CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
Faculdade de tecnologia da Baixada Santista Rubens Lara
Curso Superior de Tecnologia em Sistemas para Internet

Ayrton Felipe Sousa Tunes
Gabriel Morais Silva
Tarcísio Thallys da Costa Lima

# TROCAQUI: Plataforma para troca de itens

Santos - SP
Dezembro/2017

Ayrton Felipe Sousa Tunes
Gabriel Morais Silva
Tarcísio Thallys da Costa Lima

# TROCAQUI: Plataforma para troca de itens

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Baixada Santista Rubens Lara, como exigência parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Profa. Dra. Vanina Carrara Sigrist

Santos – SP
Dezembro/2017

LIMA, Tarcísio Thallys da Costa; SILVA, Gabriel Morais; TUNES, Ayrton Felipe Sousa.

Trocaqui: Plataforma para troca de itens / Ayrton Felipe Sousa Tunes, Gabriel Morais Silva, Tarcísio Thallys da Cota Lima. – Santos: Centro Estadual de Educação Tecnológica Paula Souza (CEETEPS), dezembro, 2017. 71 p.

Orientador: Profa. Dra. Vanina Carrara Sigrist

Trabalho de Conclusão de Curso (TCC). Centro Estadual de Educação Tecnológica Paula Souza, Faculdade de Tecnologia da Baixada Santista Rubens Lara (FATECRL). Curso Superior de Tecnologia em Sistemas para Internet.

Bibliografia.

1. Troca. 2. Internet. 3. Site

Ayrton Felipe Sousa Tunes
Gabriel Morais Silva
Tarcísio Thallys da Costa Lima

# TROCAQUI: Plataforma para troca de itens

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Baixada Santista Rubens Lara, como exigência parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Santos, 08 de dezembro de 2017.

**Banca Examinadora**

Profa. Dra. Vanina Carrara Sigrist
Faculdade de Tecnologia da Baixada Santista Rubens Lara
Presidente

Prof. Me. Ciro Cirne Trindade
Faculdade de Tecnologia da Baixada Santista Rubens Lara

Profa. Ma. Rosana Cammarosano
Faculdade de Tecnologia da Baixada Santista Rubens Lara

À instituição que nos possibilitou a conclusão deste marco.

## AGRADECIMENTOS

A Deus, por nos proporcionar tudo o que é ulterior ao nada, a existência em sua completude, que nos possibilita ações inteligíveis como a criação deste trabalho acadêmico.

Aos nossos pais e familiares, que auxiliam não com conhecimento técnico ou diretrizes acadêmicas, mas com recepção, motivação, amor e confiança, que só de um berço familiar provém.

À nossa orientadora, Profa. Dra. Vanina Carrara Sigrist, por tomar posse deste encargo e auxiliar tanto o desenvolvimento teórico, com suas aptidões para a linguagem e expressão, como o desenvolvimento prático, com percepções que muita das vezes fugiam ao nosso alcance.

À coordenadora deste trabalho, Profa. Ma. Dorotéa Vilanova Garcia, por disponibilizar seu tempo para efetuar correções em todos os passos e fornecer diretrizes que possibilitassem a conclusão com êxito.

Aos demais professores, por todo conhecimento transmitido ao decorrer de três anos, pois cada passo foi essencial e aplicável ao nosso escopo de conhecimento.

Enfim, aos que, por vínculos amigáveis, trouxeram-nos apreensão e instrução.

A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro.

Albert Einstein

## RESUMO

MORAIS, G.S; FELIPE. A.N; THALLYS, T.L. **Trocaqui**: plataforma para troca de itens. Ano 2017. 71f. Trabalho de Conclusão de Curso (Curso Superior Tecnológico em Sistemas para Internet) - Centro de Educação Tecnológica Paula Souza - Faculdade de Tecnologia da Baixada Santista “Rubens Lara”, Santos, 2017.

O atual cenário de numerosas produções comerciais, atrelado ao consumismo recorrente, implica um demasiado acúmulo de pertences nas mãos de pessoas que podem evidentemente descartá-los, mantê-los ou inutilizá-los. A internet, nessa circunstância, é predominante na conexão de cidadãos e seus interesses, podendo oportunamente ser espaço de encontro de pessoas que possuem objetos dos quais queiram se desvencilhar ou que desejem os pertences de outrem; oferecendo um caminho que não seja o descarte, o acúmulo ou a venda, mas a satisfação mútua da permuta. O presente trabalho visa a criação de um *site* que atenda a esse cenário, unificando diversas regiões e realidades em um objetivo comum: a troca. Para tanto, optou-se pelo auxílio de ferramentas e linguagens de programação que possibilitassem a implementação de uma usabilidade amigável e interativa – como as redes sociais predominantes no mercado –, um *design* que se adaptasse a diversos dispositivos e um padrão de internacionalização que mantivesse o sistema não somente no idioma português brasileiro, mas em outros.



**Palavras-chave:** Plataforma de Troca. Internet. Internacionalização. *Design* Responsivo.

## ABSTRACT

MORAIS, G.S; FELIPE. A.N; THALLYS, T.L. **Trocaqui**: plataforma para troca de itens. Ano 2017. 71f. Trabalho de Conclusão de Curso (Curso Superior Tecnológico em Sistemas para Internet) - Centro de Educação Tecnológica Paula Souza - Faculdade de Tecnologia da Baixada Santista “Rubens Lara”, Santos, 2017.

The current scenario of numerous commercial productions, coupled with recurrent consumerism, implies too much accumulation of belongings in the hands of people who can of course discard, maintain or render them useless. The internet, in this circumstance, is predominant in the connection of citizens and their interests, being able in due time to be a meeting place for people who have objects that they want to get rid of or that they want the belongings of others; offering a way other than discarding, accumulating or selling, but the mutual satisfaction of the exchange. The present work aims at the creation of a website that meets this scenario, unifying different regions and realities in a common goal: exchange. In order to do so, we opted for the use of tools and programming languages that would allow the implementation of a friendly and interactive usability - such as the social networks prevailing in the market - a design that would adapt to different devices and a pattern of internationalization that maintained not only in the Brazilian Portuguese language, but in others.



**Keywords:** Exchange website. Internet. Internationalization. Responsive Web Design.

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *AJAX* | *Javascript e XML* Assíncronos (*Asynchronous* JavaScript *and XML*) |
| *CSS* | Folha de estilo em cascata *(Cascading StyleSheets)* |
| *DOM* | Modelo de Objeto de Documento *(Document Object Model)* |
| *HTML* | Linguagem de Marcação de Hipertexto *(HyperText Markup Language)* |
| IBGE | Instituto Brasileiro de Geografia e Estatística |
| IHM | Interface Homem-Máquina |
| *JS* | *JavaScript* |
| *JSON* | Notação de Objetos *JavaScript (JavaScript Object Notation)* |
| *MVC* | Modelo-Visão-Controlador *(Model-View-Controller)* |
| PC | Computador pessoal |
| *PHP* | Pré-processador de hipertexto *(PHP - Hypertext Preprocessor)* |
| *SQL* | Linguagem de consulta estruturada *(Structured Query Language)* |
| *REST* | Transferência de Estado Representacional (*Representational State Transfer)* |
| SGBD | Sistema de Gerenciamento de Banco de Dados |
| *SOAP* | Protocolo Simples de Acesso a Objetos *(Simple Object Access Protocol)* |
| *SSH* | Shell seguro *(Secure Shell)* |
| *SSL* | Camada de Soquete Segura *(Secure Socket Layer)* |
| *UX* | Experiência do usuário *(User Experience)* |
| *WEB* | Rede de Alcance Mundial *(World Wide Web)* |
| *W3C* | Consórcio Teia Mundial de Computadores *(World Wide Web Consortium)* |

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Uma situação corriqueira que revela muito do cenário socioeconômico, tecnológico e cultural atual: alguém que queira se desfazer de um objeto, como uma peça de roupa que já não lhe serve mais ou um *smartphone* que julga “ultrapassado”, e não sabe bem para quem oferecer, se vende ou se doa, que preço atribuir e como contatar um novo proprietário para esse pertence.

Homens e objetos estão cada vez mais interligados em rede; a internet é espaço privilegiado da comunicação e de outras trocas, como as negociações financeiras: “Como nossa prática é baseada na comunicação, e a Internet transforma o modo como nos comunicamos, nossas vidas são profundamente afetadas por essa nova tecnologia da comunicação” (CASTELLS, 2001, p. 10). As distâncias geográficas não se configuram mais obstáculos físicos: “Províncias, regiões e nações, bem como culturas e civilizações, são atravessadas e articuladas pelos sistemas de informação e comunicação” (IANNI, 1997, p. 228); o volume de mercadorias aumenta drasticamente, resultante do consumo ainda excessivo, ou impensado (mesmo num cenário de crise há alguns anos); o descarte de diferentes produtos e resíduos é decorrente dessa postura e traz mais desafios aos cidadãos de todo o planeta.

