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 Urdiali da Silva	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 Urdiali da Silva	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 (2019), empresa especializada em coleta e estudo de dados do mercado de jogos digitais, e nos artigos da PUC-PR (2018) e TEMPO (2019). 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. Tal nicho vem crescendo fortemente, como diz artigo E-ARENA (2018).

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 ENEMY (2019), 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 levando 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 comprador 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 o 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 seja possível 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 sobre 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 visto no artigo de pesquisa da NEWZOO (2019) 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 ENEMY (2019).

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 tornando 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 também utilizam suas skins para uso comercial e como forma de obter capital, 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

Neste capítulo, o desenvolvimento projeto será explicitado e comentado. Por conta da metodologia de desenvolvimento ágil escolhida, cada Sprint possui uma seção no documento, contendo o que foi realizado nela.

4.1 User Stories

Como descrição das necessidades dos usuários desta aplicação, servindo como casos de uso + requisitos funcionais, as seguintes Users Stories 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 Skins do meu inventário do site, para que elas possam 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 Skin 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.

Para maiores detalhes, alguns cenários de User Story foram levantados e exibidos no Apêndice A.

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 Pivotal Tracker, a fim de inserir e organizar 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, como mostra a Figura 1, e posteriormente adicionou todos os membros da equipe e o professor de ES4A4, Daniel Morais.

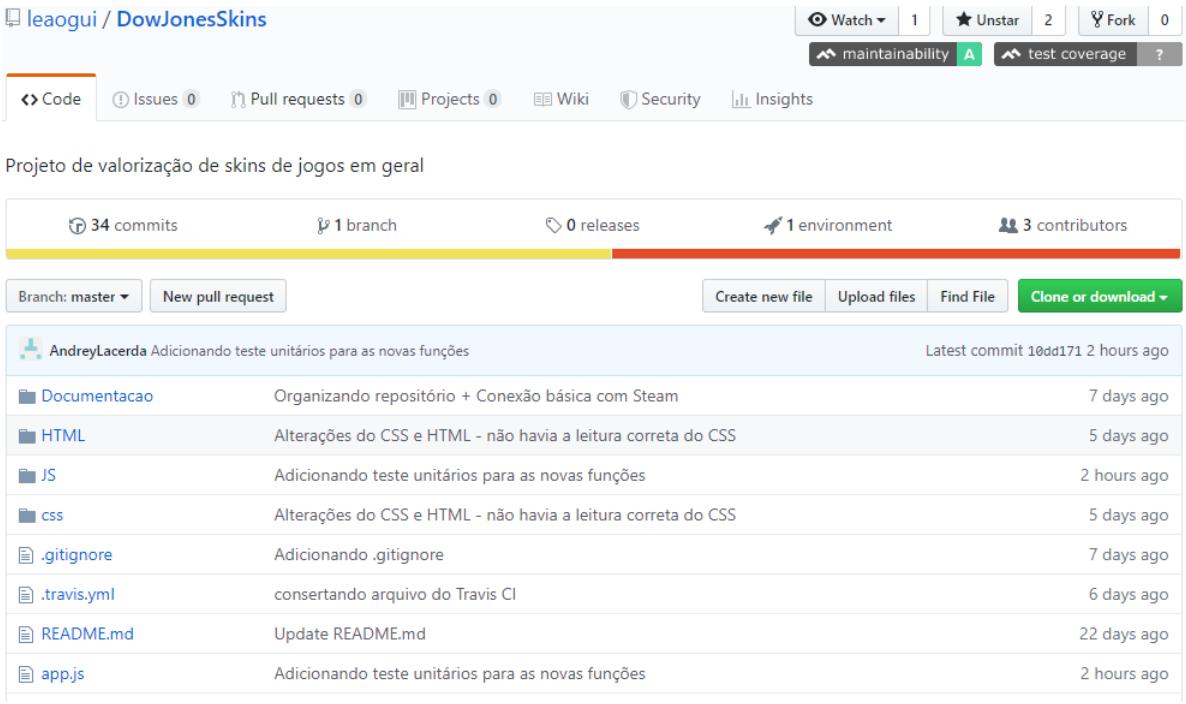

Figura 1 – Repositório do Projeto no GitHub

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, outros 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 MarketPlace".

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, como mostra a Figura 2. Após criar o Heroku App, o deploy do Heroku foi configurado para que ele seja sincronizado com o "push" para o repositório no GitHub.

