UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ANA BRUNA BELO

COTTONNECT: UM SISTEMA PARA DOAÇÃO DE PRODUTOS COM GRATIFICAÇÃO VIA MOEDA VIRTUAL

PROJETO DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

GUARAPUAVA
2020

ANA BRUNA BELO

COTTONNECT: UM SISTEMA PARA DOAÇÃO DE PRODUTOS COM GRATIFICAÇÃO VIA MOEDA VIRTUAL

Projeto de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Sistemas para Internet – TSI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Dr. Roni Fabio Banaszewski
UTFPR Câmpus Guarapuava

Coorientador: Prof. Dr. Diego Marczal
UTFPR Câmpus Guarapuava

GUARAPUAVA
2020

4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

RESUMO

BELO, Ana Bruna. Cottonnect: Um sistema para doação de produtos com gratificação via moeda virtual. 2020. 39 f. Projeto de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2020.

Atualmente, as discussões sobre questões ambientais vem se tornando cada vez mais frequentes, proporcionando uma reflexão sobre os impactos originados das práticas sociais exercidas. Diante da crescente evidência do impacto causado ao meio ambiente, destaca-se o atual padrão de consumo, muitas vezes praticado inconscientemente, que acaba por contribuir para um maior acúmulo de lixo e mau uso da matéria-prima. Neste contexto, o presente projeto consiste no desenvolvimento de uma aplicação web que visa estimular doações de produtos com gratificações quantificadas por uma moeda virtual, que simboliza a gratidão ao receber uma doação. Desta forma, a intenção é estimular nos indivíduos de diferentes comunidades uma relação mais ampla de troca de produtos, fomentando a concepção do consumo consciente a fim de contribuir com a redução da degradação gradual do meio ambiente.
Palavras-chave: Doações. Consumismo. Impacto ambiental. Desenvolvimento web

ABSTRACT

BELO, Ana Bruna. Cottonnect: A system for donation of products with gratification via virtual currency. 2020. 39 f. Projeto de Trabalho de Conclusão de Curso de graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2020.

Currently, discussions on environmental issues are becoming increasingly frequent, reflecting the impacts originated from the social practices exercised. Given the growing evidence of the impact caused to the environment, we highlight the current pattern of consumption, often practiced unconsciously, which ends up contributing to a greater accumulation of garbage and misuse of the raw material. In this context, this project consists of the development of a web application that aims to stimulate donations of products with bonuses quantified by a virtual currency, which symbolizes gratitude when receiving a donation. Thus, it is intended to stimulate in individuals from different communities a broader relationship of exchange of products, fostering the concept of conscious consumption to contribute to avoid the gradual degradation of the environment.

Keywords: Donations. Consumerism. Environmental impact. Web development

LISTA DE FIGURAS

Figura 1 – Captura de tela da opção de escolha para tipo de anúncio no *alguémquer?*. 5
Figura 2 – Captura de tela inicial do aplicativo *QUEM QUER*. 6
Figura 3 – Captura de tela inicial e de doação do *DOARFÁCIL*. 7
Figura 4 – Imagem da dinâmica do *Scrum*. 15
Figura 5 – Imagem do gráfico *Burndown* aplicado a uma *Sprint*. 16
Figura 6 – Imagem do ciclo do *TDD*. 17
Figura 7 – Captura de tela do quadro do projeto *Cottonnect* no *Trello*. 18
Figura 8 – Imagem do fluxo de trabalho do *Gitflow*. 19
Figura 9 – Imagem do fluxo de trabalho de um projeto utilizando CI/CD. 20
Figura 10 – Imagem da arquitetura da aplicação *Cottonnect*. 26
Figura 11 – Imagem da modelagem do banco de dados da aplicação *Cottonnect*. 28
Figura 12 – Imagem que apresenta a primeira etapa na criação da conta. 29
Figura 13 – Imagem que apresenta a segunda etapa na criação da conta. 30
Figura 14 – Imagem que apresenta a terceira etapa na criação da conta. 30
Figura 15 – Imagem que apresenta a tela de sucesso na criação de uma conta. 31
Figura 16 – Imagem da tela inicial que lista as doações. 32
Figura 17 – Imagem da tela do modal que permite dar um lance para uma doação. . . 32
Figura 18 – Imagem da tela onde é realizado o cadastro de uma doação. 33
Figura 19 – Imagem que apresenta todas as doações de um usuário. 34

LISTA DE QUADROS

Quadro 1 – Comparativos entre os Sistemas. 8
Quadro 2 – Histórias de Usuário 25
Quadro 3 – Cronograma de Atividades. 36

LISTA DE ABREVIATURAS E SIGLAS

RSU	Resíduos Sólidos Urbanos
TDD	Desenvolvimento Orientado por Testes
TCC	Trabalho de Conclusão de Curso
SQL	Structured Query Language (em português Linguagem de Consulta Estruturada)
IDE	Integrated Development Environment (em português Ambiente de Desenvolvimento Integrado)
API	Application Programming Interface (em português Interface de Programação de Aplicações)
PaaS	Platform as a Service (em português Plataforma como Serviço)
ORM	Object Relational Mapper (em português Mapeamento objeto-relacional)
HTTP	Hypertext Transfer Protocol (em português Protocolo de Transferência de Hipertexto)
URL	Uniform Resource Locator (em português Localizador Padrão de Recursos)
CSS	Cascading Style Sheets (em português Folhas de Estilo em Cascatas)
HTML	HyperText Markup Language (em português Linguagem de Marcação de HiperTexto)
UI	User Interface (em português Interface de Usuário)
JWT	JSON Web Token
JSON	JavaScript Object Notation (em português Notação de Objetos JavaScript)
REST	Representational State Transfer (em português Transferência Representacional de Estado)
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
PWA	Progressive Web App (em português Aplicação Web Progressiva)

SUMÁRIO

1 – INTRODUÇÃO 1
1.1 OBJETIVOS 2
1.1.1 Objetivo Geral 2
1.1.2 Objetivos Específicos 2
1.2 ORGANIZAÇÃO DO TRABALHO 3

2 – SISTEMAS SIMILARES 4
2.0.1 *Alguémquer?* 4
2.0.2 *Quem Quer* 5
2.0.3 *DoarFácil* 6
2.0.4 Estudo Comparativo 7

3 – REFERENCIAL TEÓRICO 9
3.1 TECNOLOGIAS DO LADO CLIENTE 9
3.1.1 *HTML*, *CSS* e JavaScript 9
3.1.2 *TypeScript* 10
3.1.3 *React.js* 10
3.1.4 *PWA* 11
3.2 TECNOLOGIAS DO LADO SERVIDOR 11
3.2.1 *Node.js* 12
3.2.2 *REST, JSON e JWT* 12
3.2.3 *ORM* e *TypeORM* 13
3.2.4 *Express.js* 13
3.2.5 *Whatsapp* e *MyZap* 13
3.2.6 *PostgreSQL* 14
3.3 PROCESSO DE DESENVOLVIMENTO 14
3.3.1 Metodologia de desenvolvimento ágil *Scrum* 14
3.3.2 Gráfico *Burndown* 16
3.3.3 *Kanban* 16
3.3.4 Desenvolvimento Orientado por Testes 17
3.4 FERRAMENTAS DE DESENVOLVIMENTO 17
3.4.1 *Trello* 17
3.4.2 *Git*, *GitHub*, *Gitflow* e *GitHub Actions* 18
3.4.3 *Heroku* 20
3.4.4 *Jest* 20
3.4.5 *ESLint* 21

3.5 MECANISMOS DE LEILÕES 21

4 – ANÁLISE E PROJETO DO SISTEMA 22

4.1 ANÁLISE DO SISTEMA 23

4.2 ARQUITETURA DO SISTEMA 26

4.3 MODELAGEM E DESENVOLVIMENTOS DE PROTÓTIPOS 27

4.3.1 Modelagem do Banco de Dados 27

4.3.2 Protótipos das Telas 29

5 – CONCLUSÃO 35

5.1 TRABALHOS FUTUROS 35

6 – CRONOGRAMA 36

Referências 37

1 INTRODUÇÃO

Nos primórdios da civilização humana, a prática da troca direta de objetos e alimentos sem o uso da moeda foi o sistema comercial original e mais básico utilizado para negociações. Mais recentemente, durante o início da colonização portuguesa do Brasil, esta prática de troca de produtos ainda foi amplamente utilizada para negociação com os índios brasileiros. Basicamente, trocava-se produtos excedentes ou desnecessários, por outros de maior interesse ou necessidade, sem a intervenção de instrumentos monetários. Essa prática ficou conhecida como "escambo" ou "permuta" e foi uma forma elementar de comércio utilizada por muito tempo (ROSSETTI, 2013, p. 171).

Em decorrência da necessidade de superar obstáculos encontrados no desenvolvimento do sistema de trocas diretas, como as necessidades em comum e a definição quantitativa em relação ao valor aplicado ao produto, deu-se início à prática do sistema de trocas indiretas, o qual envolve a empregabilidade de uma moeda para intermediar a aquisição de produtos. Desta forma, as relações comerciais foram facilitadas e as trocas, mesmo que de forma indireta, tornaram-se mais eficientes e fáceis de se realizarem (ROSSETTI, 2013, p. 171).

Contudo, com a evolução da indústria e comércio, ocorreu uma diversificação maior da oferta de produtos e também, um aumento da qualidade de vida e do poder aquisitivo da população. Neste sentido, as necessidades básicas como alimentação e vestuário, e de satisfação individual como joias e perfumes aumentaram de forma significante, contribuindo para um consumismo exacerbado ao criar necessidades ou desejos de consumo.

O consumo inconsciente, gera um impacto negativo no meio ambiente. Segundo a WWF-Brasil (2019), o atual padrão de consumo está diretamente relacionado com o processo de degradação do planeta, sendo que cada vez mais consome-se uma quantidade maior de recursos que a natureza consegue oferecer. De fato, relatórios ambientais indicam que no modelo atual de consumo são extraídos mais recursos naturais do que a capacidade do planeta tem de regenerar. Em 29 de julho de 2019, foi considerada a data mais recente desde a primeira vez em que se ultrapassou a capacidade de renovação de recursos naturais (O Dia da Sobrecarga da Terra) (WWF-BRASIL, 2019).

O consumo promovido por meio de impulsos, sem que de fato haja necessidade de compra, torna o bem adquirido rapidamente descartável, contribuindo para o uso da matéria-prima de maneira ineficiente, onde é retirada do ambiente natural para retornar a ele em forma de resíduos. Segundo o relatório anual da ABRELPE (2020), entre os anos 2010 e 2019, a geração de resíduos sólidos urbanos (RSU) no Brasil registrou aumento, passando de 67 para 79 milhões de toneladas por ano. Estima-se que até 2050, o país terá um acréscimo de quase 50% na geração de RSU comparado ao ano de 2019.

O consumo inconsciente acarreta em um maior acúmulo de bens, onde rapidamente estes podem ser substituídos pelas tendências atuais do mercado. Esse fato contribui para uma

maior produção de lixo, em que muitas vezes ocorre o descarte de itens que ainda poderiam ser utilizados, juntamente com o descarte em locais inapropriados. Estima-se que no Brasil, a quantidade de resíduos que têm destinação inadequada aumentou cerca de 30% (ABRELPE, 2020).

A tomada de consciência destes fatos (consumo desnecessário e lixo), juntamente com a percepção de que o atual padrão de consumo está diretamente relacionado com a crise ambiental, tem contribuído para a construção de uma sociedade mais sustentável. (Consumers International et al., 2005, p. 17). De acordo com a pesquisa de perfil dos consumidores brasileiros, realizada por Nielsen (2019), 42% deles informaram que estão mudando seus hábitos de consumo para reduzir o impacto causado no meio ambiente.

Desta forma, com base nas questões ambientais abordadas, buscando soluções a estimular a concepção de consumo consciente e contribuir para redução do descarte de produtos, produção de lixo e mau uso da matéria-prima, em um planeta que dispõe recursos limitados, apresenta-se viável o desenvolvimento de um sistema, que facilite e estimule doações de objetos. No entanto, as doações efetuadas serão convertidas em moedas virtuais, denominadas *flocos de algodão*, que simbolizam o ato de gratidão ao receber uma doação. Os *flocos de algodão* são adquiridos por meio de doações ou através do saldo inicial quando uma nova conta é criada e tem como função exclusiva ser utilizado nos leilões, permitindo a aquisição de novos produtos no sistema, promovendo uma movimentação de produtos via trocas, porém de forma indireta.

Enfim, com o advento desta aplicação computacional, buscar-se-á estimular nos indivíduos de diferentes comunidades uma relação mais ampla de troca de produtos, fomentando a concepção do consumo consciente, de forma a contribuir no combate ao descarte desnecessário, ao consumo inconsciente e irresponsável e à degradação gradual do meio ambiente.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver um sistema para estimular doações de objetos com gratificação via moedas virtuais.

1.1.2 Objetivos Específicos

- Definir um mecanismo de leilão com lances baseados no ranking de moedas virtuais a fim de escolher o participante com maior prioridade em receber uma dada doação de interesse do mesmo;
- Implementar a funcionalidade de "reserva" de um item pelo tempo necessário para o participante vencedor ir analisar fisicamente e retirar o produto;
- Implementar operação de transferência das moedas da conta do recebedor para o doador quando a doação for efetuada;

- Obter informações de localização dos usuários para que as negociações ocorram em uma região próxima a fim de evitar gastos com fretes intermunicipais e principalmente para estimular o contato pessoal e análise do produto antes da aquisição.
- Definir e implementar medidas de segurança a fim de evitar fraudes na aquisição de moedas virtuais.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em cinco capítulos, sendo estes: no Capítulo 2, SISTEMAS SIMILARES, serão abordados os sistemas que possuem objetivo similar ao deste projeto; O Capítulo 3, REFERENCIAL TEÓRICO será apresentado a fundamentação teórica que embasa o trabalho, juntamente com as tecnologias que serão utilizadas no desenvolvimento e na gestão; Por sua vez, o Capítulo 4, ANÁLISE E PROJETO DO SISTEMA, serão apresentadas as análises e considerações a respeito do projeto. Por fim, no Capítulo 5, CONCLUSÃO serão apresentadas as considerações finais acerca do tema e do projeto.