> Ao se pensar no modo como os produtos de consumo são buscados atualmente, vê-se que não há espaços para a espera de uma satisfação futura. O consumidor pós-moderno, ao buscar freneticamente novos produtos, não se preocupa com a consequência dos gastos, pois é preciso adquirir agora, paga-se depois. (FERNANDO; EDUARDO, 2004, p. 426)

A internet, ao possibilitar a interconexão entre usuários dotados de intenções, pode reunir e estabelecer grupos de pessoas com objetivos em comum, uma comunidade digital que partilhe coisas especificas de seu interesse: isso sustenta o fato de existirem diversos portais cada qual com o seu público. As intenções mútuas dentre os internautas podem ser compartilhadas com o requinte de diversos recursos como imagem, som, hipertexto e vídeo. Contudo, pessoas que desejam se desvincular de seus bens podem, com o auxílio de um site, estabelecer relações comunicativas eficientemente.

Sendo assim, esta pesquisa pretende enfrentar o seguinte cenário: a necessidade de as pessoas se desvencilharem de todo tipo de pertence que já não lhes agrada mais – peças de vestuário, acessórios, eletroeletrônicos, objetos de casa

e decoração, utensílios domésticos, livros, CDs, vinis, itens de coleção – de uma forma mais prática, através do estabelecimento de relações e contatos com outras pessoas que estejam na mesma situação, dispostas a receber esses objetos e trocá-los por outros.

Hoje, é urgente que os meios para que essas trocas aconteçam, evitando geração de lixo, propiciando condições de consumo mais acessíveis e sustentáveis, fomentando a circulação de itens ainda usáveis e com valor simbólico-cultural agregado, sejam facilitados. Uma sociedade conectada, preparada para enfrentar os novos rumos das políticas mais recrudescidas, restritivas, precisa estar atenta, por exemplo, às possibilidades de comércio sem a circulação de dinheiro, mas sim nos moldes gerais daquela prática adotada há vários séculos, nos tempos do feudalismo – o escambo.

Tendo essas motivações em vista, optou-se aqui pelo desenvolvimento de uma plataforma *online* para troca de pertences dos mais variados gêneros, que fiquem ao dispor do maior número possível de usuários, não só do Brasil, mas de outros países, para que se crie uma rede de contatos com interesses afins.

## 1.1 JUSTIFICATIVA DO PROBLEMA

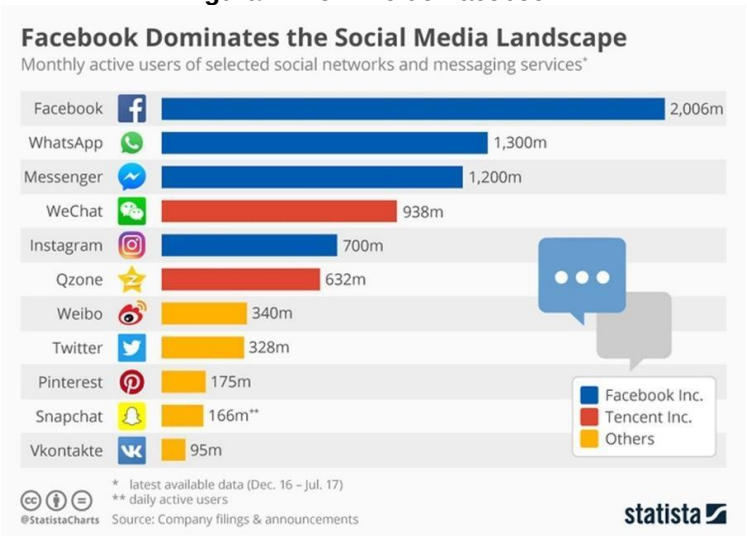
O aumento do fluxo de dados e informações pelas redes sociais digitais é observado por todos aqueles que participam desse mundo em rede e comentado por diversos especialistas que tentam, desde o início do século, analisar as tendências de mercado e os novos comportamentos dos usuários.

> Cada vez mais, as pessoas estão organizadas não simplesmente em redes sociais, mas em redes sociais mediadas por computador. Assim, não é a Internet que cria um padrão de individualismo de rede, mas seu desenvolvimento que fornece um suporte material apropriado para a difusão do individualismo em rede como a forma dominante de sociabilidade. (CASTELLS, 2001, p. 109).

> O que percebemos é que, com a própria evolução da web, esta vai também potencializando que o indivíduo apresente-se, identifique-se, personalize suas ações e, na medida em que o faz, parece de alguma forma ingressar, estar em relação dentro da rede das redes. (FISCHER, 2008, p. 43-44).

A Figura 1 mostra como o Facebook predomina em meio aos outros aplicativos ou plataformas de mídias sociais, conforme dados de 2017:

**Figura 1**: Domínio do Facebook.

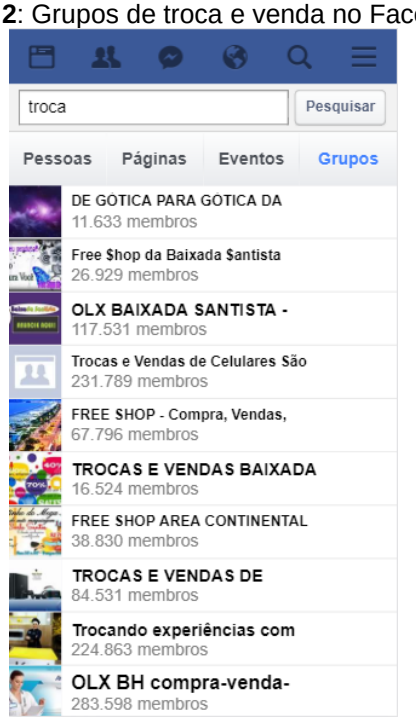



**Fonte**: Statista, 2017.

Portanto, pesquisou-se o termo “troca” nesta rede e, selecionando os 10 primeiros registros, o seguinte resultado, constado na Figura 2, foi obtido: grupos referentes a negociação – exceto o penúltimo – totalizam um número de 879.161 membros.

Obviamente, esses usuários podem estar buscando somente venda ou compra (o que não é o objetivo aqui), mas isso não enturva o sentido simbólico dos números: muitos usuários em grupos para negociação de itens em uma rede social digital.

**Figura 2**: Grupos de troca e venda no Facebook.




**Fonte**: Autor, 2017.

## 1.2 PROBLEMA DA PESQUISA

Como é possível facilitar a troca de pertences entre pessoas que talvez não se conheçam previamente, mas que tenham afinidades, já que podem estar interessadas pelos pertences umas das outras, a fim de permitir que tais objetos ganhem destinos e finalidades melhores do que o descarte puro e simples, ou mesmo a reciclagem e as práticas reversas pós-consumo?

## 1.3 OBJETIVOS

Os seguintes objetivos foram traçados:

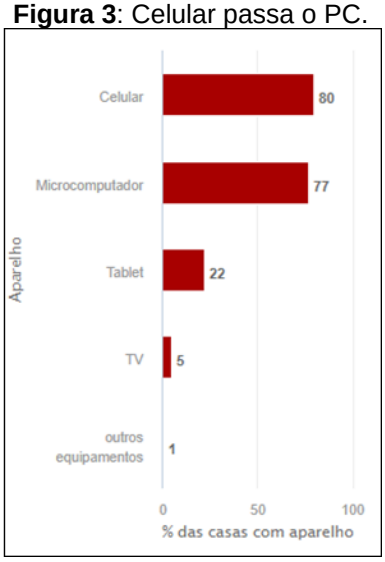
### 1.3.1 Objetivo geral

Desenvolver uma plataforma online que ofereça uma atmosfera propícia para que as pessoas possam oferecer seus pertences dos quais queiram se desfazer aos demais usuários, colocando esses itens numa “vitrina” virtual, e também possam navegar em busca de pertences de terceiros, que possam lhes servir de alguma forma.

### 1.3.2 Objetivos específicos

a) Implementar uma interface gráfica que seja de fácil manuseio e baixa complexidade, facilitando a interação do público, que pode ser formado também por usuários menos habituados aos meios digitais.
b) Programar a interface para se adaptar a computadores pessoais (PC) e dispositivos móveis, pois se sabe que atualmente as pessoas costumam acessar a internet com o celular, como mostram as estatísticas recentes do Instituto Brasileiro de Geografia e Estatística (IBGE) na Figura 3.
c) Fazer com que o site seja acessível em outros países, com o auxílio do recurso de internacionalização, que possibilita a multiplicidade de idiomas nas informações estáticas.
d) Atribuir características das redes sociais – gostei, seguir, perfil – tornando a plataforma mais dinâmica e atrativa aos usuários.