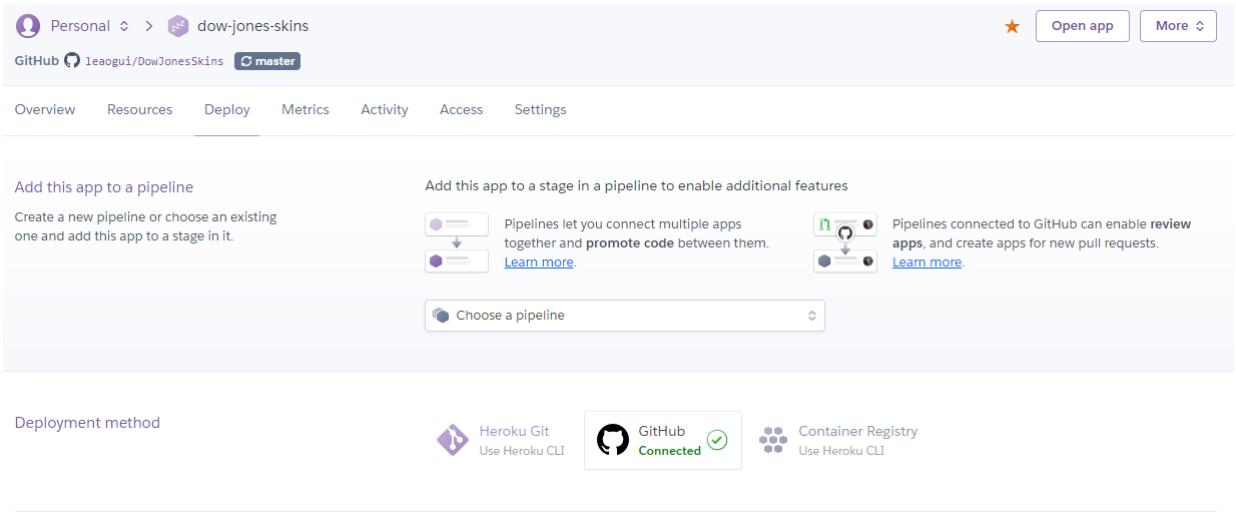

Figura 2 – App do Projeto no Heroku

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. A Figura 3 mostra o estado final da construção da primeira "Home" da aplicação.

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 arquivo "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 foram 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 Pivotal Tracker 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 User Stories no IceBox, fechando a primeira organização do Pivotal Tracker, como pode ser vista na Figura 4.

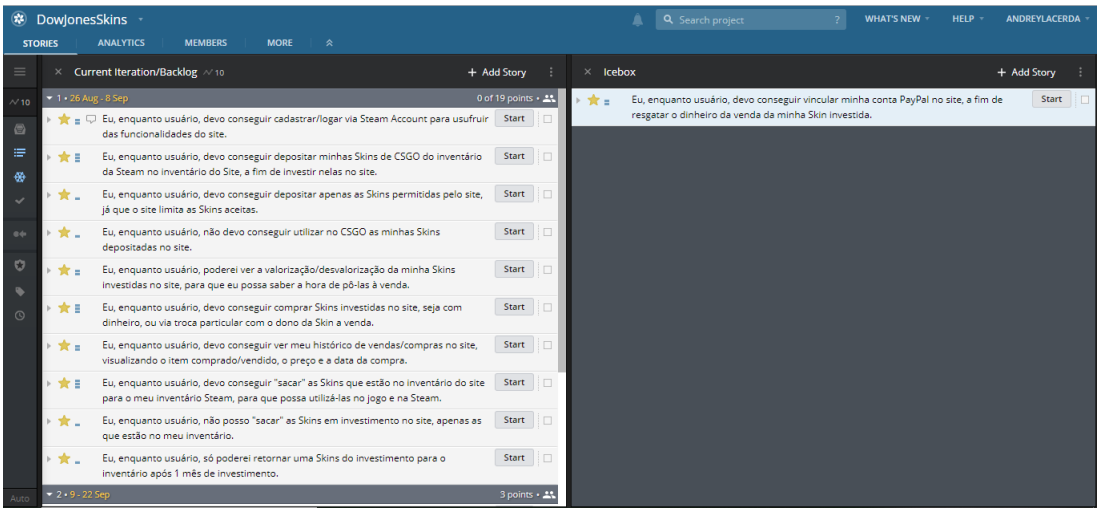

Figura 4 – PivotalTracker no 1° Sprint

4.3 2° Sprint

O segundo sprint consistiu na descrição dos User Stories usando Cucumber, na implementação do sistema de cadastro/login usando o PostgreSQL do Heroku e a Steam, implementando sessões de login na aplicação e utilizando a steam-inventory-api para visualizar o inventário do usuário logado.

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 Story 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 HEROKU-POSTGRES (2019) foram seguidos.

Após isso, as tabelas foram criadas via CMD, usando comando "heroku psql" e adicionando 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 para a 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 utilizado 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 usuário

Para obter acesso ao inventário Steam do usário logado, a biblioteca Steam-Inventory-API (2017) foi utilizada. O serviço da API é inicializado quando o servidor sobe.

Após isso, ao se logar 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 no site fonte.

