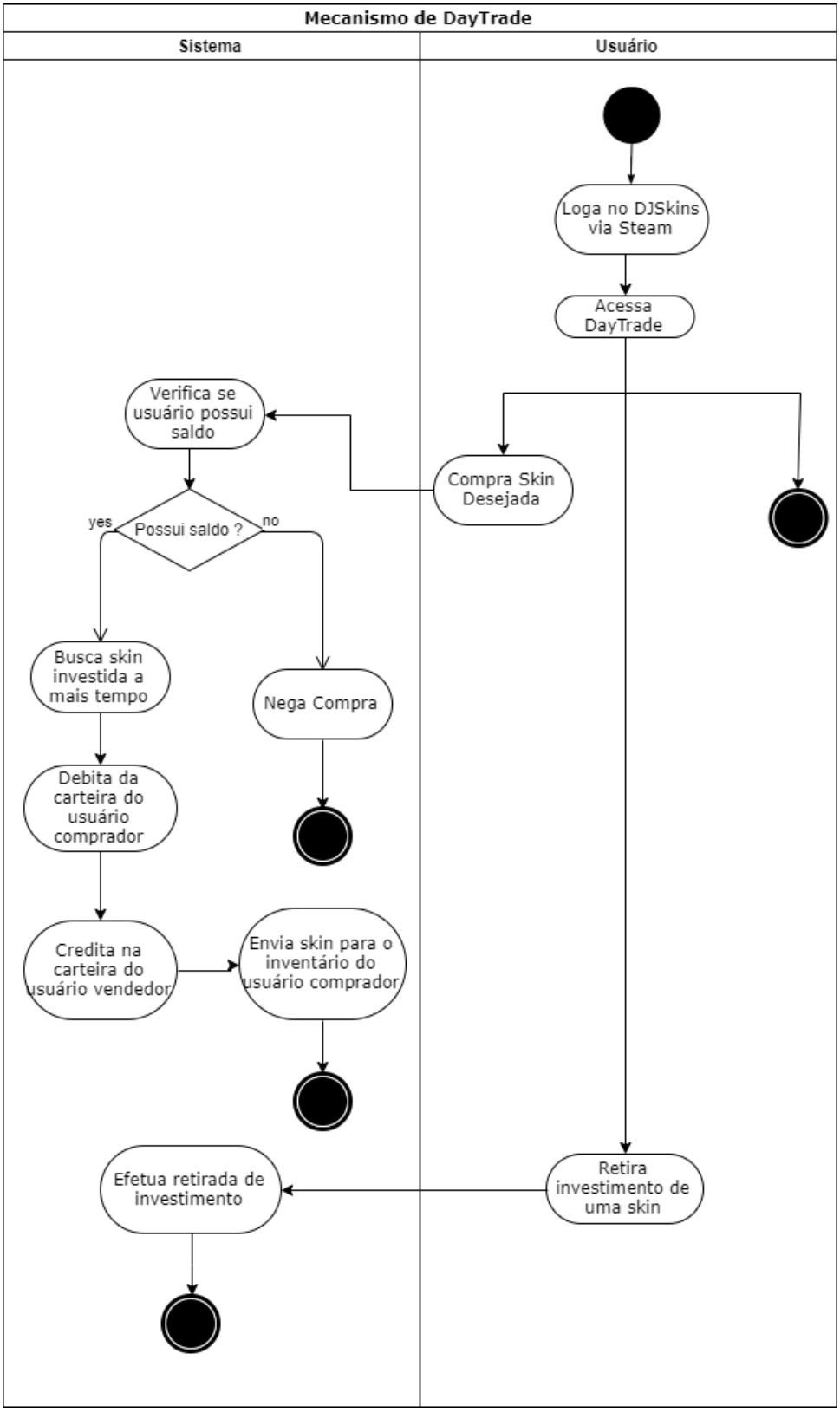

Figura 12 – Diagrama de Atividades - Mecanismo de Day Trade

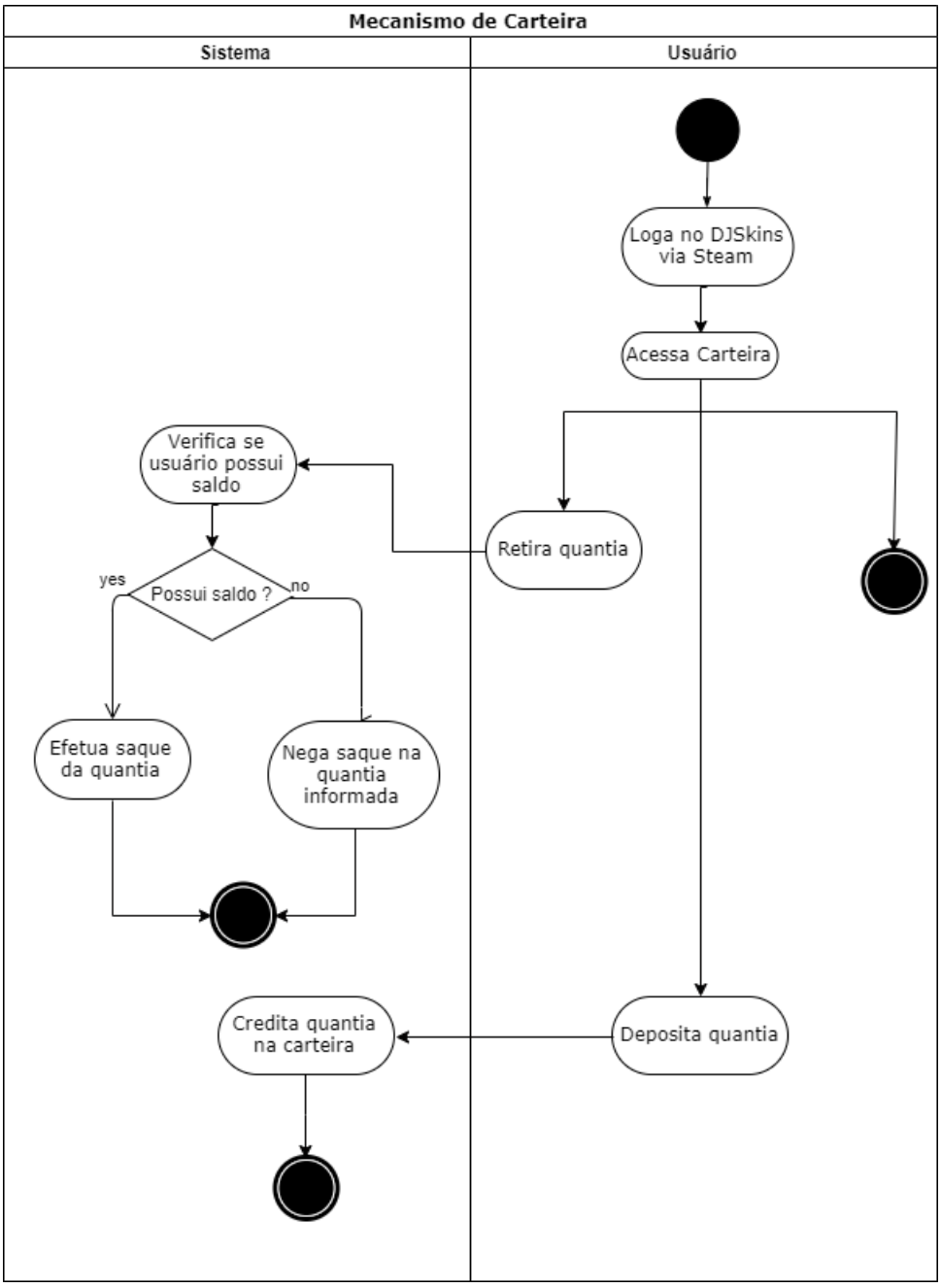

Figura 13 – Diagrama de Atividades - Mecanismo de Carteira

Conclusão

Em suma, o projeto construído consiste em uma aplicação web para vendas de skins do jogo Counter-Strike: Global Offensive.

Node.js foi utilizado para o desenvolvimento back-end, enquanto o HTML5 foi utilizado para o front-end. Como template-engine, o handlebars foi utilizado, tornando a aplicação server-side.

Como metodologia de engenharia de software, os métodos ágeis foram aplicados, e o desenvolvimento do sistema foi feito em cima de Users Stories separadas por sprints, similarmente a metodologia XP.

Para a implementação do projeto, duas APIs da Steam foram utilizadas, sendo uma para autenticação, e outra para resgate de inventário. O GitHub foi utilizado para como gerenciador de versão, enquanto o Heroku foi utilizado como ferramenta de deploy.

Além disso, foi utilizado o PostgreSQL do Heroku como banco de dados da aplicação, salvando assim as skin, as contas dos usuários, seus inventários e seus históricos de venda.

Como testes, testes unitários e de integração foram implementados em grande quantidade, enquanto um único teste end-to-end automatizado foi desenvolvido. A ferramenta de análise estática 'jshint' foi utilizada para neutralização de code smells.