**Figura 3**: Celular passa o PC.




**Fonte**: IBGE, 2016.

## 1.4 PROCEDIMENTOS METODOLÓGICOS

A natureza da pesquisa é exploratória, com caráter qualitativo e aplicado, utilizando-se de métodos de revisão bibliográfica para composição do estado da arte, métodos descritivos e comparativos para análise de plataformas semelhantes existentes no mercado, de pesquisa de campo com o intuito de saber o entusiasmo do público em relação à proposta e de metodologias próprias ao desenvolvimento da plataforma.

## 1.5 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2, será apresentado o estado da arte. As plataformas semelhantes já disponíveis na internet serão exploradas, a fim de que se aponte o diferencial do presente trabalho.

No Capítulo 3, será discutida a metodologia, quando se explanará sobre as ferramentas e técnicas utilizadas na construção da plataforma e todos os padrões utilizados para auxiliar a performance da equipe.

Já no capítulo seguinte, será documentado o desenvolvimento. Contém toda a parte técnica do projeto e os passos dados até a conclusão da plataforma, em consequência dos métodos ilustrados no capítulo anterior.

No Capítulo 5, serão exibidos os resultados referentes ao sistema em produção.

Por fim, serão apresentadas as considerações finais, com o consenso final da equipe e a descrição do destino da plataforma.

# 2 ESTADO DA ARTE

Um conceito recente que tem circulado nos meios acadêmicos nos últimos anos é o consumo colaborativo. Uma pesquisa feita por Botsman e Rogers (2009, p. 14) elucida esse cenário:

> Todos os dias as pessoas estão usando o consumo colaborativo – compartilhamento tradicional, escambo, empréstimo, negociação, locação, doação e troca – redefinido por meio da tecnologia e de comunidades entre pares. O consumo colaborativo permite que as pessoas, além de perceberem os benefícios enormes do acesso a produtos e serviços em detrimento da propriedade, economizem dinheiro, espaço e tempo, façam novos amigos e se tornem cidadãos ativos novamente.

A internet, ainda segundo os autores, cumpre papel definitivo no consumo colaborativo, pois contribui para a formação de grupos e chances de encontro entre pessoas com afinidades, criando interações do tipo “muitos para muitos”. As redes criadas com diferentes grupos permitem que as práticas de consumo, como as citadas acima, com destaque ao escambo, se fortaleçam num âmbito mais distributivo e descentralizado.

Nesse sentido, um portal que possibilite a movimentação de pessoas que almejam de desvencilhar de seus pertences e adquirir outros abre um horizonte de possibilidades. “A prova social é fundamental para o consumo colaborativo porque a maioria das formas geralmente exige que as pessoas façam alguma coisa um pouco diferente e que elas mudem velhos hábitos” (Ibid., p. 70). Essa mudança de hábitos é benéfica na medida em que possibilita superar velhos paradigmas e se adequar a novas tendências.

É preciso avaliar que já existem algumas plataformas na internet voltadas ao desapego de objetos, a serem analisadas a fim de que seja possível delinear o perfil da plataforma Trocaqui nesta pesquisa.

## 2.1 OLX

Este é o maior site de classificados do Brasil, presente em 118 países, segundo sua página oficial (2017). Apesar de possibilitar ao usuário a divulgação de seu pertence, não é um concorrente direto, pelo fato de trabalhar com compra e venda – ao passo que o escopo aqui é privilegiar a permuta, sem monetização.

**Figura 4:** Página inicial do OLX.


**Fonte:** OLX, 2017.

Como mostra a Figura 4, o site possui uma interface agradável e, quando aberto em um dispositivo móvel, é redirecionado para um domínio mobile “m.olx.com.br” que possui um código específico para a exibição da página. Esta técnica é diferente do design responsivo que será utilizado nesta monografia.

O mecanismo que o *OLX* usa possui algumas possíveis desvantagens, conforme indica Zemel (2015, p. 19):

> Você pode ter um site focado para cada dispositivo, mas há diversas desvantagens nessa abordagem. E quando houver um novo dispositivo móvel, com uma tela bem diferente? Quando precisar fazer uma alteração, terá de tomar cuidado com cada uma das várias versões do seu site.

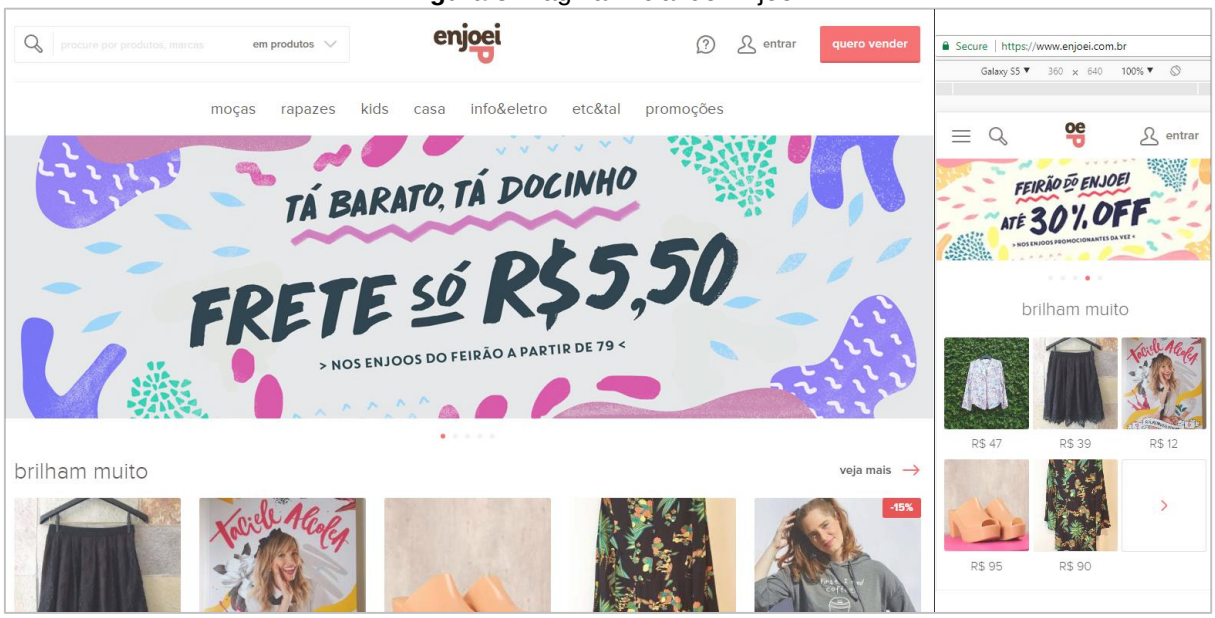
Contudo, a plataforma possui uma boa filtragem de localidade e categorias de produtos; além disso, possibilita ao usuário a criação de lojas. Contém muitos recursos que facilitam venda e compra no âmbito da internet: voltados ao e-commerce, e não às redes sociais.

## 2.2 ENJOEI

A empresa foi fundada no Brasil em 2009 e já está presente no exterior – na Argentina – no endereço “yafue.com.ar”. Segue a mesma via do *OLX*, porém possui

uma interface mais juvenil, com características de uma rede social (conforme Figura 5), e alguns termos específicos, como “*yeah yeah”,* que possui o mesmo intuito do *like*. Apesar de possuir aspectos almejados também aqui, o site não é um concorrente direto, pois ao seu público também interessa valores monetários.