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

4.4 3° Sprint

O terceiro sprint consistiu nas primeiras personalizações do site, via cookie, e na migração do front do projeto para Handlebars.

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

O quarto sprint consistiu na implementação da mecânica de depósito, saque e investimentos de skins.

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 não 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

O quinto sprint foi o último antes da entrega da documentação e apresentação do projeto, e consistiu na implementação do mecanismo de carteira, compra de skins e histórico de vendas, além do desenvolvimento de alguns testes automatizados.

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 seção 'Carteira', e uma coluna 'saldo' na tabela 'usuario' do PostgreSQL.

Ao clicar em 'Depositar' na seção 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 seção '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% foi conquistada após as refatorações requisitadas pela ferramenta.

5 Diagramas

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

Como forma de organização, os diagramas serão apresentados separadamente em dois grupos, representados pelas seções a seguir.

5.1 Banco de Dados

Dentre os diagramas construídos, dois consistem no banco de dados, para entendimento da conexão entre as entidades do sistema, sendo um deles Diagrama de Entidade-Relacionamento, que consiste na Figura 5 e mostra o relacionamento entre as entidades do sistema, enquanto o outro consiste no Diagrama do Modelo Lógico do banco de dados da aplicação, representado na Figura 6, mostrando essas relacionamentos de forma menos abstrata, já com os campos de cada tabela e seus tipos.