Ao final de todo o desenvolvimento, a aplicação construída foi um sucesso, abraçando mecânicas de depósito, retirada, compra, valorização e investimento de skins, além de ter um mecanismo de carteira. Todos esses mecanismos feitos de forma simulada, não realizando a real troca ou compra de skins, assim como deposito e saque de dinheiro da carteira.

Referências

DOCUMENTAÇÃO HEROKU POSTGRESQL. HEROKU. Disponível em: <https://devcenter.heroku
postgresql>. Acesso em: 04 Out. 2019.

DOCUMENTAÇÃO STEAM-INVENTORY-API. Disponível em: <'https://www.npmjs.com/package/
inventory-api'>. Acesso em: 02 Out. 2019.

ITENS 'COSMÉTICOS' MOVIMENTAM CULTURA E ECONOMIA DOS JOGOS. E-Arena. Disponível em: <https://e-arena.com.br/itens-cosmeticos-movimentam-cultura-e-economia-dos-jogos/>. Acesso em: 14 Out. 2019.

MERCADO DE SKINS DE CS:GO PODE MOVIMENTAR ATÉ 10 BILHÕES DE DÓLARES POR ANO. The Enemy. Disponível em:<https://www.theenemy.com.br/esports/csgo-mercado-skins-10-bilhoes-valores-precos>. Acesso em: 14 Out. 2019.

NEWZOO ARTICLES. NewZoo. Disponível em: <https://newzoo.com/insights/articles/>. Acesso em: 15 Out. 2019.

PUC PR. Mercado De Jogos Digitais Cresce No Brasil E No Mundo. G1 Globo, 08/10/2018. Disponível em: <https://g1.globo.com/pr/parana/especial-publicitario/puc-pr/profissionais-do-amanha/noticia/2018/10/08/mercado-de-jogos-digitais-cresce-no-brasil-e-no-mundo.ghtml>. Acesso em: 15 Out. 2019.

SETOR DE GAMES CRESCE ACIMA DA MÉDIA NO PAÍS, MAS É O 13º DO MUNDO. O Tempo. Disponível em: <https://www.otempo.com.br/economia/subscription-required-7.5927739?aId=1.2224441>. Acesso em: 15 Out. 2019.