**Figura 5**: Página inicial do Enjoei.




**Fonte**: Enjoei, 2017.

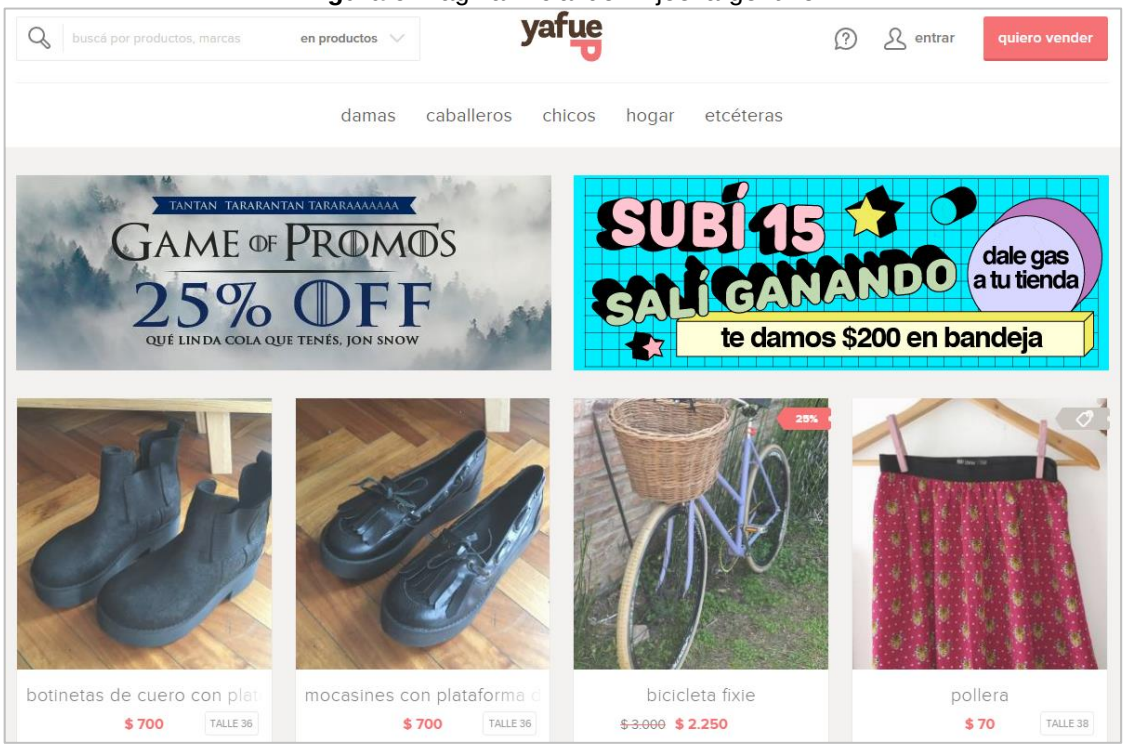
O site possui um *design* responsivo, pois o seu endereço continua o mesmo em um dispositivo móvel (diferente do *OLX*), pela adição do prefixo “m”.

Já no quesito internacionalização da linguagem, o site não dispõe de recursos. A única exceção é a página específica para a localidade argentina, que possui a língua castelhana, como visto na Figura 6. Por exemplo, ao mudar o idioma do navegador para inglês, a plataforma continuará sendo carregada no idioma português do Brasil. A plataforma Trocaqui desta pesquisa terá esse diferencial, por possuir a performance de reconhecer, no momento da requisição, o idioma do cliente, e devolvê-lo à página no respectivo idioma, caso este esteja disponível no servidor.

Segundo Ishida, Miller e Boeing (2015, s/p):

> A internacionalização afeta significantemente a facilidade de localização do produto. Atualizar um produto cultural e linguisticamente centrado para um mercado global é evidentemente muito mais difícil e demorado do que criar um item com o objetivo de lançá-lo globalmente.

**Figura 6**: Pagina inicial do Enjoei argentino.




**Fonte**: Yafue, 2017.

## 2.3 SWAPPING

Esta página é específica para negociação de utensílios de bebês. Possui opções para efetuar troca e compra. Possui um detalhe interessante: a moeda utilizada para as transações é uma criação da própria plataforma, sendo nomeada "*swap";* seus usuários podem obtê-la comprando com a moeda real através do PagSeguro.

Uma proposta bastante interessante e inovadora, visto que tal moeda pode circular entre os usuários através das negociações realizadas. Porém, assim como as duas plataformas citadas, *Swapping* não é um concorrente direto: mas é o único, até então, que possibilita uma opção de troca que notifica o usuário sobre sua intenção, semelhante à proposta aqui.

Foi arquitetado para atuar somente no Brasil, visto que não possui outros idiomas disponíveis; não conta com *design* responsivo e nem *site mobile,* contudo abarca uma interface agradável:

**Figura 7**: Pagina inicial do Swapping.


**Fonte**: Swapping, 2017.

## 2.4 PRA QUE DINHEIRO?

O nome subentende um negócio no modelo de escambo, mas esse site trabalha com valores monetários e também possui sua própria moeda, denominada “tutu”. Assim como o *OLX* e o *ENJOEI*, pode ser integrada na categoria dos classificados. Possui um *design* responsivo, conforme Figura 8, com algumas características de rede social.

**Figura 8**: Página inicial do “Pra Que Dinheiro?”.


**Fonte**: Pra Que Dinheiro?, 2017.

## 2.5 TROCA TROCA BRASIL

Um site de classificados que possui uma interface aos moldes dos jornais do segmento – diferente dos que já foram apresentados, que possuem algumas características de redes sociais – carente de um design responsivo ou site mobile (Figura 9). Apesar disso, a plataforma possui muitos acessos recentes, como se verifica ao clicar nos anúncios presentes na página inicial e ver as datas.

Possui um pequeno número de similaridades com a plataforma a ser desenvolvida aqui, mas convém a análise, visto que seu nome implica troca - uma palavra-chave desta pesquisa – e é um dos primeiros resultados do *Google* ao se pesquisar por “site de trocas”.

**Figura 9**: Página inicial do Troca Troca Brasil.

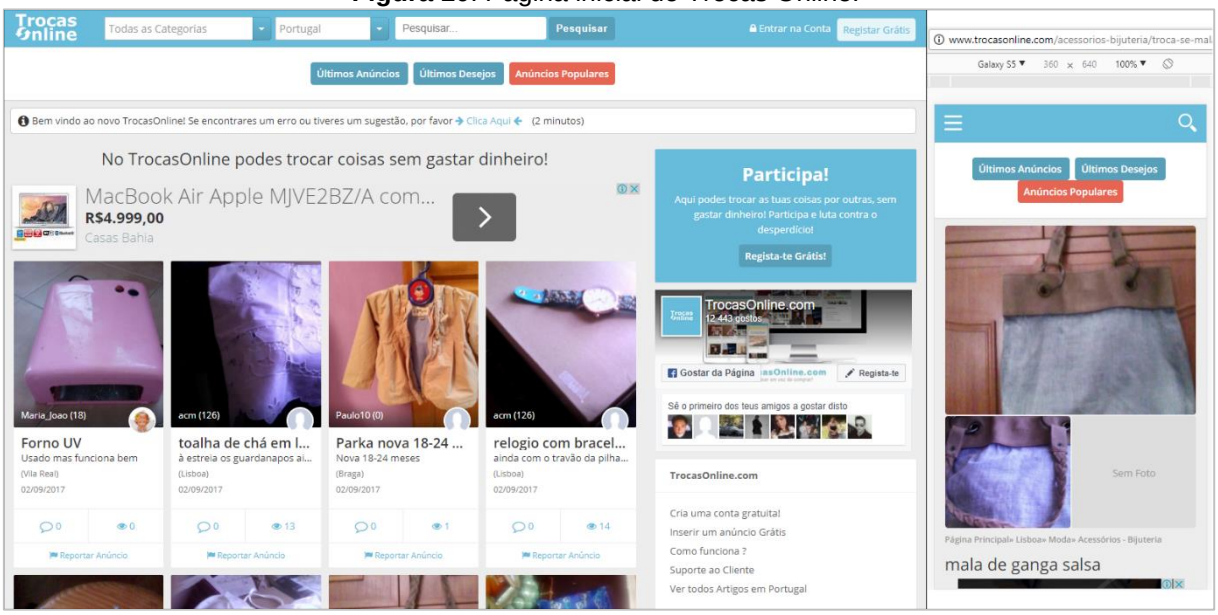

**Fonte**: Troca Troca Brasil, 2017.

## 2.6 TROCAS ONLINE

O último site a ser analisado faz jus ao termo que aqui foi aplicado: uma negociação sem fins monetários. Dentre os apresentados, é o único que pode ser considerado um concorrente direto, pelo fato de que trabalha somente com a divulgação de pertences dispostos à troca por outrem.

Conforme a Figura 10, pode-se perceber que o *site* se utiliza do *design* responsivo e possui uma interface agradável. Logo na página inicial são listados alguns itens ordenados por data em ordem decrescente.

**Figura 10**: Página inicial do Trocas Online.




**Fonte**: Trocas Online, 2017.

Ao clicar sobre algum item, o usuário é redirecionado para outra página que possui uma simples opção de troca (Figura 11), diferente dos demais, já apresentados, que envolvem compra. Após confirmar a proposta de troca, uma notificação é enviada ao dono do item anunciado.

**Figura 11**: Interação de troca do Trocas Online.


**Fonte**: Trocas online, 2017.

No perfil do usuário, tem uma simples interface de gestão (Figura 12), onde ele pode observar as suas interações executadas e recebidas. E, desse modo, concretiza-se o principal ciclo para efetuar a troca: um catálogo de itens, uma opção para oferta de troca e um painel para o dono do item recebê-la.

**Figura 12**: Perfil do usuário do Trocas Online.


**Fonte**: Trocas Online, 2017.

Apesar de o ciclo deste site atender aos objetivos descritos no presente trabalho, há uma diferença crucial: foi feito para operar em Portugal. Ou seja, ao realizar o cadastro e filtrar as pesquisas por localidades, aparecem somente cidades do país europeu.

Além disso, Trocas Online não possui o protocolo *SSL - Secure Socket Layer* (Camada de Soquete Segura), que, segundo Martins (2001, p. 1), “Tem como característica principal o estabelecimento de um canal privado (os dados são cifrados), autenticado (o servidor e o cliente podem ser autenticado) e confiável (o transporte de uma mensagem inclui uma verificação de integridade)”.