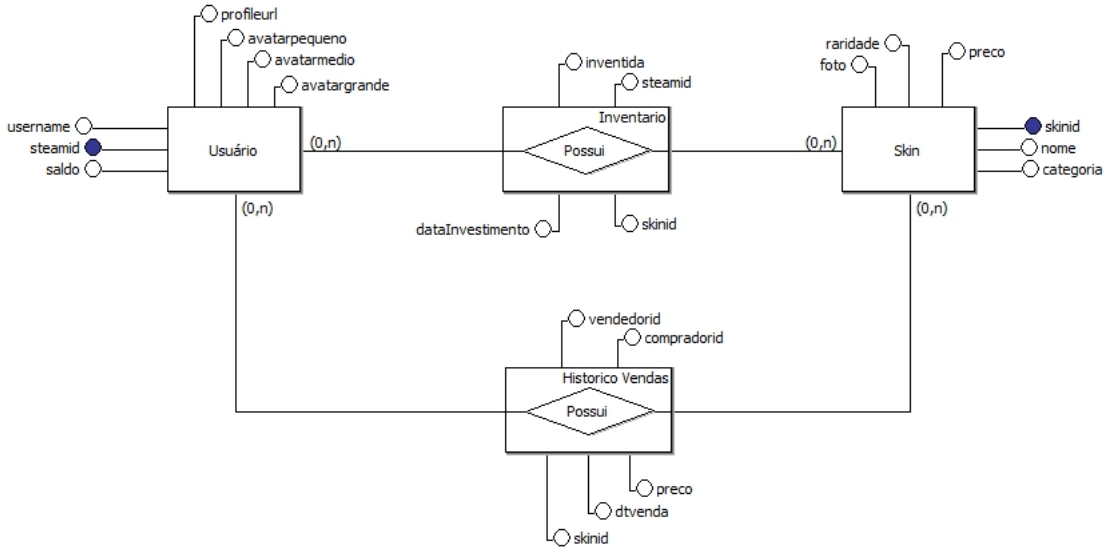

Figura 5 – Banco de Dados - Diagrama Entidade Relacionamento

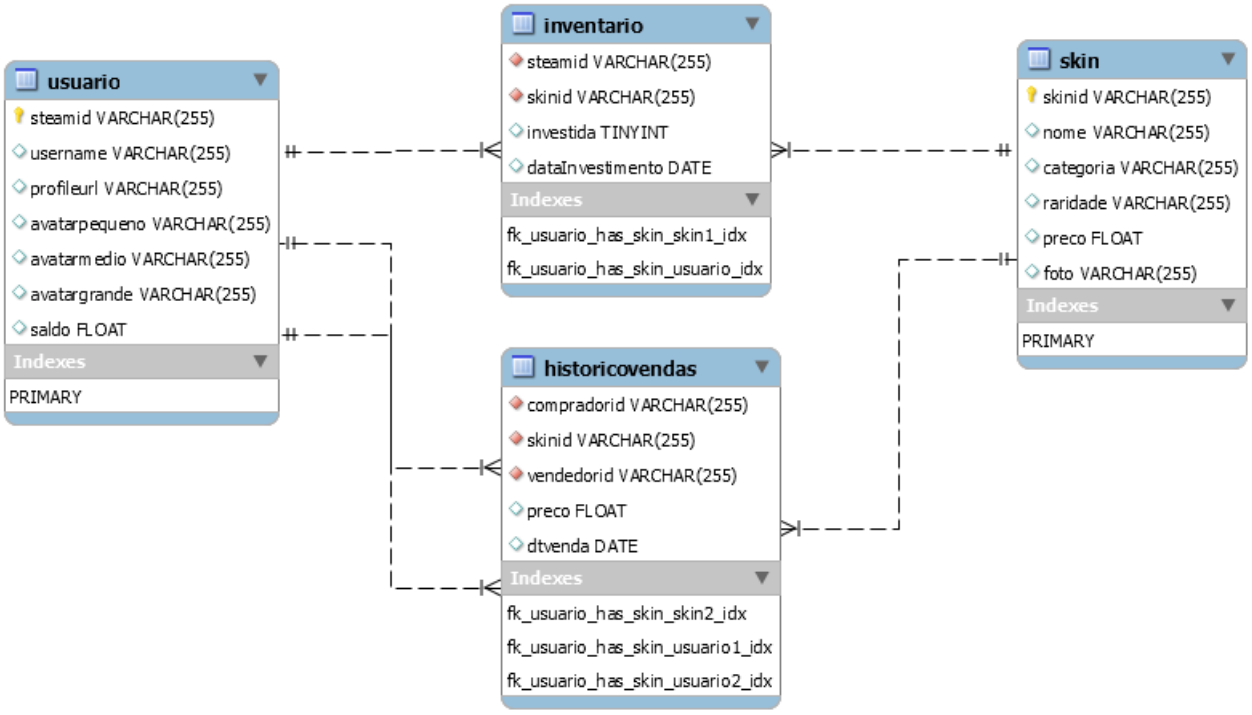

Figura 6 – Banco de Dados - Modelo Lógico

5.2 Diagramas de Atividade

Além desses dois diagramas apresentados acima, três Diagramas de Atividade foram construídos para auxiliar no alinhamento de ideias do grupo e entendimento das funcionalidades.

O diagrama representado na Figura 7 consiste no funcionamento do mecanismo de inventário do projeto, enquanto a Figura 8 representa o funcionamento da mecânica de Day Trade e a Figura 9 representa o sistema de carteira da aplicação.