2 SISTEMAS SIMILARES

Atualmente, no mercado são encontradas ferramentas que têm como objetivo a doação direta entre bens de consumo, não havendo como comparação ferramentas que utilizam intermediação monetária como recompensa das doações, onde as moedas são trocadas por produtos dentro do próprio sistema. A seguir serão apresentados alguns sistemas disponíveis no mercado que possuem um propósito similar ao objetivo desse projeto.

Vale ressaltar que no mercado existem alternativas que possibilitam trocas e doações de produtos como Mercado Livre, Olx, Enjoei e *Facebook Marketplace*, porém estes não foram implementadas com o mesmo fim do presente projeto, pensando na retribuição por meio de moedas virtuais das doação para troca posterior por outro produto.

2.0.1 *Alguémquer?*

É um aplicativo, disponibilizado para *Android* e *iOS*, voltado a ser uma rede de doações de objetos, sem restrições de categorias. O aplicativo possibilita duas opções, estas, podendo ser doações ou pedidos de doações, conforme é ilustrado na Figura 1. As doações são criadas por meio da publicação de anúncios, permitindo aos usuários doarem ou receberem objetos de qualquer tipo de categoria, como livros, roupas, eletrônicos, brinquedos e inclui uma categoria que são itens com defeitos. Os anúncios possuem restrição de visualização para usuários que tem a localização cadastrada apenas mesma cidade em que o anuncio foi publicado.

Cada anúncio contém informações do item a ser doado, como título, descrição, categorias, fotos e localização por cidade na qual poderá ocorrer a retirada do objeto, juntamente com um contato do doador, que é o seu número de telefone. O aplicativo implementa filtros, que auxiliam os usuários a encontrarem anúncios de maior preferência, sendo possível a seleção de anúncios por país, estado e cidade, através da localização da conta do usuário, assim como também por categorias a qual os objetos doados pertencem.

O aplicativo possibilita a criação de grupos, deste modo, na publicação de um anúncio para um doação, pode-se definir se ela será publicada sem restrições de visibilidade de usuário ou restringida para determinados grupos que o usuário é membro.

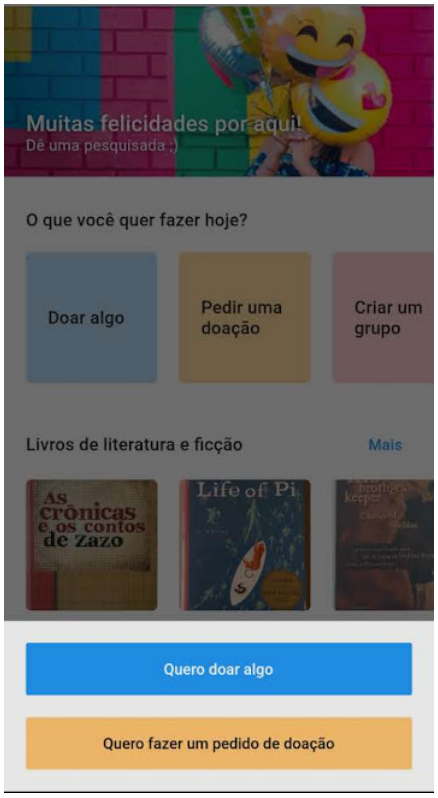


Figura 1 – Captura de tela da opção de escolha para tipo de anúncio no *alguémquer?*.
Fonte: A autora.

2.0.2 *Quem Quer*

Quem Quer é um aplicativo para dispositivos móveis, disponibilizado para plataforma *Android* e *iOS* e tem como função possibilitar doações ou vendas de produtos e serviços. Para acesso ao aplicativo é necessário possuir uma conta, que pode ser criada rapidamente com as credencias do *Facebook* do usuário. Logado ao aplicativo, para realizar doações ou vendas é necessário que o usuário confirme seu número de telefone através de um código enviado por *SMS*.

Com o sucesso na confirmação do número de telefone, o usuário tem disponível a funcionalidade da publicação das suas doações ou vendas, que podem ser criadas através de um anúncio. Os anúncios possuem informações do objeto a ser doado/vendido, como título, descrição, categorias, localização em que o objeto poderá ser retirado e uma data de início e fim que limita o período de visibilidade do anuncio.

Todos os anúncios, antes de serem publicados passam por uma análise de verificação, validando se estão de acordo com as políticas e regras da aplicação. Para comunicação das negociações com os anunciantes, o aplicativo possui bate-papo integrado. Ele também oferece o recurso para salvar os anúncios favoritos e permite que os usuários compartilhem anúncios que não são seus, ajudando no alcance da visibilidade dos objetos de outros usuários.

O aplicativo conta com um sistema de pontuação, em que os pontos são trocados

por destaques nos anúncios, esses destaques aumentam o alcance de visibilidade dos objetos oferecidos. Os pontos podem ser adquiridos anunciando produtos, compartilhando o próprio anuncio, compartilhando o anúncio alheio ou finalizando uma negociação.

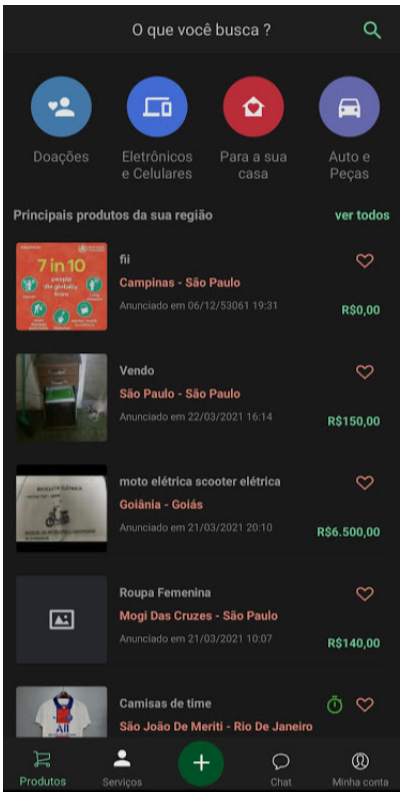
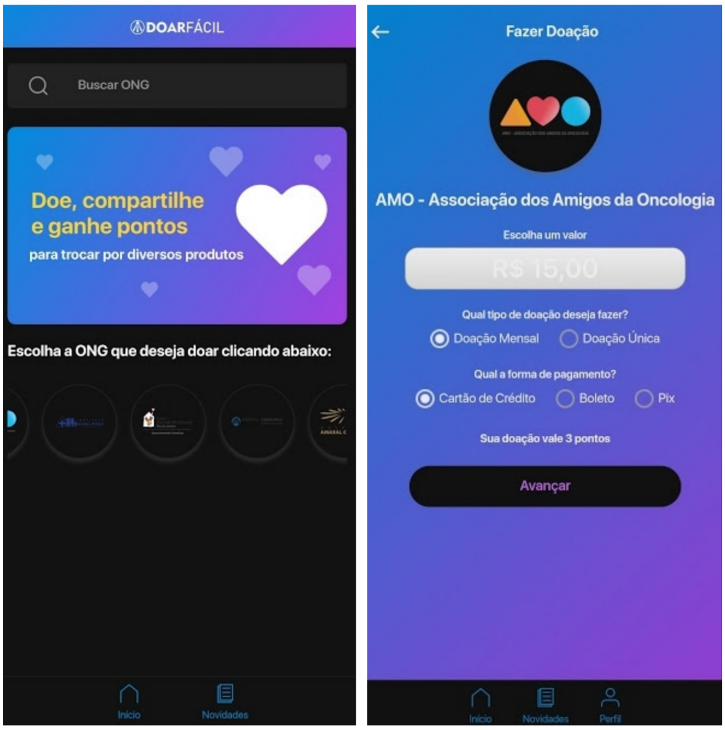


Figura 2 – Captura de tela inicial do aplicativo *QUEM QUER*.
Fonte: A autora.

2.0.3 *DoarFácil*

DoarFácil é uma aplicação que permite realizar doações únicas ou mensais em dinheiro para ONGs de todo o Brasil por meio de um aplicativo disponível para plataforma *Android* e *iOS* ou diretamente pelo site. Como recompensa, cada doação efetuada por um usuário é convertida em pontos que viram prêmios das instituições parceiras.

Para realizar uma doação é necessário estar logado ao sistema, e posterior, escolher uma ONG, conforme a Figura 3a. A etapa seguinte após a escolha da ONG é de fato realizar a doação, onde são inseridas as suas respectivas informações, como valor, método de pagamento e seu tipo, sendo possível doações mensais ou únicas. A Figura 3b, mostra a tela do aplicativo na qual é possível realizar a doação.



(a) Tela Inicial **(b) Tela Doação**

Figura 3 – Captura de tela inicial e de doação do *DOARFÁCIL*.
Fonte: A autora.

2.0.4 Estudo Comparativo

Com base na análise apresentada na Capítulo 2, os aplicativos *Alguémquer?* e *QUEM QUER* possuem como o intuito a doação de objetos entre indivíduos, já o aplicativo *DOARFÁCIL* foi apresentado para especificar a doação, neste caso por meio de dinheiro a ONGs, possuindo uma recompensa em forma de pontos, que posteriormente, podem ser trocados por prêmios.

O sistema proposto no desenvolvimento deste trabalho apresenta a união das características de doações de objetos com o fator de recompensa, tal fator é caracterizado pelas moedas virtuais. Para cada doação realizada, é estabelecido um valor monetário, o qual a pessoa recebedora da doação vai enviar ao doador como forma de agradecimento/gratidão. Consequentemente, as moedas recebidas poderão ser trocadas por outro objeto anunciado no sistema.

O Quadro 2 faz um comparativo entre os sistemas abordados, visando evidenciar as principais diferenças entre os referidos sistemas.

Quadro 1 – Comparativos entre os Sistemas.

	alguémquer?	**quemquer**	**doarfácil**	**cottonnect**
1. Doações de objetos	X	X		X
2. Doações em dinheiro a ONGs			X	
3. Recompensa por doações		X	X	X
4. Movimentação dos produtos através do sistema monetário do sistema				X
5. Trocas indiretas de objetos por meio de moedas				X
6. Doações encontradas por meio de localização	X	X		X
7. Confirmação da conta por meio do número de telefone		X		X
8. Chat Integrado		X		
9. Mobile	X	X	X	

3 REFERENCIAL TEÓRICO

O desenvolvimento de sistemas é uma área multidisciplinar que necessita organização e escolhas adequadas das ferramentas para ser bem sucedida. Desta forma, fazer uso de um *framework* de gerenciamento de projetos como o *Scrum* e seus artefatos relacionados, que auxiliem na organização dos processos, assim como nas atividades a serem desenvolvidas, é essencial para se ter sucesso no desenvolvimento de sistemas. Outrossim, o uso de práticas ágeis pode auxiliar no atendimento de requisitos e nas entregas das funcionalidades implementadas no tempo mais próximo possível do esperado.

Quando refere-se ao desenvolvimento de sistemas para web, são inúmeras as ferramentas disponíveis no mercado, desde linguagens de programação, *frameworks*, e bibliotecas que auxiliam os programadores no desenvolvimento de suas aplicações. Tanto no lado do cliente como no lado do servidor, a escolha dessas tecnologias impacta diretamente no processo de desenvolvimento, na qualidade e no tempo de desenvolvimento. Portanto, definir quais tecnologias serão utilizadas é uma parte que possui grande importância no sucesso de uma aplicação.

Assim como a definição de tecnologias adequadas, é importante manter boas práticas de programação e fazer uso de técnicas que auxiliem na manutenção e na integridade da aplicação, como práticas de implementação de teste e *TDD*, além de validadores da qualidade do código, como o *ESLint, JSHint, Prettier, RuboCop*, entre outros.

Nesse capítulo, algumas características e particularidades de uso das tecnologias mencionadas acima que contribuirão para o desenvolvimento do projeto serão apresentadas.

3.1 TECNOLOGIAS DO LADO CLIENTE

O lado cliente, ou *front-end*, pode ser classificado como a parte visual de uma aplicação. Por meio dela torna-se possível a iteração entre um usuário e o sistema. Nesta seção, serão abordadas algumas tecnologias de desenvolvimento de aplicações que são utilizadas no lado cliente.

3.1.1 *HTML*, *CSS* e JavaScript

HTML (*HyperText Markup Language*) é uma linguagem para marcação de hipertexto. De maneira sucinta, pode-se definir hipertexto como todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos (SILVA, 2011). Basicamente, o *HTML* é utilizado para a construção da estrutura das páginas web. Ele consiste em uma série de elementos que identificam cada parte da página, informando ao navegador como exibir o conteúdo, podendo corresponder a títulos, parágrafos, imagens, etc (W3SCHOOLS, 2021).

Como o *HTML* tem seu uso exclusivo na construção das estruturas que formam uma pagina na web, é necessário fazer uso de outra tecnologia que permite a estilização dessas páginas. Para este fim, é utilizado a linguagem de estilo *CSS*. *CSS* é a abreviação para o termo em inglês *Cascading Style Sheet*, traduzido para o português como folhas de estilo em cascata e cabe a ele todas as funções de estilo de um documento (SILVA, 2011). *CSS* é uma linguagem de estilo utilizada para definir a aparência de uma página, descrevendo como os elementos *HTML* devem ser exibidos em tela, incluindo o *design, layout*, cores e variações de exibição para diferentes dispositivos e tamanhos de tela.