Por mais que o site não trafegue informações confidenciais, existe, então, um déficit de segurança.

# 3 METODOLOGIA

Neste capítulo, abordam-se as ferramentas e os padrões necessários para o desenvolvimento da plataforma em questão, desde os métodos julgados mais eficientes às linguagens necessárias para a programação de computadores.

## 3.1 GIT

Este é um sistema para versionamento de códigos que “registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas mais tarde” (CHACON; STRAUB, 2014, s/p). Com isso, pode-se manter um histórico organizado e reutilizável das etapas de desenvolvimento do código-fonte que também podem ser úteis como portfólio futuramente: atestando a veracidade qualitativa dos feitos.

> O método de controle de versão de muitas pessoas é copiar os arquivos para outro diretório (talvez um diretório com carimbo de tempo, se eles forem espertos). Esta abordagem é muito comum, porque é tão simples, mas também é incrivelmente propenso a erros. É fácil esquecer em qual diretório você está e acidentalmente sobrescrever o arquivo errado ou copiar arquivos que não quer. (Ibid., s/p)

### 3.1.1 Github

> O *GitHub* é um site em que você pode carregar uma cópia de seu repositório *Git*. Ele permite que você colabore muito mais facilmente com outras pessoas em um projeto. Isso é feito por meio da disponibilização de um local centralizado para compartilhar o repositório, uma interface web para visualiza-lo e recursos como *forking*, *pull requests, issues* e *wikis,* que permitem especificar, discutir e revisar alterações junto à equipe de maneira eficiente. (BEER; BELL, 2015, p. 13)

Esta ferramenta facilita o trabalho em equipe e torna o *Git* mais legível e acessível. Atuando em uma plataforma online, possibilita a inserção de membros em um determinado projeto. E, com isso, cada desenvolvedor pode desenvolver suas partes e mesclar ao todo após finalizado.

Há um recurso que também é de grande utilidade e será utilizado para beneficiar as futuras manutenções do projeto no servidor: *deploy* automático. Sobre o

auxílio do protocolo *SSH - Secure Shell* (Shell Seguro), a plataforma em produção será atualizada após cada mesclagem final efetuada no *github*.

## 3.2 REST

Um paradigma de arquitetura de sistemas na *WEB - World Wide Web* (Rede de alcance mundial) que possibilita a comunicação de aplicações, o qual, conforme Xavier (2011, p. 41), “foi introduzido originalmente como um estilo arquitetural para construção de aplicações hipermídia de grande escala distribuídas”. Sua importância aqui é a de construir um padrão de comunicação onde os dados e métodos possam trafegar do dispositivo do usuário até o servidor, independente da diferença entre eles, visando a interoperabilidade.

Sendo assim, implementações futuras, como a de um sistema para *iOS* ou *Android*, poderão desfrutar desses dados e métodos, bastando apenas a aplicação de uma interface que possibilite acesso a um *endpoint* – endereço que recepta e responde a comunicação. “A arquitetura REST foca principalmente em recursos para interação com os clientes. Tais recursos são identificados através de URIs. Com isso, cada recurso tem um endereço global que pode ser usado para descoberta de serviços” (Ibid., p. 41).

Existem outras notações, como *SOAP - Simple Object Access Protocol* (Protocolo Simples de Acesso a Objetos), mas a *REST - Representational State Transfer* (Transferência de Estado Representacional) - se apresenta como mais viável para aplicações *web* pelo fato de suportar *JSON - JavaScript Object Notation* (Notação de Objetos *JavaScript*). Pois, como relata Xavier (2011, p. 20), “baseada na linguagem JavaScript, essa notação consegue descrever diferentes tipos de estrutura de dados (como objetos e *arrays*), seguindo a sintaxe da própria JavaScript”. E, assim, pode-se manipular os dados pertinentes ao projeto como se fossem entidades: um item que possui nome, cor e diversas descrições etc., através do auxílio de um *endpoint*.

> Os recursos são manipulados utilizando um conjunto de quatro operações: inserção, atualização, exclusão e listagem/leitura. Cada operação utiliza um método de requisição HTTP diferente. O método GET retorna o estado corrente do recurso em alguma representação. PUT insere um novo recurso, que pode ser excluído com DELETE. POST transfere um novo estado para um recurso (atualização). (Ibid., p. 41-42)

## 3.3 BACK-END

Sessão pertinente aos utensílios que trabalham de forma restrita no servidor e fornecem páginas, regras de negócios, permissões e acesso ao banco de dados.

### 3.3.1 MySQL

> Em um mundo onde a globalização está cada vez mais presente, os processos cada vez mais automatizados e as barreiras de distância sendo quebradas pelo aumento da popularização da internet, a necessidade de armazenamento de dados e informações de mercado torna-se o primeiro passo para a migração de seu negócio para a internet. É nessa etapa que entra o MySQL, visando a suprir essa necessidade da melhor forma possível. (MILANI, 2006, p. 22)

É o sistema de gerenciamento de banco de dados (SGBD) – que usufrui da linguagem *SQL - Structured Query Language* (Linguagem de consulta estruturada) – utilizado para operar em função das necessidades da plataforma: armazenando os dados dos usuários, itens, interações e relações de troca efetuadas. Está sendo usada sua licença livre, de grande potencial, vale dizer, conforme indica Milani (Ibid., p. 22): “banco de dados *open source* com maior capacidade para concorrer com programas similares de código fechado, tais como *SQL Server* (*Microsoft*) e *Oracle*”; possui uma grande compatibilidade com a linguagem de programação em uso: *PHP - Hypertext Preprocessor* (Pré-processador de hipertexto).

### 3.3.2 CodeIgniter

Como já dito, o *PHP* é a linguagem utilizada na escrita do código no servidor. Possui as suas facilidades e grande conteúdo e comunidade presentes na internet pelo fato de ser *open source*. Foi conjecturada a ideia do uso da linguagem *Haskell* por possuir uma base matemática satisfatória ao conhecimento da equipe; porém, sua comunidade não é tão grande como a da escolhida.

> O PHP é uma das linguagens mais utilizadas na web. Milhões de sites no mundo inteiro utilizam PHP. A principal diferença em relação às outras linguagens é a capacidade que o PHP tem de interagir com o mundo web, transformando totalmente os websites que possuem páginas estáticas. (NIEDERAUER, 2011, p. 23)

Todavia, para facilitar ainda mais o uso dessa linguagem e preservar o curto tempo de desenvolvimento, a *framework CodeIgniter* foi escolhida, levando em conta a documentação detalhada e requintada, que, como relata Antunes (2011), decrementa o tempo gasto com leitura e incrementa o passar do tempo no desenvolvimento.

> Com o CI, é possível desenvolver sites, APIs e sistemas das mais diversas complexidades, tudo de forma otimizada, organizada e rápida. Suas bibliotecas nativas facilitam ainda mais o processo de desenvolvimento, e ainda à necessidade de cada projeto. Diversas bibliotecas de terceiros (third-party) estão disponíveis no GitHub, Composer e em outros repositórios de arquivos, e podem ser muito úteis. (Ibid., s/p)

Apesar de o *framework* possuir uma diretiva somente para as interfaces gráficas trabalharem acopladas à regra de negócio, optou-se pelo desacoplamento das camadas com o auxílio do *REST*. Dessa forma, a implementação do padrão de projeto *MVC - Model-View-Controller* (Modelo-Visão-Controlador) utilizado na ferramenta torna-se mais aguçada e condizente com a teoria.

> A abordagem MVC é composta por três tipos de objetos. O modelo é o objeto de aplicação, a Visão é a apresentação na tela e o Controlador é o que define a maneira como a interface do usuário reage às entradas do mesmo. Antes da MVC, os projetos de interface para o usuário tendiam a agrupar esses objetos. A MVC separa esses objetos para aumentar a flexibilidade e a reutilização. (GAMMA, 2000, p. 20)

#### 3.3.2.1 Internacionalização

Atendendo a um dos principais objetivos, a *framework* tem um grande suporte para o multi-idioma. Já vem com uma pasta denominada *language* na qual se pode colocar diferentes arquivos voltados a um idioma específico. E o controlador é o responsável por indicar qual arquivo utilizar, mediante o idioma do dispositivo do usuário. “Language – Diretório que armazena os arquivos com o dicionário de idiomas, assim você pode desenvolver em multi-idiomas de forma fácil...” (ANTUNES, 2011, s/p).

#### 3.3.2.2 Migration

Sucintamente, Antunes (2011, s/p) indaga sobre uma grande questão: “Em algum momento durante o desenvolvimento ou pós-desenvolvimento de uma aplicação, pode ser necessário fazer mudanças na estrutura do banco de dados, alterando a estrutura original”. Isso pode trazer alguns problemas ao decorrer do desenvolvimento, porém a *framework* nos oferece o *migration,* ferramenta capaz de alterar o banco de dados de maneira organizada, como justifica Antunes.