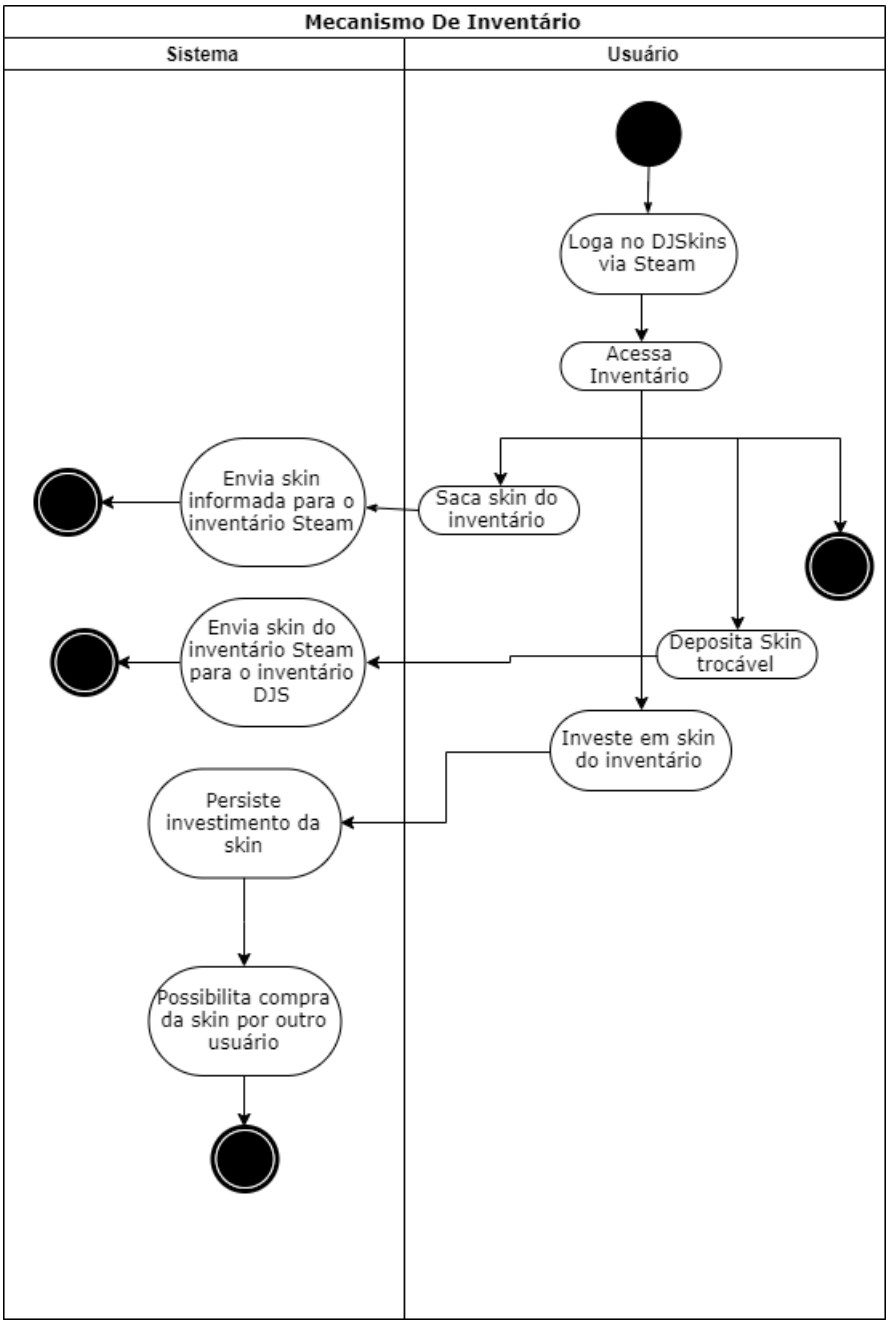

Figura 7 – Diagrama de Atividades - Mecanismo de Inventário

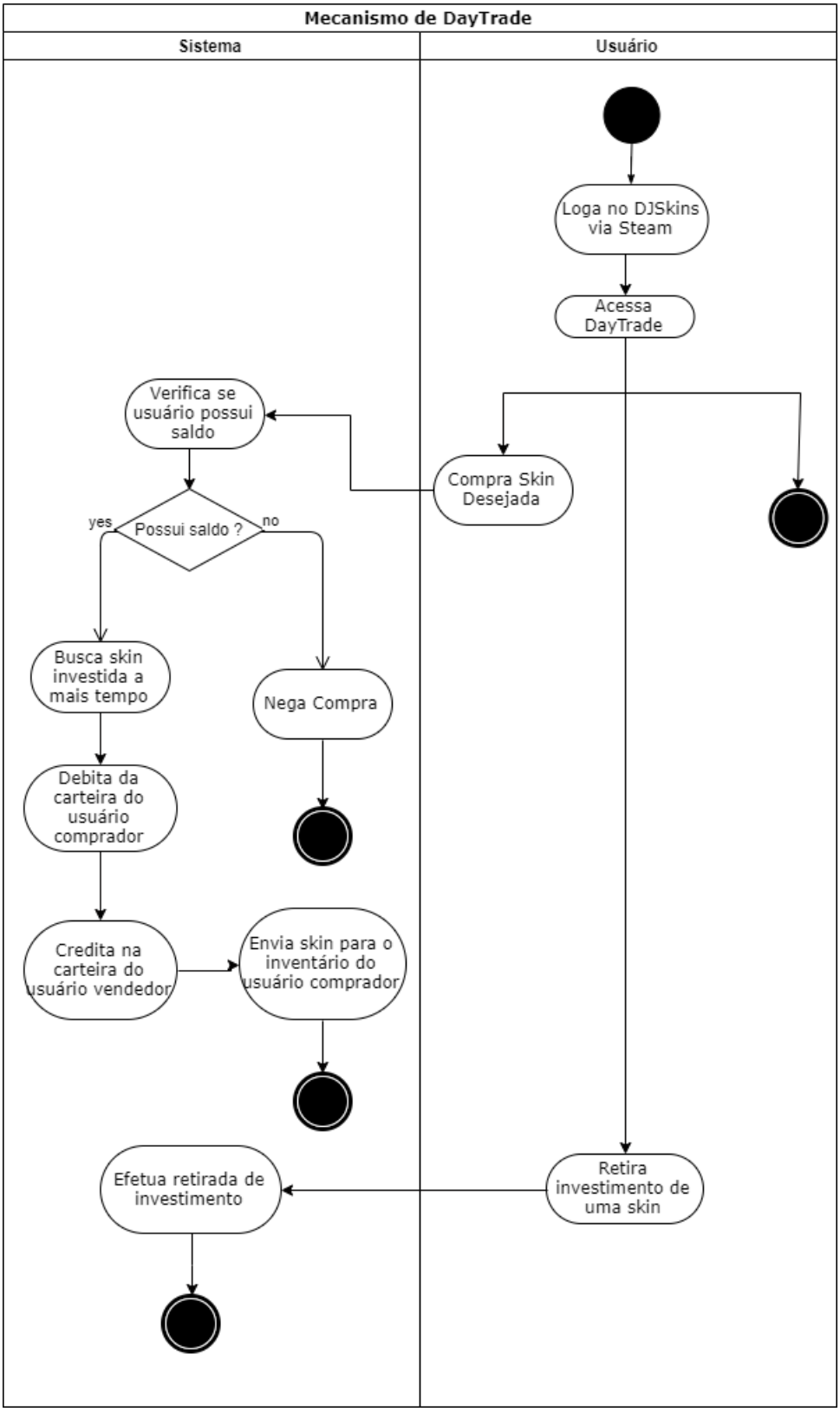

Figura 8 – Diagrama de Atividades - Mecanismo de Day Trade

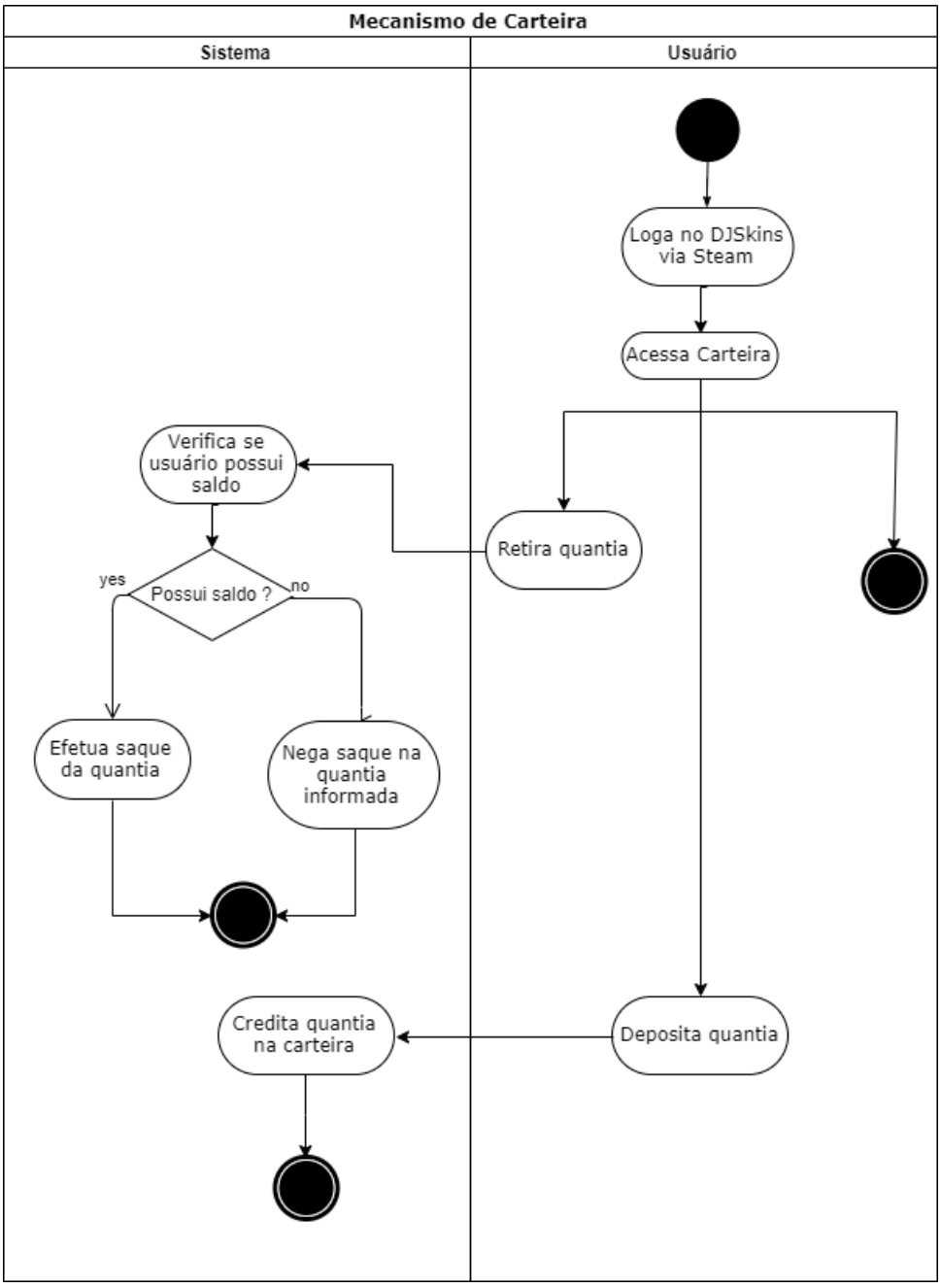

Figura 9 – Diagrama de Atividades - Mecanismo de Carteira

Conclusão

Em conclusão, a aplicação foi construída usando Node.js como back-end e HTML5 como front-end. As dificuldades encontradas nesta fase foram causadas por conta da falta de conhecimento em back-end e em Node.js, o que ocasionou a utilização do template engine Handlebars. Contudo, o aprendizado foi consideravelmente rápido, o que possibilitou a finalização das mecânicas principais prometidas ao início do projeto.

O GitHub foi utilizado como gerenciador de versão e não levantou grandes problemas ou mistérios. Em geral, a utilização foi feita de forma correta, versionando apenas os arquivos essenciais e excluindo outros, como o node_modules. Porém, a documentação do projeto foi guardada no mesmo repositório que o código, o que não foi uma boa prática. Por mais que não estrague o repositório, o ideal seria utilizar um segundo.