Como os elementos *HTML* são construídos com conteúdo estáticos, ou seja, não possibilitam iteração com funcionalidades dinâmicas, é necessário fazer uso de uma linguagem que permita a criação de conteúdos que se atualizam de forma dinâmica, dando "vida" às aplicações que antes eram apenas estruturadas estática. O *JavaScript* é uma linguagem que pertence a essa tríade do desenvolvimento *front-end*, onde permite realizar funcionalidades dinâmicas na construção de páginas da web. *JavaScript* é uma linguagem de programação da Web, utilizada na ampla maioria dos sites, navegadores, computadores de mesa, consoles de jogos, *tablets* e *smartphones*, tornando-a a linguagem de programação mais onipresente da história. Pode ser utilizada no lado cliente, dando "movimento" as páginas, assim como pode ser utilizada no lado servidor, na construção de sistemas *back-end*, desde *APIs*, *webservices*, até servidores de aplicação. *JavaScript* é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais (FLANAGAN, 2013).

3.1.2 *TypeScript*

TypeScript é uma linguagem de código aberto com base na linguagem *JavaScript*, oferecendo entre outras características, a possibilidade de definir tipos de dados. O código *TypeScript* é transformado em código *JavaScript* por meio do compilador *TypeScript* ou Babel. Este código compilado pode ser executado em qualquer ambiente que o *JavaScript* possa ser executado (TYPESCRIPT, 2021).

A linguagem TypeScript pode ser considerada um superconjunto tipado de *JavaScript*, que é compilado para *JavaScript* simples. Isto faz programas escritos em *TypeScript* altamente portáteis, pois podem ser executados em quase qualquer máquina. A verificação de tipo estático reduz erros causados pelo uso indevido acidental de tipos, assim como permite que as ferramentas de desenvolvimento forneçam mecanismos de autocompletar inteligente, aumentando o nível de produtividades dos programadores (FENTON, 2017).

3.1.3 *React.js*

React.js é frequentemente referido como *ReactJS* ou apenas *React*. O *React.js* é uma biblioteca *JavaScript* de código aberto, criada pela equipe do *Facebook* para construir interfaces de usuário (UI - *User Interface*). Seu desenvolvimento é baseado em componentes, isso permite

a divisão das UI em partes independentes, reutilizáveis e isoladas, onde cada componente possui estados. Mesmo não sendo obrigatório, o *React* recomenda o uso do *JSX* para criar elementos para utilização nos *templates*. O *JSX* é uma extensão de sintaxe para *JavaScript*, em que pode lembrar uma linguagem de *template*, como *HTML*, mas vem com todo o poder do *JavaScript* (REACTJS, 2021). O *React* faz uso do paradigma de programação declarativa, que concentra-se em descrever o que deve-se fazer e não em como seus procedimentos funcionam.

Para manipulação da DOM *(Document Object Model)*, que é a representações dos dados que compõem a estrutura e o conteúdo de uma pagina na *Web*, ele faz uso de um DOM virtual. Isso permite atualizar e renderizar apenas os componentes necessários na medida em que os dados mudam, permitindo um desempenho mais eficiente (HAMEDANI, 2021).

3.1.4 *PWA*

PWA (Progressive Web App) é um modelo de aplicação web que oferece tecnologias que permitem a experiência de uso muito próxima da oferecida pelos aplicativos móveis. A principal vantagem em utiliza-lo é a de não ser necessário a instalação da aplicação nos dispositivos, as aplicações são leves e sua construção é relativamente simples quando o sistema web segue os padrões requeridos para ser um PWA. As características de um PWA são: ser progressivo, responsivo, independente de conectividade, semelhante a aplicativos nativos, estar sempre atualizado, ser seguro devido ao HTTPS, descobrível, envolvente por utilizar recursos em que é possível engajar com o usuário, simular o estado de instalado ao inserir um ícone da aplicação nas áreas de aplicativos dos dispositivos e permitir acesso e compartilhamento através de links (ANTUNES, 2021).

Existem algumas tecnologias Web por trás do conceito de PWA, que são as seguintes: Service Worker, Cache API e Web App Manifest. Basicamente, o Service Worker é que torna a tecnologia de PWA possível. Ele intercepta as requisições da aplicação e guarda os resultados no lado do cliente, possibilitando um consumo dos dados extremamente veloz, e permitindo que a aplicação funcione offline. Por sua vez, o Cache API é uma forma de se armazenar as requisições feitas localmente e por fim, o Web App Manifest é um documento que tem como finalidade padronizar as aplicações Web (ANTUNES, 2021).

3.2 TECNOLOGIAS DO LADO SERVIDOR

O lado servidor, ou *back-end*, é responsável por receber requisições e realizar processamentos que estão diretamente ligadas as regras de negócios, como persistir dados, obter informações, realizar cálculos, entre outras. O lado servidor está diretamente interligado com o lado cliente, o qual requisita informações a ele e aguarda o processamento dessas requisições. Por sua vez, o lado servidor deverá devolver o resultado dessas informações para o cliente que as requisitou. Nesta seção, serão abordadas as tecnologias de desenvolvimento de aplicações que tem seu uso no lado servidor.

3.2.1 *Node.js*

Node.js é um ambiente de execução *JavaScript* assíncrono orientado a eventos. Ele foi construído com base na *engine V8*. O *engine V8* é um interpretador de código *JavaScript* desenvolvido pelo *Google* que executa a linguagem em navegadores. Desta forma, o uso do *Node.js* permite que a execução da linguagem *JavaScript* seja possível fora dos navegadores, tornando o uso da linguagem viável tanto no lado do cliente quanto no lado do servidor.

Sua execução é *single-thread*, diferente das outras tecnologias disponíveis no mercado, ou seja, no modelo *Node.js* apenas uma *thread* é responsável por tratar as requisições (NODE, 2021).

3.2.2 *REST, JSON e JWT*

REST (*Representational State Transfer*) é um conjunto de princípios de arquitetura de software para a criação de *web services*. O *REST* teve origem na tese de doutorado de Roy Fielding, e visa desempenho de interação, simplicidade, modificabilidade, portabilidade e confiabilidade (FIELDING, 2000). O protocolo *REST* é baseado nos conceitos do protocolo HTTP, que é o protocolo de camada de aplicação para transmissão de documentos hipermídia, como o *HTML*.

Basicamente, o *REST* é guiado pelo que seriam as boas práticas de uso do protocolo HTTP. Estas boas práticas caracterizam-se pelo uso adequado dos métodos HTTP (*GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT*), usando-os de maneira a respeitar o objetivo original para o qual realmente foram criados. Conforme as práticas do *REST*, as *URLs* escolhidas devem ser padronizadas dentro do sistema e devem ser únicas por recurso. Também, o uso de códigos de status devem ser padronizados para representação de sucessos ou falhas e, os cabeçalhos HTTP devem ser usados de forma adequada para atender o objetivo no qual foram realmente criados. Desta forma, esta arquitetura permite a criação de sistemas distribuídos, eficientes, confiáveis e escaláveis (SAUDATE, 2013).

Por sua vez, o JSON (*JavaScript Object Notation*) é um formato para armazenamento e transmissão de informações em texto, leve, fácil para desenvolvedores lerem e escreverem e fácil para as máquinas analisarem. É baseado em um subconjunto do padrão de linguagem de programação JavaScript (especificada na ECMA-262) e é um formato de texto completamente independente da linguagem. A representação do formato é composto por nome/valor, ou seja, para cada valor representado, atribui-se um nome que descreve o seu significado (JSON.ORG, 2021).

Por fim, o *JSON Web Token (JWT)* é um padrão aberto que define uma forma compacta e independente para transmitir informações com segurança entre aplicações. Na autenticação, quando o usuário efetua login, um *JSON Web Token* é retornado, e sempre que o usuário requisitar acesso à uma rota ou recurso protegido, o agente do usuário deve enviar esse *Token*. Desta forma, se o *Token* estiver correto, o recurso é liberado (JWT, 2021).

3.2.3 *ORM* e *TypeORM*

Com o utilização da orientação a objetos, surge uma técnica de programação denominada *ORM (Object Relational Mapping)*. O *ORM* consiste na persistência automatizada (e transparente) de dados que representa como objetos as tabelas de um banco de dados relacional, utilizando metadados que descrevem o mapeamento entre os objetos e banco de dados. A utilização de um *ORM* permite um aumento de produtividade, fazendo com que os desenvolvedores se concentrem nos problemas de negócios e facilitando a manutenção nos projetos, tornando-os mais compreensíveis e portáveis para outros sistemas de bancos de dados (BAUER; KING, 2005).

TypeORM é um *ORM* que pode ser executado em diversas plataformas e pode ser usado com as linguagens *TypeScript e JavaScript*. Seu objetivo é fornecer recursos adicionais que ajudem a desenvolver qualquer tipo de aplicativo que use bancos de dados, através da técnica de mapeamento objeto relacional (TYPEORM, 2021).

3.2.4 *Express.js*

Express é um *framework* web de roteamento e *middleware* para aplicações que usam *Node.js* e tem como a funcionalidade mínima fornecer uma série de chamadas de funções de *middleware*. As funções de *middleware* são funções que tem acesso ao objeto de solicitação (*request*), ao objeto de resposta (response) e a próxima função de *middleware* no ciclo solicitação-resposta do aplicativo. O *Express.js* permite a manipulação de roteamento, determinando como um aplicativo responde a uma solicitação do cliente, que é requisitada por meio de uma caminho e um método de solicitação HTTP específico (e.g. GET, POST, etc) (EXPRESS, 2021).

O *Express* vem com um manipulador de erros integrado, que cuida de qualquer erro que possa ser encontrado no aplicativo. Ele permite a definição de funções de *middleware* de manipulação personalizada de erros. Estas possuem quatro argumentos, a saber: *err, req, res, next*. Ele também permite realizar configurações comuns de aplicação web, como a porta a ser usada para conexão e a localização dos modelos que são usados para renderizar a resposta (EXPRESS, 2021).

3.2.5 *Whatsapp* e *MyZap*

Whatsapp é um aplicativo gratuito multiplataforma que oferece serviços de mensagens e chamadas e possibilita o envio e recebimento de diversos arquivos de mídia como textos, fotos, vídeos, documentos e localização, além de chamadas de voz (WHATSAPP, 2021).

Por sua vez, *MyZap* é uma *API* construída em *Node.js e Express*, gratuita e de código aberto que possibilita a automação de envio de mensagens para contas no *WhatsApp*. Ela usa como base o *Venom-bo*, utilizado para criar *bots* para *WhatsApp*. Ele funciona como um navegador virtual sem interface gráfica que abre o *Whatsapp* web e executa todos os comandos

via código, possibilitando assim a automação de todas as funções para utilizar o *Whatsapp* (BARSCH, 2021).

3.2.6 *PostgreSQL*

PostgreSQL é um sistema gerenciador de banco de dados (SGBD) objeto relacional, gratuito e de código aberto com mais de 30 anos de desenvolvimento ativo, utilizado no armazenamento de informações (POSTGREESQL.ORG, 2021). Além de controlar o armazenamento de dados, um *SGBD* deve ser responsável por controlar o acesso ao banco de dados, gerenciando quem pode visualizar e alterar os dados. O *PostgreSQL* como sendo um *(SGBD)*, oferece estas funções.

O *PostgreSQL* usa e estende a linguagem *SQL*, que é a linguagem de pesquisa declarativa padrão utilizada para os banco de dados relacionais. O PostgreSQL é executado em todos os principais sistemas operacionais e é compatível com o conjunto de propriedades de transação *ACID*, garantindo atomicidade, consistência, isolamento e durabilidade. Assim como os demais *(SGBDs)*, ele oferece recursos necessários para realizar replicações em servidores. Ele também permite gerenciar várias conexões com o banco de dados ao mesmo tempo, por meio do recurso de *multithread* oferecidos pelos sistemas operacionais. Ainda, tem suporte nativo à *SSL*, que é um protocolo de segurança projetado para fornecer segurança nas comunicações em uma rede de computadores. E ainda vale mencionar que ele também suporta de forma eficiente e confiável grandes tamanhos de dados em suas tabelas. (MILANI, 2008).

3.3 PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento de uma aplicação constitui todo o período em que a aplicação estará em desenvolvimento. Neste período é importante fazer uso de práticas e metodologias que auxiliam a gerenciar e organizar todo processo, principalmente em projetos que são desenvolvidos em equipes. Ao se trabalhar em equipes, há necessidade de uma maior organização para que todos os membros saibam exatamente o que é necessário fazer, e como resultado final, possam entregar produtos com qualidade e dentro dos prazos estipulados. Neste sentido, a corrente seção apresenta algumas práticas e metodologias que auxiliam no processo de gerenciamento, organização e desenvolvimento de uma aplicação.

3.3.1 Metodologia de desenvolvimento ágil *Scrum*

O *Scrum* é um *framework* estrutural que vem sendo usado desde o início de 1990. Seu principal evento são as Sprints, elas podem ser determinadas por um espaço de tempo (de uma a duas semanas, geralmente) em que uma versão potencialmente utilizável do produto irá ser criada. A metodologia *Scrum* define também um Time *Scrum*, o qual é composto pelo *Product Owner*, um *Scrum Master* e o Time de Desenvolvimento. O *Product Owner* é o responsável por fornecer novos requisitos juntamente com a ordem que devem ser executados, além de

avaliar a qualidade final das entregas. O *Scrum Master* é responsável por garantir que o *Scrum* seja entendido e aplicado. O Time de Desenvolvimento consiste nos profissionais que realizam o trabalho de entregar uma versão usável que incrementa o produto ao final de cada *Sprint*. A Figura 4 apresenta a dinâmica do processo do *Scrum*. Para fornecer transparência ao processo, assegurando que todas as informações sejam transmitidas às equipes e que todos tenham o mesmo entendimento, são aplicados alguns artefatos (SCHWABER; SUTHERLAND, 2013). Segundo Schwaber e Sutherland (2013), os artefatos do *Scrum* são:

- **Product Backlog:** é uma lista de tudo que deve ser necessário no produto, e é a única origem dos requisitos para qualquer mudança a ser feita no produto.
- **Sprint Backlog:** é um conjunto de itens do *Backlog* do Produto selecionados para uma dada *Sprint*, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da *Sprint*.