## 3.4 FRONT-END

Em oposição ao capítulo anterior, este trata dos utensílios que trabalham publicamente, dando assistência à Interface Homem-Máquina (IHM), fornecendo ao usuário todos os elementos necessários para findar os propósitos de usabilidade na plataforma.

### 3.4.1 Materialize

Um *framework* que opera em benefício da perfeita mescla das linguagens *HTML - HyperText Markup Language* (Linguagem de Marcação de Hipertexto), *CSS - Cascading StyleSheets* (Folha de estilo em cascata) e *JavaScript (JS)* – usadas para o *front-end*. São essências para a anatomia da página, sendo o *HTML* a base para a criação da estrutura que exibe conteúdos; o *CSS* a formatação e estilização da estrutura proveniente do item anterior que é limitada em estética (também fornecendo recursos para o design responsivo, como as *media queries*); e o *JS* possibilita a programação da estrutura sucedida pelos seus antecessores. (CASTRO; MANARA, 2014).

Material *Design* é um estilo de *design* projetado pela *Google* e está presente em seus sistemas *Android*, *Gmail*, *Youtube* etc. A *framework* fornece esse estilo para a confecção de *sites* com diversos recursos e uma grande documentação em seu site oficial.

#### 3.4.1.1 Design Responsivo

Como descrito nos objetivos, um requisito essencial e enfático é a possibilidade de a plataforma se ajeitar aos diferentes tipos de dispositivo. “A chave para o design responsivo é fazer um design flexível e adaptável, que se ajuste às características do navegador, do dispositivo e do contexto do usuário” (LOPES, 2013, p. 29), e a *framework* atende a esses requisitos. Fornece o conceito de *grid* que oferece colunas e linhas, voltadas a diferentes resoluções, para assim organizar o conteúdo da página.

> Outra abordagem para criação de layouts fluídos é utilizar grids flexíveis. Grids são um conjunto de linhas bases que fornecem uma estrutura para o seu layout. O sistema de grids divide o site em colunas de mesma largura onde se organiza o conteúdo, sendo normalmente utilizados grids de até 12 colunas. (ALMEIDA, 2014, p. 12)

### 3.4.2 JQuery

*Framework JavaScript* que possui um grande potencial em facilitar o uso dessa linguagem. Em especial, no presente trabalho, para realizar os efeitos de *UX - User Experience* (Experiência do usuário) e manipulação do *DOM - Document Object Model* (Modelo de Objeto de Documento), pois, como diz Balduíno (2014), ao escrever-se ou ler-se os dados em página *HTML*, estamos mantendo um contato direto com a estrutura *DOM*; o *JQuery* ameniza os esforços para tal. Conforme indica a Tabela 1, há grande redução na codificação ao escrever a manipulação do *DOM* com a *framework*.

**Tabela 1**: Comparação das sintaxes.

| Sintaxe JavaScript | Sintaxe jQuery |
|---|---|
| document.getElementsByTagName("p") | $("p") |
| document.getElementById("um").setAttribute("class", "cor") | $("#um").attr("class", "cor") |

**Autor**: Silva, 2009.

Também está sendo usada para redigir os códigos que realizam as conexões com o *back-end* através da tecnologia *AJAX - Asynchronous* JavaScript *and XML* (*Javascript e XML* Assíncronos) – nativa do *JS* – que, segundo Silva (2009), é facilitada pelo *jQuery*, que agrupa as suas funcionalidades e trata as requisições de maneira simplificada.

### 3.4.3 Vue

Este *framework* foi escolhido para persistir os dados assíncronos da plataforma, ou seja, os dados dos itens e usuários que são recebidos do *back-end*, e aqueles que serão enviados e se encontram sendo manipulados no *DOM*. O *Angular*, seu concorrente, foi cogitado, porém possui falta de clareza quando comparado ao *Vue* e uma complexidade em seu mecanismo interno que dificulta a inserção de outros padrões, por exemplo, o próprio *jQuery*. Incau (2017, s/p) capta esse diferencial entre esses dois *frameworks*:

> O vue se diferencia por ser uma biblioteca não intrusiva, ou seja, ele não tem código que o force a seguir o padrão dele, como é o caso do Angular 2 com TypeScript. A maioria do código para Vue é escrito em JavaScript puro. Ele possui uma sintaxe muita clara e limpa.

## 3.5 FIREBASE CLOUD MESSAGING

Um serviço, fornecido pela *Google*, que possibilita o envio de notificações entre plataformas. Provêm chaves de acesso para que o servidor possa requisitar o envio de notificações e também para que os usuários sejam identificados como o possível destino da mensagem. Sua aplicação está tanto no *back-end* como no *front-end*.

# 4 DESENVOLVIMENTO

Os procedimentos, códigos-fonte e análises técnicas inescusáveis à construção da plataforma *online* Trocaqui estão registrados neste capítulo.

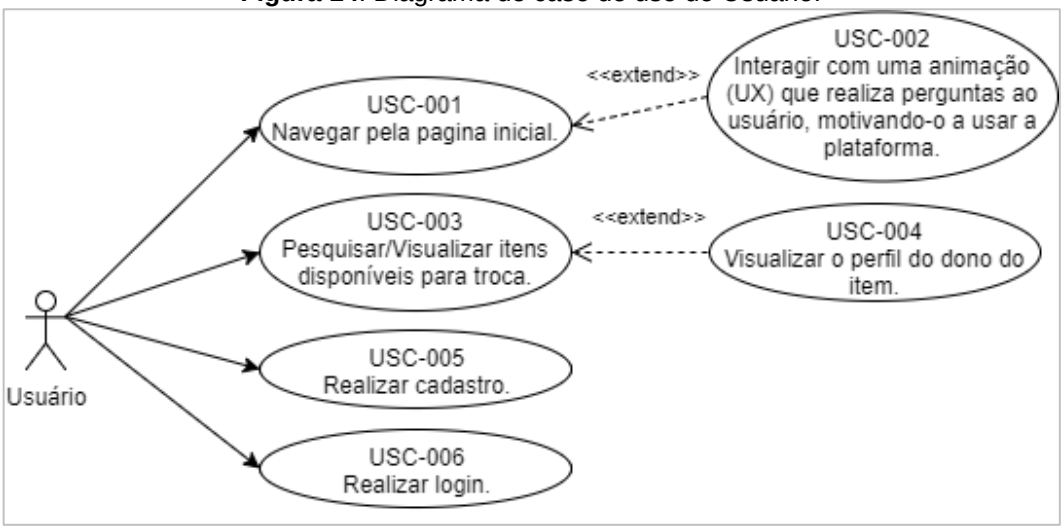
## 4.1 DIAGRAMA DE CASO DE USO

Diagramação que descreve as funcionalidades do sistema em pauta. Através dela, foi possível introduzir toda uma prototipagem – até então teorética e opinativa – num ambiente computacional; eleger, dentre diversas alternativas, os principais casos de uso que atendam ao objetivo descrito: delimitando a área de trabalho para o remate.

### 4.1.1 Usuário

Obviamente, todos que acessam o sistema são considerados usuários. Porém, algumas ações são concebidas somente a usuários autenticados. Na figura 14, o estudo foi voltado aos usuários "*offline*".