Como ferramenta de deploy, o Heroku foi utilizado e nenhum problema foi encontrado em sua utilização, pois sua documentação foi de bom tamanho para um aprendizado rápido e boa utilização da ferramenta. O addon do PostgreSQL do Heroku foi utilizado e tal demandou uma pesquisa mais aprofundada, pois a documentação por si só não se identificou suficiente para sua utilização de maneira limpa. Todavia, após poucas provas de conceito e algumas pesquisas, sua utilização foi efetiva.

O desenvolvimento do projeto foi feito em sprints, o que foi totalmente novo para a equipe, que se adaptou bem à metodologia de desenvolvimento e conseguiu realizar o projeto a tempo, dividindo razoavelmente bem as tarefas.

Quanto as tarefas, é importante ressaltar que suas escolhas, definições e mensurações foram bem discutidas, o que foi bem crucial para a implementação do projeto, pois culminaram na simplificação das User Stories, exclusão de algumas e troca de ideias em relação a que tecnologias seriam utilizadas para criação do sistema.

Referências

E-ARENA. *ITENS COSMÉTICOS MOVIMENTAM CULTURA E ECONOMIA DOS JOGOS*. 2018. Disponível em: <https://web.archive.org/web/20190111131423/http://e-arena.com.br/itens-cosmeticos-movimentam-cultura-e-economia-dos-jogos/>. Acesso em: 14 Out. 2019. Citado na página 7.

ENEMY, T. *MERCADO DE SKINS DE CS:GO PODE MOVIMENTAR ATÉ 10 BILHÕES DE DÓLARES POR ANO*. 2019. Disponível em: <https://web.archive.org/web/20190721085256/http://www.theenemy.com.br/esports/csgo-mercado-skins-10-bilhoes-valores-precos>. Acesso em: 14 Out. 2019. Citado 2 vezes nas páginas 7 e 8.

HEROKU-POSTGRES. *Heroku Postgres*. 2019. Disponível em: <https://web.archive.org/web/20190615202544/https://devcenter.heroku.com/articles/heroku-postgresql>. Acesso em: 04 Out. 2019. Citado na página 16.

NEWZOO. *NEWZOO ARTICLES*. 2019. Disponível em: <https://web.archive.org/web/20191004203801/https://newzoo.com/insights/articles/>. Acesso em: 15 Out. 2019. Citado 2 vezes nas páginas 7 e 8.

PUC-PR. *Mercado De Jogos Digitais Cresce No Brasil E No Mundo*. 2018. Disponível em: <https://web.archive.org/web/20181009121243/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. Citado na página 7.

STEAM-INVENTORY-API. *Steam inventory API*. 2017. Disponível em: <https://web.archive.org/save/https://www.npmjs.com/package/steam-inventory-api>. Acesso em: 02 Out. 2019. Citado na página 16.

TEMPO, O. *SETOR DE GAMES CRESCE ACIMA DA MÉDIA NO PAÍS, MAS É O 13º DO MUNDO*. 2019. Disponível em: <https://www.otempo.com.br/economia/subscription-required-7.5927739>. Acesso em: 15 Out. 2019. Citado na página 7.

Apêndices

APÊNDICE A – Cenários das User Stories

Alguns cenários foram traçados para indicar possíveis situações que poderiam ocorrer em determinadas User Stories.

Tais cenários foram feito no formato de feature do Cucumber. As User Stories e os cenários são:

```
Feature: Depósito de Skins
    In order to invest in my CSGO Skins
    As an Logged User
    I should be able to deposit my skins on 'DJ Skins Inventory'

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site

    Scenario: A logged user wants to deposit his skin
        Given I am on 'User Page'
        When I click on 'Deposit my Skins'
        Then I should be redirected to 'Dj Skins Deposit page'

    Scenario: A logged user select skins to deposit
        Given I am on 'Dj Skins Deposit page'
        When I select skins to deposit
        Then I should be able to click on 'Confirm Deposit'

    Scenario: A logged user select allowed skins to deposit
        Given I am on 'Dj Skins Deposit page'
        When I select allowed skins to deposit
        And I click on 'Confirm Deposit'
        Then I should be redirected to 'Dj Skins Inventory page'
        And I should see my deposited skins there

    Scenario: A logged user wants to deposit a Skin not allowed by the system
        Given I am on 'Dj Skins Deposit page'
        When I select not allowed skins to deposit
        And I click on 'Confirm Deposit'
        Then I must be warned that I can not do it
```