Na composição da lista do *Product Backlog* é comum utilizar para descrever os itens que devem ser realizados as Histórias de Usuário. Uma História de Usuário descreve de forma natural e informal uma funcionalidade que será valiosa para um usuário de um sistema. Uma História pode utilizar um *template* que pode ser descrito da seguinte forma: COMO ator QUERO algo PARA um motivo (COHN, 2004), onde primeiramente define-se o usuário que executará tal funcionalidade, seguido do que ele quer realizar e qual o objetivo a atingir com a implementação. O uso das Histórias de Usuário facilita a criação de sentido e a comunicação no que deve ser desenvolvido, ajudando as equipes na compreensão do sistema.

Para auxiliar na gestão de tempo e divisão das *Sprints*, é possível definir pontos às História de Usuário. Os pontos são unidades de medida utilizados para expressar uma estimativa do esforço necessário para implementar uma funcionalidade requerida. Desta forma, os desenvolvedores com base em complexidade e quantidade de trabalho, atribuem pontos às Histórias. Com a quantidade de pontos, pode-se definir quais e quantas Histórias podem fazer parte de cada *Sprint*.

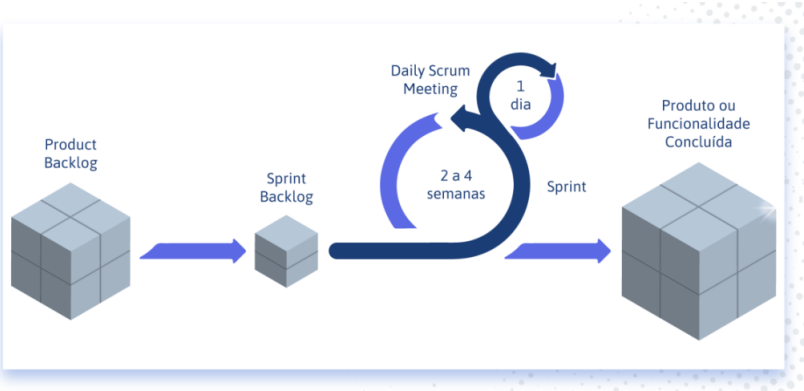


Figura 4 – Imagem da dinâmica do *Scrum*.
Fonte: (JUSTO, 2017).

3.3.2 Gráfico *Burndown*

O gráfico de *Burndown* é uma ferramenta utilizada pelas equipes *Scrum* para apresentar o trabalho concluído por dia em relação ao valor de conclusão estipulado para uma Sprint. Ele é composto por dois eixos: Y, que vai representar o trabalho que precisa ser realizado, e X, que representa o tempo ou quantidade de trabalho, estipulada em dias ou horas, para concluir a demanda. Um exemplo do gráfico pode ser observado na Figura 5. Basicamente, o gráfico inicialmente começa com um total de pontos planejados na *Sprint* e a medida que as tarefas são executadas, a pontuação no gráfico vai sendo "queimada". No final do último dia de duração, espera-se que tenha um total de 0 pontos planejados, caracterizando que todas as tarefas definidas na *Sprint* foram realizadas. Desta forma, o gráfico irá ajudar a equipe a visualizar o progresso na conclusão de uma *Sprint*, sendo possível verificar se a equipe está adiantada, dentro do cronograma ou em atraso (CAMARGO, 2020).

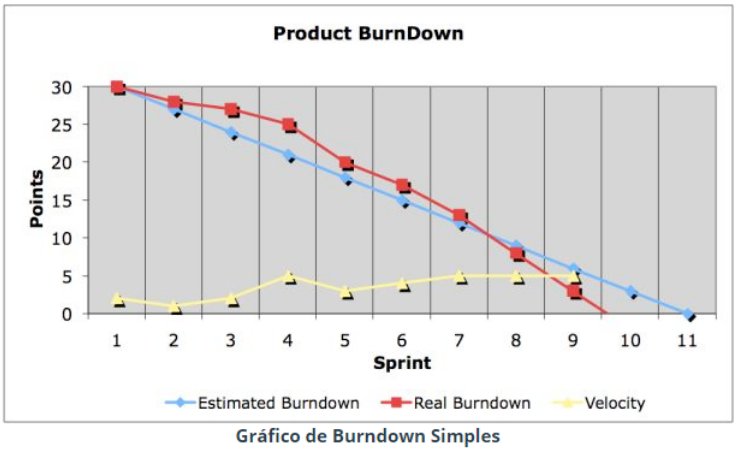


Figura 5 – Imagem do gráfico *Burndown* aplicado a uma *Sprint*.
Fonte: (CAMARGO, 2020).

3.3.3 *Kanban*

Kanban é um sistema ágil e visual para controle de produção ou gestão de tarefas. Ele possui 3 (três) principais componentes, os quais são denominados de: cartões, colunas e quadros. O cartão representa uma tarefa ou ação que precisa ser tomada para que o resultado final seja entregue. Por sua vez, as colunas representam os status dos cartões que geralmente são representados por três colunas: A Fazer, Em Execução e Feito. Porém, estas podem ser alteradas de acordo com a necessidade do projeto. Por fim, o quadro é o *Kanban* como um todo, organizado em colunas e cartões. Desta forma, a organização se dá pela movimentação dos cartões pelas colunas, indicando o seu atual estado no momento. Esses componentes podem ser visualizados na Figura 7. A união entre o modelo visual apresentado pelo *Kanban* e os métodos definidos pelo *Scrum* trazem vantagens as equipes, permitindo que as histórias de usuário sejam organizadas em cartões e as Sprints definidas em colunas, facilitando a visualização dos

riscos do projeto. Assim, o *Product Owner* e o Time de Desenvolvimento podem visualizar possíveis gargalos que estão atrasando entregas e indicando a produtividade individual de cada membro, já que cada cartão entregue pode ter um responsável definido. (ARTIA, 2021).

3.3.4 Desenvolvimento Orientado por Testes

Do inglês *Test Driven Development*, ou simplesmente TDD, é uma prática na qual se escrevem os testes antes mesmo de escrever o código de produção. Ao escrever os testes, o desenvolvedor garante que boa parte do sistema tem um teste que assegura o seu funcionamento. Desta forma, consegue-se verificar a integridade do sistema quando novas funcionalidades são adicionadas. O mecanismo de testes consiste em escrever um teste que falha e em seguida o fazer passar da maneira mais simples possível e, por fim, refatorar o código (ANICHE, 2014). Esse ciclo é conhecido como Ciclo Vermelho-Verde-Refatora e pode ser observado na Figura 6.

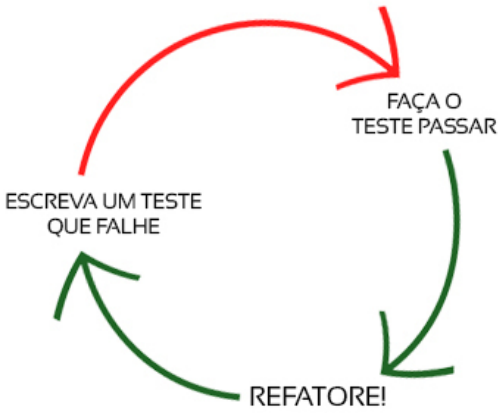


Figura 6 – Imagem do ciclo do *TDD*.
Fonte: (ANICHE, 2014).

3.4 FERRAMENTAS DE DESENVOLVIMENTO

As ferramentas de desenvolvimento são requisitos crucias para auxiliar os desenvolvedores na construção de sistemas computacionais. Com o uso de ferramentas adequadas, pode-se aumentar a produtividade do desenvolvimento, minimizando a necessidade de efetuar práticas manuais, assim como permitir uma organização e qualidade no sistema à ser desenvolvido. Nesta seção, serão apresentadas algumas ferramentas que auxiliam na construção de aplicações, ajudando na manutenção, organização e melhores práticas de desenvolvimento.

3.4.1 *Trello*

O *Trello* é uma ferramenta de gerenciamento de projetos online. Cada projeto criado é denominado de quadro e pode ser compartilhado com vários usuários. Cada quadro possui

colunas que contêm cartões. Os cartões são os elementos essenciais para a organização do trabalho em um quadro, utilizados para representar tarefas e histórias do usuário das metodologias ágeis. O *Trello* permite integrar ferramentas de trabalho, conectando *power-ups* que ajudam a atender a uma necessidade específica. Como exemplo, há o *power-up Burndown Chat*, que permite visualizar gráficos de *Burndown* para as *Sprints* e o *Card Priority by Screenful* que permite adicionar prioridades as Histórias de Usuário. Também há a possibilidade de uso de extensões, tal como a *Scrum for Trello*, que permite trabalhar com as práticas do *Scrum* e *Card Numbers for Trello* que permite adicionar números de identificação (*IDs*) nos cartões do quadro (TRELLO, 2021). A Figura 7 apresenta a tela da organização de um quadro.

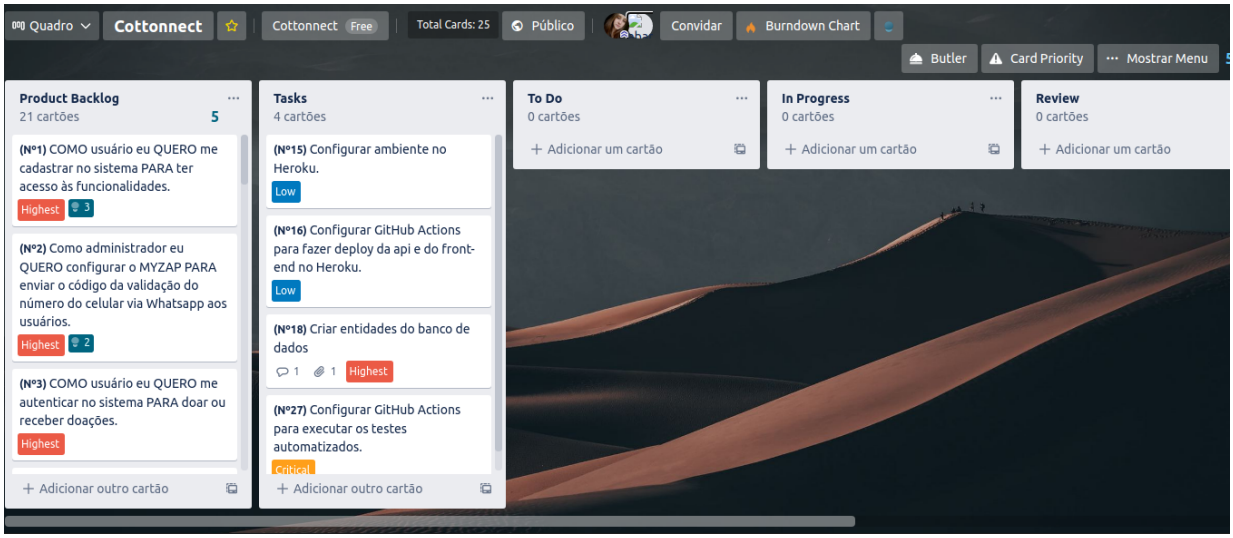


Figura 7 – Captura de tela do quadro do projeto *Cottonnect* no *Trello*.
Fonte: A autora.

3.4.2 *Git*, *GitHub*, *Gitflow* e *GitHub Actions*

Git é um sistema de controle de versão de arquivos, distribuído, gratuito e de código aberto. Ele foi projetado para hospedar desde projetos pequenos a projetos grandes mantendo a mesma velocidade e eficiência (GIT, 2021). Para gerenciar o fluxo de trabalho no *Git*, pode-se seguir um modelo de trabalho denominado *Gitflow*. O *Gitflow* é uma ideia abstrata do fluxo de trabalho do *Git*, em que define um modelo de ramificação rigoroso projetado com base no lançamento do projeto. Em termos técnicos, o *Gitflow* define um fluxo de trabalho para gerenciar as *branches*, que são ramificações do projeto nos repositórios.

Em vez da única *branch* principal, este fluxo de trabalho usa duas ramificações para registrar o histórico do projeto. A *branch* principal (chamada de main) armazena o histórico do lançamento oficial e a ramificação de desenvolvimento (chamada de develop) contem o código para a próxima versão de lançamento. Cada nova funcionalidade deve residir na própria ramificação, que deve ser criada a partir da *branch* de desenvolvimento. Quando uma funcionalidade é concluída, ela é mesclado na ramificação de desenvolvimento e quando a

ramificação de desenvolvimento adquiriu recursos o suficiente para um lançamento, ela é bifurcada em uma ramificação de lançamento (chamada de release). Quando estiver pronta para ser lançada, essa ramificação de lançamento é mesclada com o *branch* principal e de desenvolvimento. Para corrigir com rapidez problemas que ocorrem na versão que está em produção, são criadas as ramificações de manutenção ou de *"hotfix"*, em que são baseadas a partir da *branch* principal, uma vez que esta branch reflete o código em produção (ATLASSIAN, 2021). Pode-se visualizar o fluxo de trabalho do *Gitflow* na Figura 8

O *GitHub* é uma plataforma online de código aberto que permite a hospedagem de projetos gratuitamente com repositórios públicos e privados de forma ilimitada. Em conjunto, ele oferece uma série de funcionalidades que ajudam na revisão e gerenciamento dos projetos (GITHUB, 2021).