**Figura 14**: Diagrama de caso de uso do Usuário.



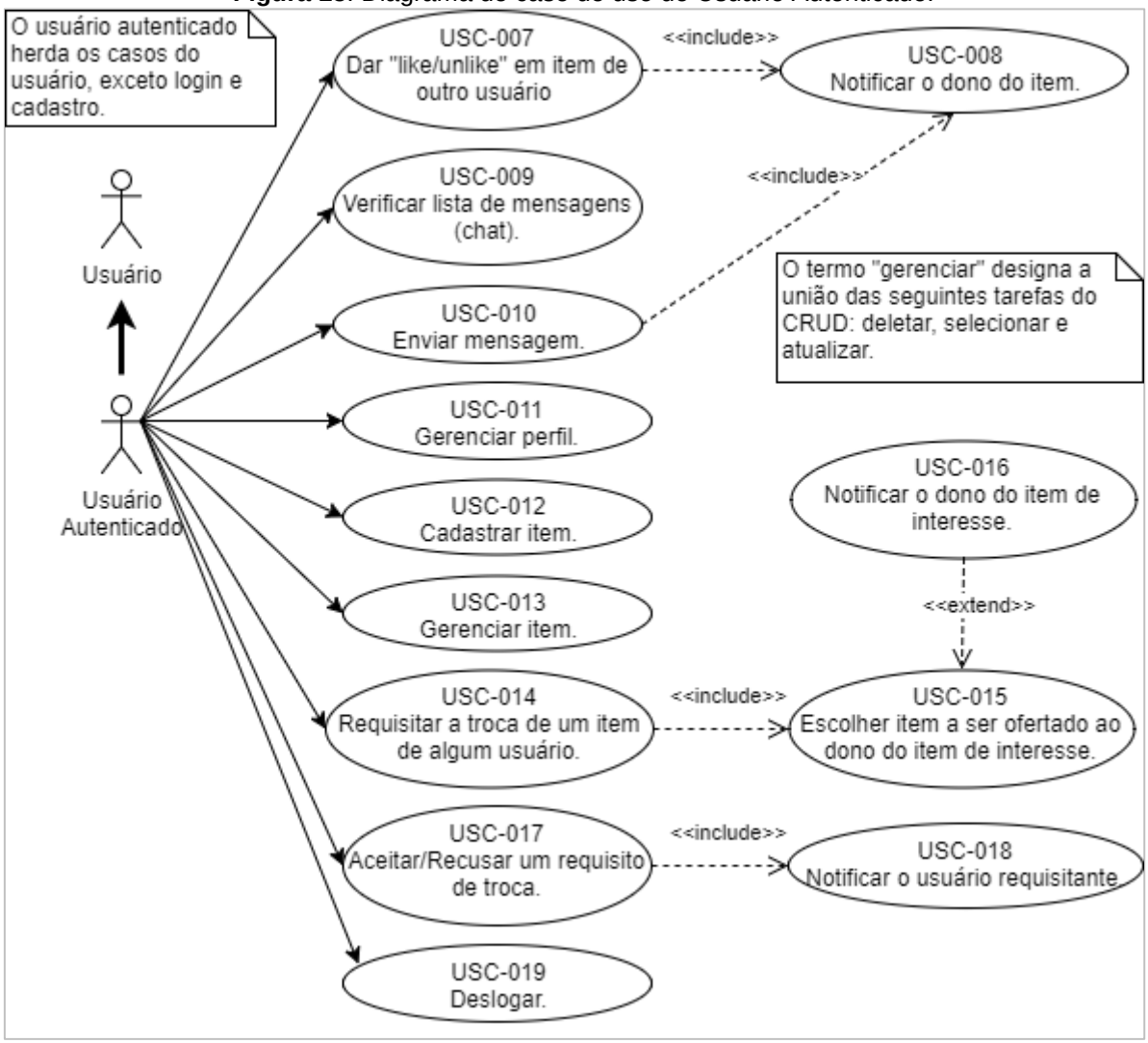

**Fonte**: Autor, 2017.

### 4.1.2 Usuário Autenticado

Quando autenticado, o usuário, além da posse de novas funcionalidades – correspondentes aos principais objetivos do trabalho, operantes em prol da efetuação

da troca – abarca as descritas na ilustração acima. A Figura 15 descreve tais funcionalidades que deram margem ao levantamento dos dados necessários para todas as operações ilustradas com os diagramas.

**Figura 15**: Diagrama de caso de uso do Usuário Autenticado.




**Fonte**: Autor, 2017.

## 4.2 REGRA DE NEGÓCIO

A seção de troca – cerne da plataforma – requereu um escrutínio, pois envolve uma conexão entre dois usuários e algumas variáveis. Para resolver essa questão, foi desenvolvida uma simples regra de negócio (Tabela 2), que delineia o ciclo de troca e o destino dos itens trocados.

**Tabela 2**: Regra de negócio da troca.

| Código | Nome | Descrição |
|---|---|---|
| [RN1.0] | Requisito de troca | A requisição somente é possível caso o usuário selecione o item-alvo de outro usuário e, em seguida, possua algum item para oferecer-lhe em troca, selecionando-o; caso contrário, não será possível requisitá-lo. |
| [RN1.1] | Notificação de troca | Após o requisitado receber uma notificação de troca, pode aceitá-la ou recusá-la. Caso recuse, não ficará em seu histórico de trocas; caso contrário, ficará em exibição e não será possível o regresso. O requisitante será notificado conforme ambas as opções. |
| [RN1.2] | Ciclo de vida da troca | Quando requisitada, o registro de troca é criado como pendente, caso aceito, constará como finalizado; caso contrário, como recusado. Conquanto não seja finalizado, o usuário requisitante pode cancelar o requisito deletando-o da base de dados. |
| [RN1.3] | Itens trocados | Os itens que estiverem em uma troca finalizada continuarão sendo exibidos ao público nas pesquisas, porém, em último nos resultados, tarjados como “trocado” e, evidentemente, indisponíveis para relações. |

**Fonte**: Autor, 2017.

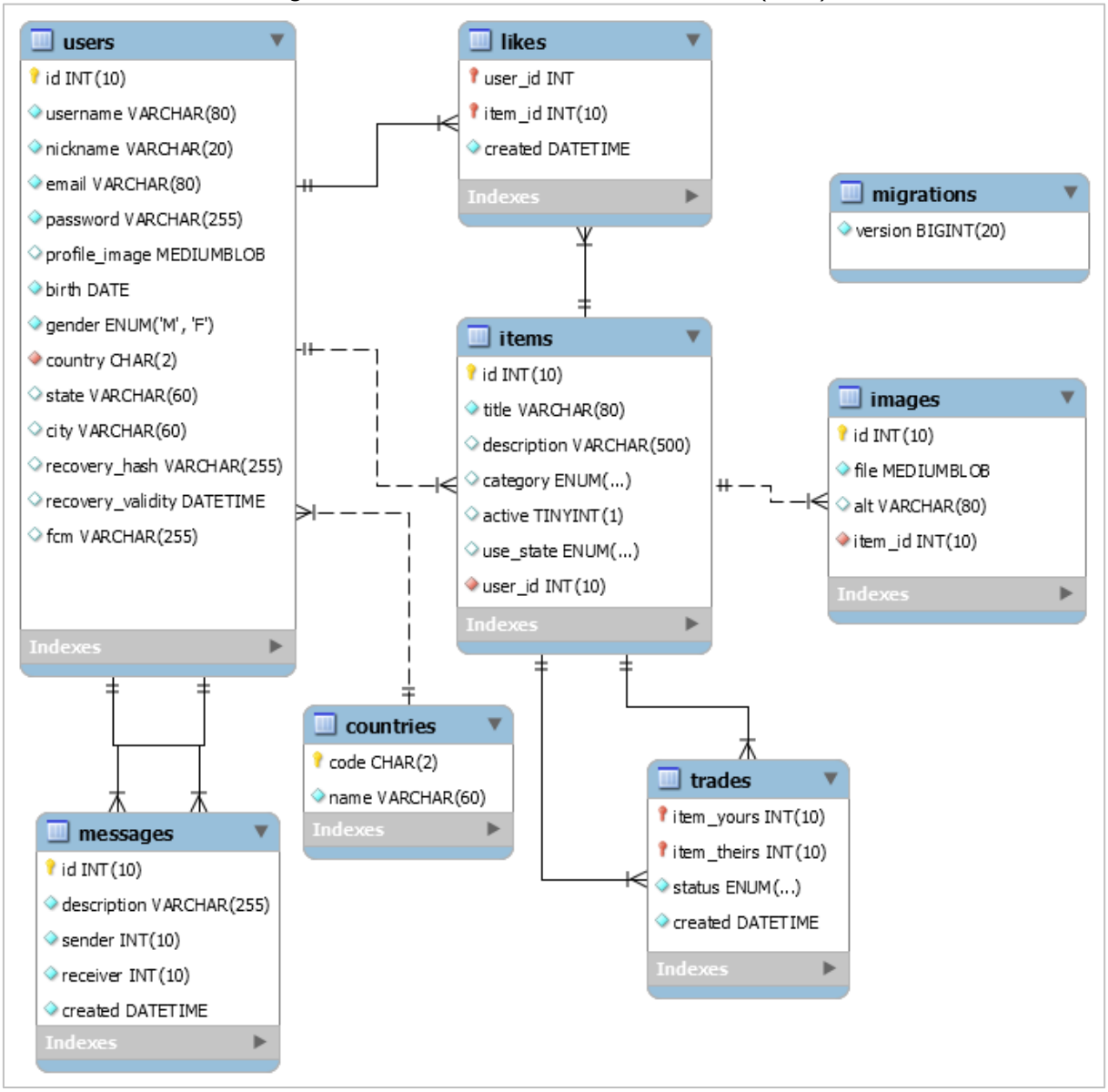
## 4.3 BANCO DE DADOS

Lançamento das informações pertinentes aos dados que serão armazenados e gerenciados pela plataforma.

### 4.3.1 Modelo Entidade Relacionamento

Uma modelagem que conceitua e reivindica as entidades necessárias à solidificação das regras de negócios e casos de uso, para que o sistema opere atendendo às premissas da pesquisa. A partir desta representação abstrata (Figura 16), foi possível dar continuidade aos passos ulteriores: codificação.