Figura 10 – User Story de Depósito de skins e seus cenários

```
Feature: Investimento de Skins
    In order to value/devalue my CSGO Skins
    As an Logged User
    I should be able to invest in my deposited skins

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site
        And I have skins deposited

    Scenario: A logged user wants to invest his skin
        Given I am on 'Investment Page'
        When I select my wanted skins
        Then I should be able to click on 'Confirm Investment'

    Scenario: A logged user confirms his investment
        Given I am on 'Investment Page'
        And I already selected my wanted skins
        When I click on 'Confirm Investment'
        Then I should be redirected to 'DJ Skins My investments page'
        And I should see my skins there
        And I should not see my skin on 'DJ Skins Inventory page'

    Scenario: A looged user wants to retain his investmento to his inventory before 1 month
        Given I am on 'DJ Skins My investments page'
        And I select my investment that is not 1 month old
        When I try to click on 'Retain to Inventory'
        Then I should be warned that I can not do it

    Scenario: A looged user wants to retain his investmento to his inventory after 1 month
        Given I am on 'DJ Skins My investments page'
        And I select my investment that is more than 1 month old
        When I click on 'Retain to Inventory'
        Then I should cancel this investment
        And I should not see my skin on 'DJ Skins My investments page'
        And I should see my skin on 'DJ Skins Inventory page'
```

Figura 11 – User Story de Investimento de skins e seus cenários

```
Feature: Sistema de valorização/Desvalorização das Skins investidas
    In order to the Valuation and Devaluation of Skins
    As an Logged User
    I should be able to see an Investment Chart and all Skins invested

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site

    Scenario: A logged user wants to see the Valuation and Devaluation of Skins
        Given I am on 'DJ Skins Tnvestment Page'
        When I look at the page
        Then I should be able to see a Investment Chart
        And I should be able to see the price of all skins invested
```

Figura 12 – User Story de Valorização de skins e seus cenários

```
Feature: Compra/troca das skins investidas
    In order to retain skins or money
    As an Logged User
    I should be able to trade my invested skins

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site
        And I have invested skins
        And I want to trade skins as well

    Scenario: A logged user wants to buy a Skin
        Given I am on 'DJ Skins Tnvestment Page'
        When I select and Invested Skin
        Then I should be able to click on 'Trade'

    Scenario: A logged user trade a Skins succesfully
        Given I am on 'DJ Skins Trade Page'
        When I put the value(money or other skin) of the trade
        And the seller agree
        And the seller click on 'Confirm Trade'
        Then I should be able to click on 'Confirm Trade'
        And buy the skin

    Scenario: A logged user trade a Skins unsuccessfully
        Given I am on 'DJ Skins Trade Page'
        When I put the value(money or other skin) of the trade
        And the seller disagree
        Then I should be able to continue the trade until he accept it
        And give him other proposal

    Scenario: A logged user wants to trade for my skin and I disagree the value
        Given I am on 'DJ Skins Trade Page'
        When I disagree with the value fo the trade
        Then I should be able to click on 'Inform Disagreement of the Term'

    Scenario: A logged user wants to trade for my skin succesfully
        Given I am on 'DJ Skins Trade Page'
        When I agree with the value fo the trade
        Then I should sell my Skin and get my amount of the trade
```

Figura 13 – User Story de Compra de skins e seus cenários

```
Feature: Histórico de Vendas/Compras de investimentos
    In order to see my historic of buys and sells
    As an Logged User
    I should be able to see and Sales Historic Table/Board

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site

    Scenario: A logged user wants to see his historic of buys and sells
        Given I am on 'User Page'
        When I click on 'See my Historic'
        Then I should be redirected to the 'Historic Board Page'

    Scenario: A logged user in the 'Historic Board Page' once upon a time bought/sold Skins on 'DJ
        Given I am on 'Historic Board Page'
        When I see the Board
        Then I should see all my Sales historic, with the price, date and description

    Scenario: A logged user in the 'Historic Board Page' never bought/sold Skins on 'DJ Skins'
        Given I am on 'Historic Board Page'
        When I see the Board
        Then I should see a blank Board
```

Figura 14 – User Story de Histórico de Vendas e seus cenários

```
Feature: Saque das Skins
    In order to use my deposited skins on CSGO
    As an Logged User
    I should be able withdraw my skins on 'DJ Skins'

    Background:
        Given I am an User registered in the Site using my Steam Account
        And I am logged in the Site
        And I have skins deposited

    Scenario: A logged user wants to withdraw his Skin on Inventory
        Given I am on 'DJ Skins Inventory page'
        When I select my wanted skins
        Then I should be able to click on 'WithDraw to Steam'

    Scenario: A logged user confirm WithDraw
        Given I am on 'DJ Skins Inventory page'
        When I select my wanted skins
        And I click on 'WithDraw to Steam'
        Then I should confirm the trade
        And I should be my skin back to Steam Account
        And I should not see my skin at 'DJ Skins Inventory page'
```

Figura 15 – User Story de Saque de skins e seus cenários