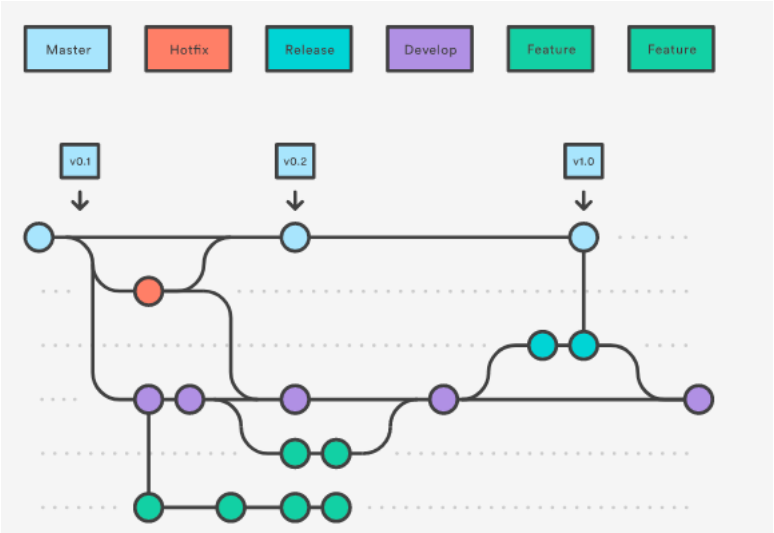


Figura 8 – Imagem do fluxo de trabalho do *Gitflow*.
Fonte: (ATLASSIAN, 2021).

Por sua vez, o *GitHub Actions* é utilizado para automatizar, personalizar e executar fluxos de trabalho de desenvolvimento em integração contínua (CI) e implantação contínua (CD), diretamente em projetos que estão hospedados em repositórios do *GitHub*. O CI que usa *GitHub Actions* oferece fluxos de trabalho que podem ser configurados quando ocorre um evento no *GitHub* (por exemplo, quando um novo código é enviado para o repositório), desta forma ele pode executar os testes da aplicação (GitHub Actions, 2021). Já o CD, se refere ao lançamento automático das mudanças do repositório à produção, que podem ser configuradas através de uma mesclagem a *branch* principal, por exemplo. Na Figura 9 é possível visualizar um fluxo de trabalho utilizando os conceitos de CI/CD.

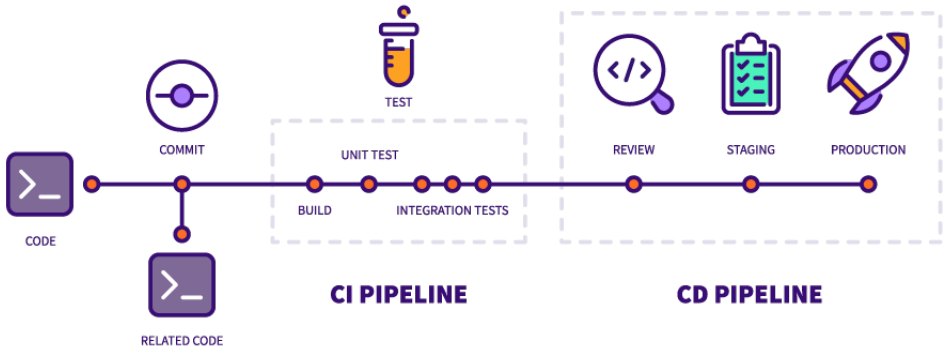


Figura 9 – Imagem do fluxo de trabalho de um projeto utilizando CI/CD.
Fonte: (GITLAB, 2021).

3.4.3 *Heroku*

Heroku é uma plataforma como serviço (PaaS) em nuvem baseada em contêiner, ou seja, oferece serviços de hospedagem de aplicações. É utilizado para implantar, gerenciar e dimensionar aplicativos. É flexível e de fácil utilização, oferecendo aos desenvolvedores o caminho mais simples para colocar os aplicativos em produção. *Heroku* é totalmente gerenciável, dando aos desenvolvedores a liberdade de se concentrar em seu produto sem a distração de manter servidores, hardware ou infraestrutura. Ele também fornece serviços, ferramentas, fluxos de trabalho, projetados para aumentar a produtividade do desenvolvedor HEROKU (2021b).

O *Heroku* oferece o recurso *Heroku Developer Experience* que é uma abordagem para entrega de software em modo produção, para que os desenvolvedores possam se concentrar na criação e entrega contínua de aplicativos, sem se distrair com servidores ou infraestrutura. Os desenvolvedores implantam as aplicações diretamente de ferramentas populares como *Git*, *GitHub* ou sistemas de integração contínua (CI) HEROKU (2021a).

3.4.4 *Jest*

Jest é um *Framework* de testes projetado para garantir a correção de qualquer código *Javascript* (JEST, 2021). Foi criado inicialmente para testar aplicações que foram construídas com *React*, porém sua implementação se tornou muito mais ampla, sendo atualmente utilizado também em aplicações que utilizam *Node.js* e *Redux*, assim como em plataformas que utilizam *TypeScript* como *Angular* e *Ionic*.

É caracterizado por ser de fácil utilização, tendo "zero configuração" na maioria dos projetos, possuindo agilidade, devido a execução paralela dos testes de forma confiável. Ele também possui a opção de gerar relatórios de cobertura de código e fornece um contexto rico do motivo dos testes terem falhados, como: *toBe*, *toBeCloseTo*, *toEqual*, *toStrictEqual*, *toHaveProperty*, *toMatchSnapshot* e *toThrowError* (JEST, 2021).

3.4.5 *ESLint*

ESLint é um utilitário *linting* para a linguagem *JavaScript*, de código aberto que realiza análise estática de códigos, frequentemente utilizada para encontrar padrões problemáticos ou códigos que não obedecem a certas regras de estilo (ESLINT, 2021). A análise estática é o processo que tem como objetivo produzir uma lista de erros, advertências e pontos de melhoria no código de um projeto. Assim, o *ESLint* permite que desenvolvedores descubram problemas ou apliquem melhorias que ajudam na legibilidade do código e estimulam boas práticas de programação. Ele permite que os desenvolvedores criem suas próprias regras de *linting* e é projetado para ter todas as regras completamente plugáveis. Ele também pode ser executado como parte de um *pipeline* de uma integração contínua (ESLINT, 2021).

3.5 MECANISMOS DE LEILÕES

Um leilão é um mecanismo que tem como objetivo promover a aquisição competitiva de produtos entre pessoas, que em um leilão são denominadas licitantes. Leilões são mecanismos dinâmicos e eficientes utilizados na comercialização de bens em mercados, principalmente quando não existe uma referência estável de preço. Os produtos são adquiridos por meio de um conjunto de regras pré-definidas conhecidas pelos licitantes antes do inicio do leilão e um critério de avaliação preciso especificado pelo leiloeiro, que é a pessoa responsável por mediar os eventos de leilão, oferecendo os bens que serão colocados à venda e recebendo as ofertas. Um leilão é um mecanismo transparente que deve se alcançar por meio de um resultado justo, reduzindo as oportunidades de corrupção (MAURER; BARROSO, 2011).

Nos últimos anos, houve um grande aumento no destaque e variedade de leilões na economia, permitindo a existência de vários tipos de leilão, sendo de múltiplos ou único itens, preços crescente ou decrescentes, lances abertos ou fechados, etc. Para Ausubel (2003), os principais tipos de leilão de um único item são os seguintes:

- **Leilões com lances fechados** É considerado um leilão estático, onde são enviados os lances sem saber os lances dos oponentes. Depois de acabar o prazo do envio, o leiloeiro abre os lances e define o ganhador. Podem ser considerados leilões de lances fechados o leilão de primeiro preço e o leilão de segundo preço. No leilão de primeiro preço, o maior lance ganha o item e paga exatamente o valor do seu lance. No leilão de segundo preço, o maior lance ganha o item, mas paga o valor do segundo maior lance.
- **Leilões dinâmicos de um único item** Os participantes fazem seus lances e com isso aprendem uma estratégia a partir dos lances de seus oponentes durante o leilão. Podem ser considerados leilões dinâmicos de um único item o leilão inglês e o leilão holandês. No leilão inglês, os licitantes submetem sucessivamente lances mais altos para o item, onde o licitante final ganha o item. No leilão holandês, o leiloeiro começa com um preço alto e sucessivamente anuncia preços mais baixos até alguém demostrar interesse no item, ganha quem dar o primeiro lance no item.

4 ANÁLISE E PROJETO DO SISTEMA

O sistema proposto é caracterizado por ser um sistema web que posteriormente será evoluído para conformidade com a tecnologia *PWA*. Basicamente, o sistema tem como objetivo permitir que usuários anunciem doações de objetos em leilões virtuais para que outros usuários possam dar lances com moedas virtuais para obter o respectivo objeto.

O tipo de leilão escolhido para uso no sistema é o leilão de primeiro preço. A escolha deste modelo se deu pela quantificação do valor que o objeto possui para o usuário que demostra interesse nele. Assim, segundo a análise do participante, ele irá definir o preço que julga mais justo para adquirir o produto e recompensar o doador sem a visualização dos lances dos adversários. Esta omissão evita influência na definição do valor do lance. Desta forma, os usuários participantes poderão dar um lance único por leilão, e ao final do mesmo, o vencedor será o que deu o maior lance. Em caso de empate, ganhará o participante que possuir mais moedas no sistema. A justificativa para uso desta estratégia é a de que o participante com mais moedas supostamente contribuiu com mais doações. Por fim, se novamente houver empate, o ganhador será o que deu o lance primeiro.

Para os lances com moedas virtuais, será utilizado um sistema monetário próprio, definido como flocos de algodão. Um floco de algodão simboliza o ato de agradecimento por uma doação, por ser leve, macio e remeter a sensações agradáveis. Basicamente, a função desta moeda virtual é intermediar as doações, ou seja, quando uma doação é efetuada, o doador receberá sua gratificação em flocos de algodão e este pode enviar lances para requisitar uma doação de outro item.

O nome da aplicação a ser desenvolvida é *Cottonnect*. Este nome é originado da junção das palavras *Cotton* (Algodão) e *Connect* (conectar), trazendo o significado da aplicação ser uma comunidade de doações conectada pelos flocos de algodão, pois é por meio deles que as doações são possíveis.

Uma particularidade que vale salientar é a que envolve a criação da conta e caixa inicial de um usuário. Cada nova conta criada será iniciada com uma quantidade de flocos de algodão configurável pelo administrador do sistema, para que novos usuários comecem a interagir com a comunidade. Com o intuito de evitar fraudes na aquisição de flocos de algodão, as contas serão criadas e integradas a um número de telefone, utilizando a *API MyZap*, descrita na Subseção 3.2.5 (*Whatsapp* e *MyZap*). Com isso, haverá certa dificuldade na criação de novas contas *fakes*, visto que, não se torna vantajoso a aquisição de novos números de telefone apenas para obter alguns flocos de algodão.

As características do sistema proposto são detalhadas nas subseções a seguir. Na Seção 4.1, serão apresentados os artefatos gerados na análise do sistema e na Seção 4.2, será abordada a arquitetura da aplicação. Por fim, na Seção 4.3 será apresentada a modelagem do banco de dados e a prototipação das principais telas.

4.1 ANÁLISE DO SISTEMA

Essa seção apresentará a análise necessária para que haja um melhor entendimento de como irá ocorrer o desenvolvimento do sistema, utilizando as tecnologias e conceitos abordados no Capítulo 3.

Para o planejamento e gestão no desenvolvimento do sistema, o projeto seguirá as práticas da metodologia ágil *Scrum*, definida no Subseção 3.3.1. Os papéis de *Product Owner* e *Scrum Master* serão realizados pelo Professor Orientador e o time de desenvolvimento será composto apenas pelo aluno autor deste projeto. As Histórias de Usuário[1]que compõem o *Backlog* são apresentadas no Quadro 2, que foram previamente levantadas pelo aluno em conversas com o professor orientador no papel de *Product Owner*. Desta forma, conseguiu-se a identificação das principais funcionalidades que o sistema propõe-se a fazer.

Para manter um padrão na definição das Histórias de Usuário, foi utilizado o *template* COMO...QUERO...PARA e definido números de identificação (*IDs*) para cada História de Usuário, utilizando a extensão *Card Numbers for Trello*, a fim de facilitar a identificação e referenciação no texto. Também foi adicionado pesos as Histórias com o *power-up* do *Trello* chamado *Card Priority by Screenful*, sendo definido prioridades de 1 a 5, em que 5 tem o maior peso. A duração para a entrega de uma *Sprint* foi definida com período de uma semana e cada respectivo ponto de uma História tem a duração de 1 dia, que representará 3 horas. Mais precisamente, o aluno proponente deste projeto tem a disponibilidade de trabalhar 3 horas por dia, que serão aplicados em 5 dias da semana, caracterizando 15 horas semanais de desenvolvimento.

Para a organização e gerenciamento das Histórias do *Backlog* foi utilizado o sistema *Trello*, definido na Subseção 3.4.1, utilizando o recurso *Scrum for Trello* integrado juntamente com a visualização em modelo de *Kanban*, apresentado na Subseção 3.3.3, onde as colunas definidas foram nomeadas como *Product Backlog, Tasks, To Do, In Progress, Review e Done* (por *Sprint*), e podem ser visualizadas na Figura 7, sendo:

- **Product Backlog:** são Histórias de Usuário que devem ser implementadas;
- **Tasks:** são tarefas com funcionalidades que não estão ligadas diretamente a uma requisição de usuário;
- **To Do:** são as Histórias da *Sprint* corrente;
- **In Progress:** são Histórias que estão sendo desenvolvidas no momento;
- **Review:** são Histórias que estão aguardando revisão para conclusão;
- **Done (por Sprint):** são Histórias que já foram revisadas e então concluídas na *Sprint* atual. Cada *Sprint* finalizada terá a sua própria coluna.

Mais precisamente, a movimentação dos *cards* será dada do *Product Backlog* para o *To Do* quando a data da História entrar na *Sprint* corrente. Quando a tarefa for iniciada, será movida para *In Progress*. Por sua vez, quando for finalizada, será movida para o *Review*. Por fim, se a tarefa for aprovada, será movida para a coluna *DONE* da sua respectiva *Sprint*.

Para monitoramento do andamento do projeto, a fim de identificar se o desenvolvi-

mento ocorrerá dentro do prazo estipulado, será utilizado o gráfico de *Burndown*, descrito na Subseção 3.3.2. Ele será utilizado integrado ao *Trello*, que dispõe a ferramenta *Burndown for Trello* para acompanhamento do gráfico de maneira automatizada.

A primeira *Sprint* definida para o início do desenvolvimento do projeto compõe as História de Usuário de *IDs* 01 e 02 considerando que no início do desenvolvimento também serão destinadas algumas horas em configurações de ferramentas.

Quadro 2 – Histórias de Usuário

ID	Peso	Descrição da História
01	5	COMO usuário eu QUERO me cadastrar no sistema PARA ter acesso às funcionalidades
02	5	Como administrador eu QUERO configurar o MYZAP PARA enviar o código da validação do número do celular via Whatsapp aos usuários
03	5	COMO usuário eu QUERO me autenticar no sistema PARA doar ou receber doações
04	5	COMO usuário eu QUERO cadastrar doações PARA doá-las
09	4	COMO usuário eu QUERO visualizar as doações ofertadas PARA encontrar algo para aquisição
11	4	COMO usuário eu QUERO dar lances com flocos de algodão em doações PARA ganhar uma doação
07	4	COMO usuário eu QUERO visualizar minhas doações PARA verificar quais doações possuo
12	3	COMO usuário eu QUERO saber quem é o ganhador da minha doação PARA concluí-la
21	3	COMO usuário eu QUERO marcar como recusada ou recebida a doação que ganhei PARA finalizar a doação
13	3	COMO usuário eu QUERO receber os flocos de algodão de uma doação concluída PARA utilizá-los na aquisição de outras doações
05	2	COMO usuário eu QUERO editar a quantidade de flocos de algodão e o prazo limite de término de uma doação PARA alterar a doação
19	2	COMO usuário eu QUERO receber uma notificação PARA que eu possa saber se ganhei ou perdi um leilão de uma doação
06	2	COMO usuário eu QUERO deletar doações PARA não mais ofertá-las para doação
10	2	COMO usuário eu QUERO filtrar as doações por categoria PARA visualizar doações de maior interesse
14	1	COMO usuário eu QUERO alterar os dados da minha conta PARA atualizar informações
20	1	COMO usuário eu QUERO visualizar as doações no qual efetuei lances PARA visualizar meus lances
22	1	COMO Administrador QUERO cadastrar novas categorias PARA adicionar novos tipos de categorias
23	1	COMO Administrador QUERO cadastrar novos status PARA adicionar novos tipos de privilégio
24	1	COMO Administrador QUERO cadastrar novos papéis de usuário PARA adicionar novos tipos de usuário.
25	1	COMO Administrador QUERO cadastrar novos usuários de maior privilegio PARA ajudar na manutenção da aplicação
26	1	COMO Administrador QUERO definir a quantidade de moedas virtuais iniciais das contas PARA que quando um usuário crie sua conta ele tenha disponível uma quantidade de moedas limitadas

[1]Todas as historias de usuário podem ser visualizadas em: https://trello.com/b/JsbGrF2o/cottonnect

4.2 ARQUITETURA DO SISTEMA

A respeito do desenvolvimento do sistema, este será dividido em dois projetos. A comunicação entre eles é definida pelo padrão cliente/servidor apresentado pelo *REST*, descrito na Subseção 3.2.2 e pode ser analisado na Figura 10

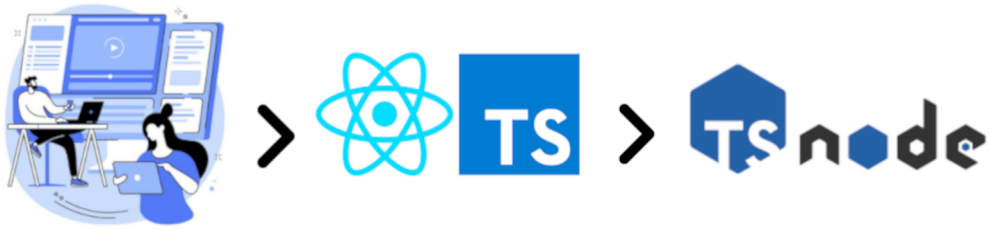


Figura 10 – Imagem da arquitetura da aplicação *Cottonnect*.
Fonte: A autora.

- **Servidor/API:** é o *back-end* do sistema. Será responsável, em termos gerais, por abrigar as implementações das regras de negócio. A *API* será implementada no estilo de arquitetura *REST* e seguirá o padrão *JSON Web Token (JWT)* para a autenticação. Ela será desenvolvida utilizando a biblioteca *Node.js* descrita na Subseção 3.2.1, permitindo o uso do *TypeScript*, que será a linguagem de desenvolvimento utilizada, citada na Subseção 3.1.2. Para auxiliar no controle de rotas da aplicação, será utilizado o *framework Express*, descrito na Subseção 3.2.4. A fim de garantir integridade ao sistema, a *API* será desenvolvida utilizando as práticas do *TDD*, definidos na Subseção 3.3.4 a fim de realizar os teste unitários. Para a implementação dos testes será utilizado a biblioteca *Jest* definida na Subseção 3.4.4. Os testes serão integrados juntamente com a ferramenta *GitHub Actions*, que os executará de formar automatizada.
- **Cliente:** é o *front-end* da aplicação. Será responsável por implementar tudo o que é visual para os usuários. Para o seu desenvolvimento será usada a tecnologia *React.js*, definida na Subseção 3.1.3, juntamente com a utilização na linguagem *TypeScript*. A aplicação *front-end* obterá os dados acessando a *API* do projeto, em que cada requisição será necessário o envio do *Token* para permissão de acesso e as informações serão obtidas em formato *JSON*.

Os códigos dos projetos serão gerenciados utilizando o *GIT* e hospedados no *GITHUB* referenciados na Subseção 3.4.2. As aplicações serão hospedadas no *Heroku*, citado na Subseção 3.4.3 usando a automação de *deploy* automático com a ferramenta *GitHub Actions* e a fim de garantir a qualidade do código escrito, a *API* e o *front-end* utilizarão o *linter ESlint* descrito na Subseção 3.4.5.

4.3 MODELAGEM E DESENVOLVIMENTOS DE PROTÓTIPOS

Nesta seção serão apresentadas as modelagens e os protótipos gerados para a elaboração do projeto do sistema. A Subseção 4.3.2 apresenta o protótipo das principais telas do sistema e as descrições de suas funcionalidades.

4.3.1 Modelagem do Banco de Dados

O sistema gerenciador de banco de dados definido para o desenvolvimento da aplicação foi o *PostgreSQL*, descrito na Subseção 3.2.6. Para facilitar sua utilização, será empregado o uso da ferramenta *ORM TypeORM*, descrita na Subseção 3.2.3.

Para representar o banco de dados foi criado o diagrama de modelagem[2] de dados com os atributos e relacionamentos das principais entidades que o sistema possui, como pode ser analisado na Figura 11. O modelo apresentado corresponde a um protótipo das tabelas que serão implementadas no sistema, com os relacionamentos, e as colunas com seus respectivos tipos.

No diagrama, pode-se visualizar a existência da tabela chamada *Users*, que é responsável por conter os dados de acesso do usuário, como nome, informações adicionais, número de telefone, *e-mail*, senha, data de nascimento e o status da verificação do número de telefone. Também armazenará a quantidade de flocos de algodão que um usuário possui. Como o sistema possui dois tipo de usuários, foi definida a tabela *Roles* para indicar qual tipo de usuário a conta pertence. O usuário possui também a informação de sua localidade, sendo uma informação importante para a aplicação, pois é por meio dela que os usuários poderão encontrar doações, permitindo obter doações próximas a ele, essa localização é definida pela cidade e estado em que ele reside, sendo respectivamente armazenadas nas tabelas de *Cities* e *States*.

Auctions é a tabela principal da aplicação, a qual representa os leilões. Os leilões possuem o dia do fechamento, o usuário que está doando o objeto, assim como o lance ganhador, definido após o fechamento do leilão. O principal artefato dessa tabela são as doações, representadas na tabela *Donations-Objects*. Os objetos possuem título, descrição, fotos e categorias. Como um objeto pode possuir múltiplas categorias foi criada a tabela *Donation-category* que referencia quais categorias, armazenadas na tabela *Categories*, o objeto de doação possui.

Para representar os lances do leilão foi definida a tabela *Biddings*, que possui o valor do lance e em qual leilão ele será efetuado. Os leilões e as doações possuem um *status*. O leilão tem como *status* 'aberto', 'fechado' ou 'concluído'. Os lances só podem ser efetuados quando o status do leilão está aberto. Devido ao fato de que o ganhador pode recusar o objeto em análise, e ele é o único usuário que disputou o leilão, será necessário abrir novamente o leilão para os lances, passando do *status* de fechado novamente para aberto. Quando o ganhador

[2] A imagem completa do modelo físico do banco de dados pode ser encontrada clicando aqui

aceitar a doação, o *status* do leilão passa a ser concluído, não podendo retornar aos *status* anteriores.

Já as doações possuem *status* de em análise, recusada e recebida. O *status* estará em análise quando está aguardando o usuário ganhador receber o produto. Caso ele recuse, o *status* passará para recusado, onde deverá ser gerado outro ganhador. Caso não haja outros usuários, o leilão será aberto para lances novamente. Quando a doação for concluída, o seu *status* será definido como recebido, que é o momento que ocorre a transferência das moedas para a conta do doador.

A modelagem do banco de dados implementa a tabela denominada *audits* que faz referência a auditoria das moedas virtuais das contas dos usuários, salvando a data e horário que foi realizada a movimentação, qual foi o usuário que a realizou, a quantidade de moedas virtuais e o número identificador do leilão que ocorreu a doação.

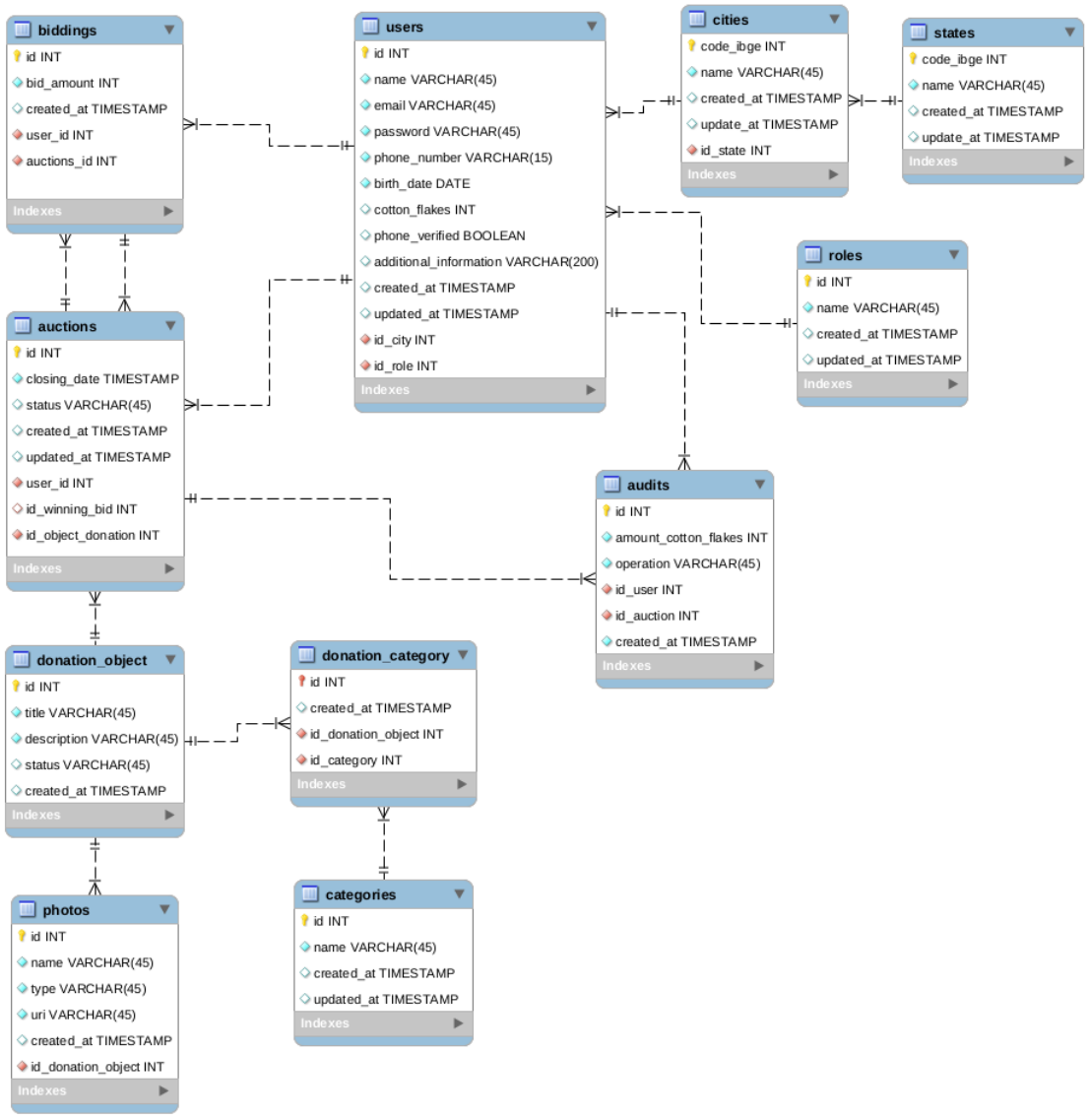


Figura 11 – Imagem da modelagem do banco de dados da aplicação *Cottonnect*.
Fonte: A autora.

4.3.2 Protótipos das Telas

O processo de prototipação das telas[3] foi realizado utilizando a ferramenta *Figma*, com o auxílio de *templates* disponibilizados pela comunidade. A seguir são apresentadas as prototipações das principais telas que compõem o sistema.

A criação da conta do usuário se dá por meio da construção de quatro telas principais. A primeira representada na Figura 12, dispõe as informações de nome, *e-mail*, senha, telefone, data de nascimento e um campo para informações adicionais que o usuário poderá inserir à respeito do seus contatos. Este campo é importante para facilitar que o doador entre em contato com o ganhador, pois será disponibilizado ao doador, quando uma doação for concluída. A Figura 13 apresenta a segunda etapa do cadastro, onde o usuário deve informar sua localização, por meio da cidade e estado. Na Figura 14 é apresentada a terceira etapa do cadastro, onde o usuário deverá informar o código que foi enviado para confirmação do seu número de telefone e por fim, a Figura 15 apresenta a tela de sucesso quando uma conta é criada.



Figura 12 – Imagem que apresenta a primeira etapa na criação da conta.
Fonte: A autora.

[3] O layout com as imagens pode ser encontrado em:https://www.figma.com/file/v6I75nZj6EBvds8WtapVbj/Cottonnect



Figura 13 – Imagem que apresenta a segunda etapa na criação da conta.
Fonte: A autora.



Figura 14 – Imagem que apresenta a terceira etapa na criação da conta.
Fonte: A autora.

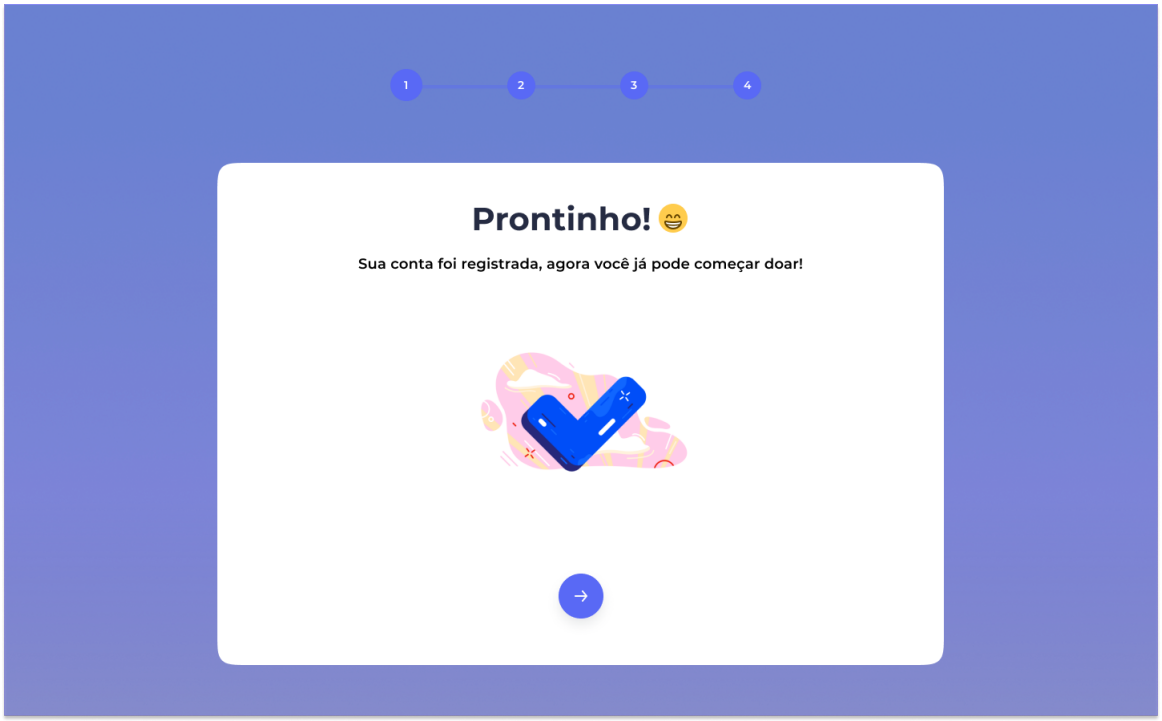


Figura 15 – Imagem que apresenta a tela de sucesso na criação de uma conta.
Fonte: A autora.

A Figura 16 apresenta a tela principal da aplicação, onde ocorre a listagem de todas as doações disponíveis para o usuário por meio do cadastro da sua localização. Há um menu lateral, onde os usuários poderão navegar pelo sistema, sendo possível, acessar as páginas para anunciar uma doação, visualizar suas próprias doações, visualizar as doações ganhas, filtrar as doações por categorias e localização assim como também é possível filtrar as doações por meio do filtro na tela principal. Logo acima, no menu principal, o usuário poderá acessar seu perfil e também poderá receber notificações quando ganhar um produto, que serão apresentadas no ícone de sino, ao lado do identificador do usuário.

Para efetuar um lance em uma doação é disponibilizado o botão chamado "eu quero", clicando nele abrirá um modal, que pode ser visualizado na Figura 17 em que o usuário poderá informar a quantidade de moedas virtuais que ele quer dar de lance para a doação escolhida.

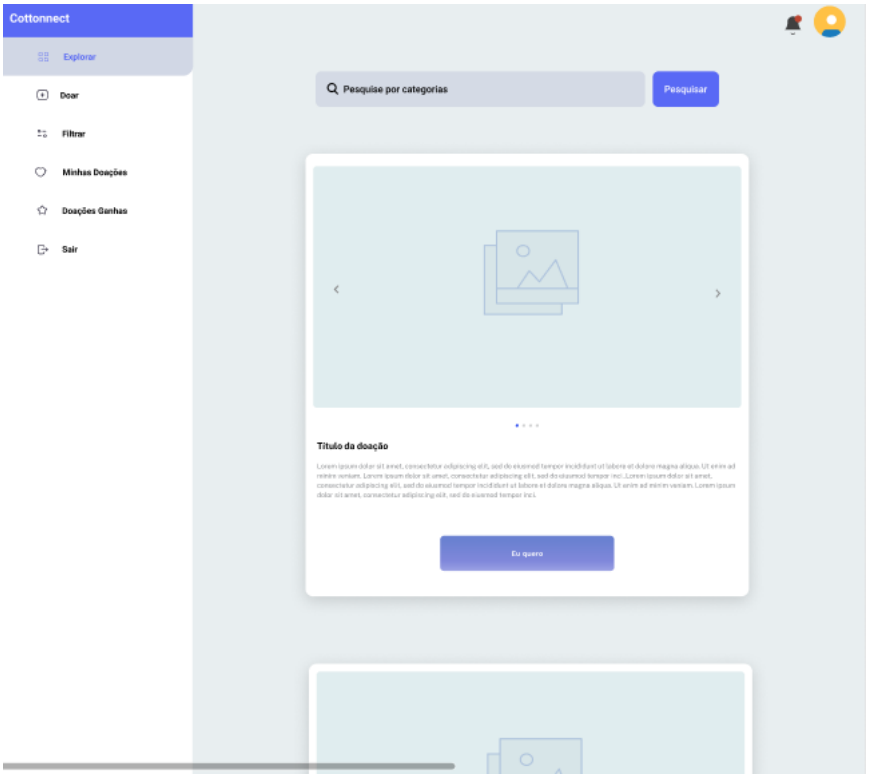


Figura 16 – Imagem da tela inicial que lista as doações.
Fonte: A autora.

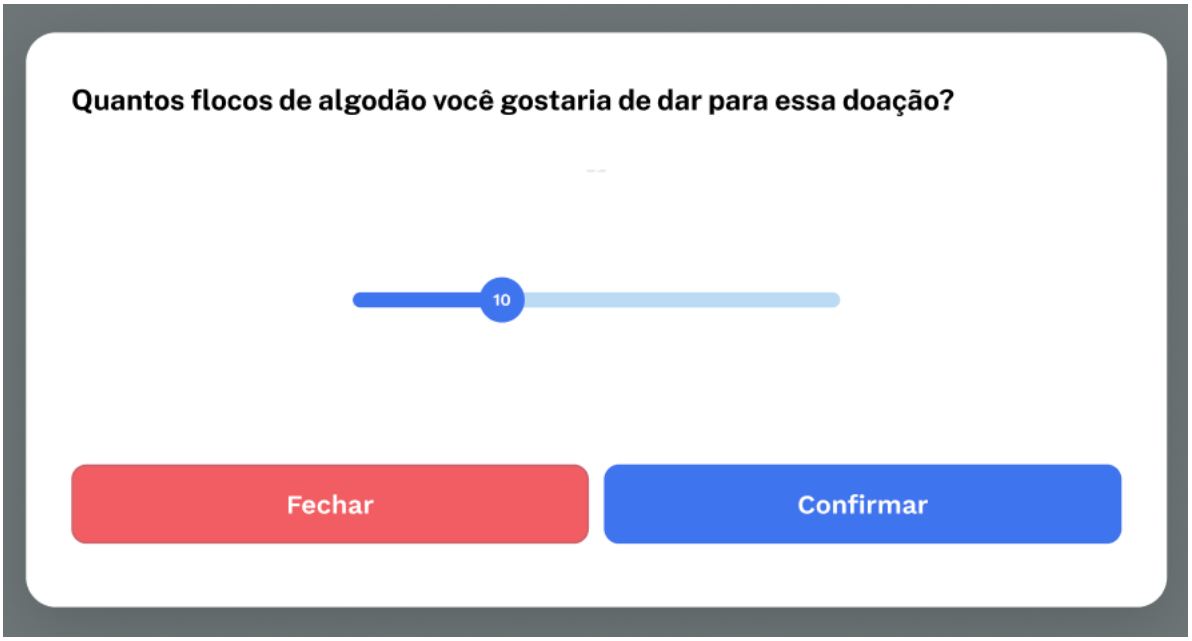


Figura 17 – Imagem da tela do modal que permite dar um lance para uma doação.
Fonte: A autora.

A Figura 18 apresenta a página em que é possível cadastrar uma doação. Por meio dela é possível inserir as informações do objeto a ser doado como título, descrição, categorias, fotos e a data de fechamento do leilão, no qual o sistema irá definir o ganhador da doação.

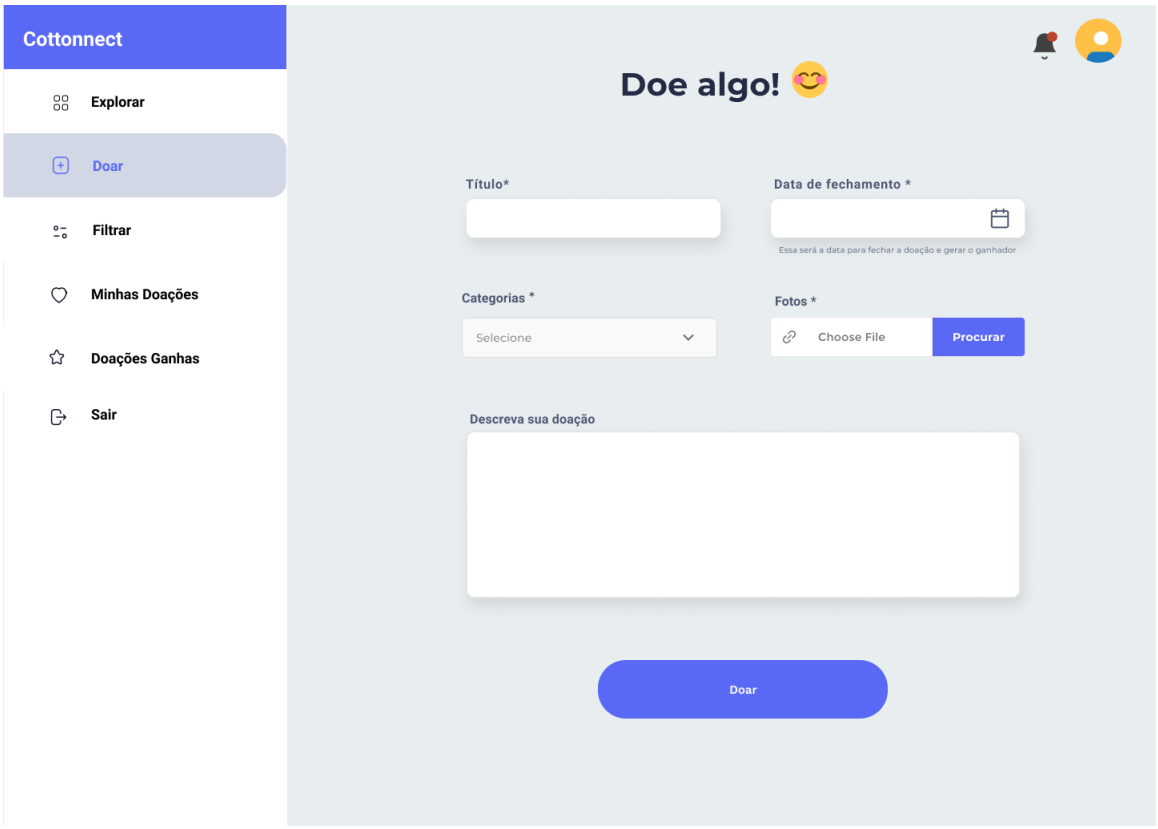


Figura 18 – Imagem da tela onde é realizado o cadastro de uma doação.
Fonte: A autora.

A Figura 19 apresenta todas as doações que foram cadastradas pelo usuário dono da conta. Elas podem ser visualizadas de acordo com seu status, e por meio delas é possível visualizar o ganhador quando esta for concluída.

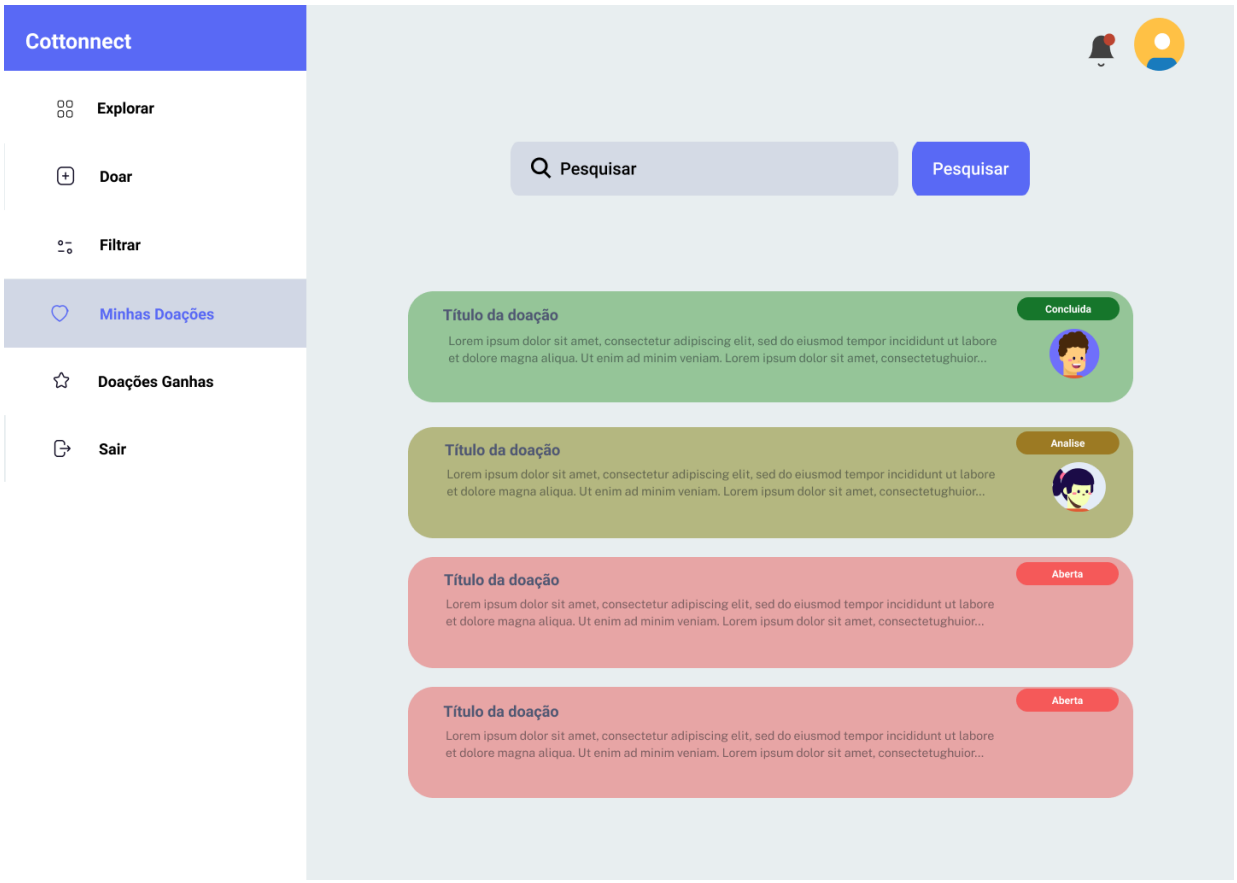


Figura 19 – Imagem que apresenta todas as doações de um usuário.
Fonte: A autora.

5 CONCLUSÃO

O resultado final pretendido com o presente trabalho é a entrega de um sistema para *web*, que posteriormente será evoluído para conformidade com a tecnologia *PWA*, que estimule doações de objetos entre comunidades, a fim de promover um consumo consciente nos indivíduos da sociedade, com o intuito de ajudar ao combate da degradação gradual do meio ambiente.

A aplicação deve permitir que indivíduos ofereçam seus objetos e em troca sejam retribuídos com moedas virtuais. Tais moedas são representadas simbolicamente por flocos de algodão. Desta forma, em posse das moedas será possível requisitar outras doações no sistema, na qual disputará o objeto por meio de um leilão *online* fechado.

Conforme exposto no Capítulo 4, o trabalho possui alguns resultados preliminares, como a definição das Histórias de Usuário, modelagem do banco de dados e o desenvolvimento dos protótipos das principais telas. A etapa posterior será dada pela continuidade da definição das ferramentas a serem utilizadas sendo seguida pelo início do desenvolvimento da aplicação.

5.1 TRABALHOS FUTUROS

Um dos trabalhos a ser implementado futuramente está diretamente relacionando com uma maior segurança ao sistema e aos usuários. Pensando em evitar fraudes na aquisição de moedas por doações falsas através de perfis falsos, futuramente poderá ser implementado um método de validação de veracidade de usuários através de verificação humana de fotos com algum documento de identificação. Desta forma, garante-se a real identidade do usuário e dificulta a criação de perfis falsos que fazem mau uso da aplicação.

6 CRONOGRAMA

O planejamento do trabalho que será desenvolvido pelo aluno, ao longo do período letivo, está descrito no cronograma do Quadro 3. Neste cronograma constam todas as atividades com seus respectivos prazos para o cumprimento. As atividades já concluídas foram marcadas com um "X". Em função da diminuição nos meses que constituem o período do semestre que ocorre o desenvolvimento do projeto e visando a construção de uma aplicação que envolve boas práticas, existe a possibilidade de extensão no prazo para o desenvolvimento caso o aluno em contato com o professor orientador identifiquem que não será possível entregar o projeto no período de tempo estipulado.

Quadro 3 – Cronograma de Atividades.

Atividades	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**
1. Defesa da proposta do TCC	X				
2. Revisão dos apontamentos da banca	X				
3. Revisão bibliografica	X	X			
4. Elaboração do projeto de TCC	X	X			
5. Defesa do projeto de TCC		X			
6. Revisão dos apontamentos da banca		X			
7. Desenvolvimento do sistema					
8. Validação dos requisitos implementados					
9. Teste do programa resultante					
10. Redação da monografia do TCC					
11. Defesa da monografia do TCC					

Referências

ABRELPE. **Panorama de Resíduos Sólidos no Brasil 2020**. [S.l.], 2020. Citado 2 vezes nas páginas 1 e 2.

ANICHE, M. Test-driven development. **Caelum**, 2014. Disponível em: <https://tdd.caelum.com.br>. Acesso em: 28 de março de 2021. Citado na página 17.

ANTUNES, A. Pwa: O que sÃo progressive web apps e por que usar? **GoBacklog**, 2021. Disponível em: <https://gobacklog.com/blog/progressive-web-apps>. Acesso em: 07 de maio de 2021. Citado na página 11.

ARTIA. Kanban. o que é e tudo sobre como gerenciar fluxos de trabalho. 2021. Disponível em: <https://artia.com/kanban>. Acesso em: 05 de maio de 2021. Citado na página 17.

ATLASSIAN. Fluxo de trabalho de gitflow. 2021. Disponível em: <https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow>. Acesso em: 11 de maio de 2021. Citado na página 19.

AUSUBEL, L. M. Auction theory for the new economy. 2003. Disponível em: <https://www.ausubel.com/auction-papers/auction-theory-new-economy.pdf>. Acesso em: 24 de abril de 2021. Citado na página 21.

BARSCH, B. Myzap - free open source whatsapp api. 2021. Disponível em: <https://github.com/billbarsch/myzap>. Acesso em: 07 de maio de 2021. Citado na página 14.

BAUER, C.; KING, G. **Hibernate in Action**. 6. ed. [S.l.]: Manning Publications, 2005. Citado na página 13.

CAMARGO, R. Burndown: conheça o gráfico que mede produtividade. **ROBSON CAMARGO Projetos e Negócios**, 2020. Disponível em: <https://robsoncamargo.com.br/blog/Burndown>. Acesso em: 27 de março de 2021. Citado na página 16.

COHN, M. **User Stories Applied: For Agile Software Development**. [S.l.]: Pearson Education, Inc., 2004. Citado na página 15.

Consumers International et al. **Manual de Educação para o Consumo Sustentável**. 2. ed. Brasília, 2005. Citado na página 2.

ESLINT. About. 2021. Disponível em: <https://eslint.org/docs/about/>. Acesso em: 28 de abril de 2021. Citado na página 21.

EXPRESS. Express. 2021. Disponível em: <https://expressjs.com>. Acesso em: 28 de abril de 2021. Citado na página 13.

FENTON, S. **Pro TypeScript: Application-Scale JavaScript Development**. 2. ed. [S.l.]: Apress, 2017. Citado na página 10.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**: Tese (phd in computer science) — university of california. [s.n.], 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Acesso em: 06 de maio de 2021. Citado na página 12.

FLANAGAN, D. **JavaScript**: O guia definitivo. 6. ed. [S.l.]: Bookman, 2013. Citado na página 10.

GIT. Git. 2021. Disponível em: <https://git-scm.com>. Acesso em: 28 de abril de 2021. Citado na página 18.

GITHUB. Github. 2021. Disponível em: <https://github.com>. Acesso em: 28 de abril de 2021. Citado na página 19.

GitHub Actions. Sobre integração contínua. 2021. Disponível em: <https://docs.github.com/pt/actions/guides/about-continuous-integration>. Acesso em: 18 de maio de 2021. Citado na página 19.

GITLAB. 2021. Disponível em: <https://about.gitlab.com/>. Acesso em: 18 de maio de 2021. Citado na página 20.

HAMEDANI, M. React virtual dom explained in simple english. 2021. Disponível em: <https://programmingwithmosh.com/react/react-virtual-dom-explained>. Acesso em: 04 de maio de 2021. Citado na página 11.

HEROKU. The heroku platform. 2021. Disponível em: <https://www.heroku.com/platform>. Acesso em: 18 de maio de 2021. Citado na página 20.

HEROKU. What is heroku? 2021. Disponível em: <https://www.heroku.com/about>. Acesso em: 28 de abril de 2021. Citado na página 20.

JEST. Jest. 2021. Disponível em: <https://jestjs.io/pt-BR>. Acesso em: 28 de abril de 2021. Citado na página 20.

JSON.ORG. Introducing json. 2021. Disponível em: <https://www.json.org/json-en.html>. Acesso em: 06 de maio de 2021. Citado na página 12.

JUSTO, A. S. 2017. Disponível em: <https://www.euaxcom.br/2017/05/metodologia-scrum-o-que-voce-precisa-saber-2>. Acesso em: 27 de março de 2021. Citado na página 15.

JWT. Introduction to json web tokens. 2021. Disponível em: <https://jwt.io/introduction>. Acesso em: 06 de maio de 2021. Citado na página 12.

MAURER, L. T.; BARROSO, L. A. Electricity auctions: An overview of e cient practices. The World Bank, 2011. Disponível em: <http://www.cgti.org.br/publicacoes/wp-content/uploads/2016/03/Teoria-dos-Leilo%CC%83es-Formulac%CC%A7o%CC%83es-e-Aplicac%CC%A7o%CC%83es-no-Setor-Ele%CC%81trico..pdf>. Acesso em: 11 de maio de 2021. Citado na página 21.

MILANI, A. **PostgreSQL - Guia do Programador**. [S.l.]: Novatec, 2008. Citado na página 14.

NIELSEN. Brasileiros estÃo cada vez mais sustentÁveis e conscientes. In: . 2019. Disponível em: <https://www.nielsen.com/br/pt/insights/article/2019/brasileiros-estao-cada-vez-mais-sustentaveis-e-conscientes/>. Acesso em: 20 de março de 2021. Citado na página 2.

NODE. About node.js. 2021. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 26 de abril de 2021. Citado na página 12.

POSTGREESQL.ORG. About postgresql. 2021. Disponível em: <https://www.postgresql.org/about>. Acesso em: 06 de maio de 2021. Citado na página 14.

REACTJS. React.js. 2021. Disponível em: <https://pt-br.reactjs.org>. Acesso em: 04 de maio de 2021. Citado na página 11.

ROSSETTI, J. P. **Introdução à economia**. 20. ed. [S.l.]: Atlas, 2013. Citado na página 1.

SAUDATE, A. **Rest - Construa APIs inteligentes de maneira simples**. 1. ed. [S.l.]: Casa do Código, 2013. Citado na página 12.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum**: Um guia definitivo para o scrum: As regras do jogo. [S.l.], 2013. Disponível em: <https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 27 de março de 2021. Citado na página 15.

SILVA, M. S. **HTML 5**: A linguagem de marcaÇÃo que revolucionou a web. 1. ed. [S.l.]: Novatec Editora, LTDA, 2011. Citado 2 vezes nas páginas 9 e 10.

TRELLO. Trello. 2021. Disponível em: <https://trello.com/pt-BR>. Acesso em: 28 de abril de 2021. Citado na página 18.

TYPEORM. Typeorm. 2021. Disponível em: <https://typeorm.io/#/>. Acesso em: 28 de abril de 2021. Citado na página 13.

TYPESCRIPT. O que é typescript? 2021. Disponível em: <https://www.typescriptlang.org>. Acesso em: 25 de abril de 2021. Citado na página 10.

W3SCHOOLS. O que é html? 2021. Disponível em: <https://www.w3schools.com/html/html_intro.asp>. Acesso em: 10 de maio de 2021. Citado na página 9.

WHATSAPP. Sobre o whatsapp. 2021. Disponível em: <https://www.whatsapp.com/about>. Acesso em: 11 de maio de 2021. Citado na página 13.

WWF-BRASIL. **Em 2019, Terra entra no cheque especial a partir de 29 de julho**. [S.l.], 2019. Disponível em: <https://www.wwf.org.br/?72262/Em-2019-Terra-entra-no-cheque-especial-a-partir-de-29-de-julho>. Acesso em: 01 de março de 2021. Citado na página 1.