**Figura 16**: Modelo Entidade Relacionamento (MER).




**Fonte**: Autor, 2017.

### 4.3.2 Dicionário de Dados

Aqui as entidades apresentadas na figura anterior aparecem detalhadas.

#### 4.3.2.1 Usuário

Esta é usada para armazenar os dados dos usuários registrados no sistema (Tabela 3). O usuário pode usufruir da plataforma sem autenticação, porém, precisá-la-á para interagir integralmente com o sistema.

**Tabela 3**: Entidade usuário.

| **Users** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| id | 10 | INT | PK | Identificador que torna o registro deste usuário único nas operações internas do sistema (back-end). |
| username | 80 | VARCHAR | | Nome completo do usuário. |
| nickname | 20 | VARCHAR | UK | Apelido de usuário usado para identificá-lo de maneira mais agradável nas operações externas do sistema (front-end). |
| email | 80 | VARCHAR | UK | E-mail que o usuário usará para efetuar a autenticação no sistema e recuperar sua senha, caso necessário. |
| password | 255 | VARCHAR | | Senha do usuário. |
| profile_image | - | MEDIUM BLOB | | Imagem do perfil do usuário. |
| birth | - | DATE | | Data de nascimento do usuário. |
| gender | - | ENUM('M','F') | | Gênero do usuário. |
| country | 2 | CHAR | FK | Sigla do país do usuário. |
| state | 60 | VARCHAR | | Estado do usuário. |
| city | 60 | VARCHAR | | Cidade do usuário. |
| recovery_hash | 255 | VARCHAR | | Chave que é gerada e enviada ao e-mail do usuário para execução da recuperação de senha. |
| recovery_validity | - | DATETIME | | Prazo de validade da chave gerada para recuperação de senha. |
| fcm | 255 | VARCHAR | | Token gerado pela API do google para o envio de notificações push. |
| **RESTRIÇÕES** | | | | |
| id[not null; unsigned; PRIMARY], username[not null], nickname[not null; UNIQUE], email[not null, UNIQUE], password[not null], birth[not null], gender[not null], country[not null; FOREIGN] | | | | |
| **RELACIONAMENTOS** | | | | |
| Users.country = Countries.code | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.2 Item

Registra as informações de cada item inserido pelo usuário:

**Tabela 4**: Entidade item.

| **Items** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| id | 10 | INT | PK | Identificador que torna o registro deste item único nas operações do sistema. |
| title | 80 | VARCHAR | | Título do item. |
| description | 500 | VARCHAR | | Descrição do item. |
| category | | ENUM(…) | | Categoria do item. |
| active | - | BOOLEAN | | Campo que identifica se o item está ativo ou não. Quando ativo, fica disponível para troca. |
| use_state | - | ENUM('NOVO','USADO','SEMI-NOVO | | Estado de uso do item. |
| user_id | 10 | INT | FK | Identificador que vincula o item ao dono (usuário). |
| **RESTRIÇÕES** | | | | |
| id[not null; unsigned; PRIMARY], title[not null], user_id[not null; FOREIGN] | | | | |
| **RELACIONAMENTOS** | | | | |
| Items.user_id = Users.id | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.3 Migração

Entidade que atende aos chamados do *framework* CodeIgniter para realizar o versionamento e manutenção do banco de dados através da ferramenta *migration*:

**Tabela 5**: Entidade migração.

| **Migrations** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| version | 20 | BIGINT | FK | Campo que armazena o valor da versão. |
| **RESTRIÇÕES** | | | | |
| Version[not_null] | | | | |
| **RELACIONAMENTOS** | | | | |
| - | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.4 Imagens do Item

Um item pode ter mais de uma imagem – no máximo 5. Sendo assim, é necessária uma tabela específica que registre imagem(ns) e vincule-a(s) ao item em questão:

**Tabela 6**: Entidade imagens do item.

| **Images** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| id | 10 | INT | PK | Identificador que torna o registro desta imagem único nas operações do sistema. |
| file | - | MEDIUMBLOB | | O código binário que gera a imagem. |
| alt | 80 | VARCHAR | | Descrição da imagem utilizada para fins de acessibilidade. |
| item_id | 10 | INT | FK | Identificador que vincula a imagem ao item. |
| **RESTRIÇÕES** | | | | |
| id[not null; unsigned; PRIMARY], image_blob[not null], item_id[not null; FOREIGN] | | | | |
| **RELACIONAMENTOS** | | | | |
| Item_images.item_id = Items.id | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.5 Like

Entidade responsável por controlar os "*likes"* efetuados pelo usuário em um determinado item (Tabela 7). Essa ação somente é possível com autenticação (login).

**Tabela 7**: Entidade like.

| **Likes** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| user_id | 10 | INT | FK | Identificador que vincula o usuário ao like. |
| item_id | 10 | INT | FK | Identificador que vincula o item ao like. |
| created | - | DATETIME | | Data que a ação foi registrada. |
| **RESTRIÇÕES** | | | | |
| PRIMARY(user_id[not null; unsigned; FOREIGN], item_id[not null; unsigned; FOREIGN]), created[not null; CURRENT TIMESTAMP] | | | | |
| **RELACIONAMENTOS** | | | | |
| Likes.user_id = Users.id; Likes.item_id = Items.id | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.6 Troca

Tabela que registra a troca realizada por dois usuários, sendo um o interessado em algum item disponível e outro o possuinte:

**Tabela 8**: Entidade troca.

| **Trades** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| item_theirs | 10 | INT | FK | Identificador que vincula o item-alvo de interesse à troca. |
| item_yours | 10 | INT | FK | Identificador que vincula o item a ser ofertado ao item-alvo de interesse à troca. |
| created | - | DATETIME | | Horário que a relação de troca foi efetuada pelos usuários. |
| status | - | ENUM('DONE' ,'PENDENT',' REFUSED') | | Campo que armazena o status da troca, podendo ser aceito, recusado e pendente. |
| **RESTRIÇÕES** | | | | |
| PRIMARY(user_id[not null; unsigned; FOREIGN], item_id[not null; unsigned; FOREIGN]), done[not null], created[not null; CURRENT TIMESTAMP] | | | | |
| **RELACIONAMENTOS** | | | | |
| Trades.item_theirs = Items.id, Trades.item_yours = Items.id | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.7 Mensagens

Entidade que armazena as mensagens dos usuários:

**Tabela 9**: Entidade mensagens.

| **Messages** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| id | 10 | INT | PK | Identificador que torna a mensagem única. |
| description | 255 | VARCHAR | | Corpo da mensagem. |
| sender | 10 | INT | FK | Identificador que vincula o remetente à mensagem. |
| receiver | 10 | INT | FK | Identificador que vincula o destinatário à mensagem. |
| created | - | DATETIME | | Horário que a mensagem foi enviada. |
| **RESTRIÇÕES** | | | | |
| id[not null; unsigned; PRIMARY], description[not null], sender[not null; FOREIGN], receiver[not null; FOREIGN], created[not null; CURRENT TIMESTAMP] | | | | |
| **RELACIONAMENTOS** | | | | |
| Messages.sender = Users.id, Messages.receiver = Users.id | | | | |

**Fonte**: Autor, 2017.

#### 4.3.2.8 Países

Tabela que armazena todos os países e seus respectivos códigos que serão vinculados ao usuário cadastrado:

**Tabela 10**: Entidade país.

| **Countries** | | | | |
|---|---|---|---|---|
| **Tipo de campo** | **Tamanho do campo** | **Tipo de dados** | **Tipo de chave** | **Descrição** |
| code | 2 | CHAR | FK | Código/sigla que torna o registro deste país único. |
| name | 60 | VARCHAR | | Nome do país. |
| **RESTRIÇÕES** | | | | |
| code[not null; PRIMARY], name[not null] | | | | |
| **RELACIONAMENTOS** | | | | |
| - | | | | |

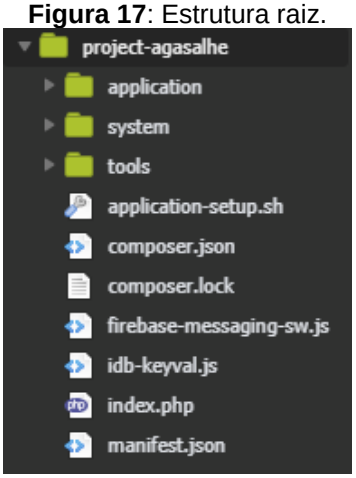
**Fonte**: Autor, 2017.

## 4.4 BACK-END

Aqui são dispostos os principais feitos da camada de negócio que dão existência computacional às especificações dos casos de uso – inclusive abarca a camada de visualização em sua estrutura –, regra de negócio e dados que serão utilizados pelo banco de dados, além da conexão com o próprio banco.

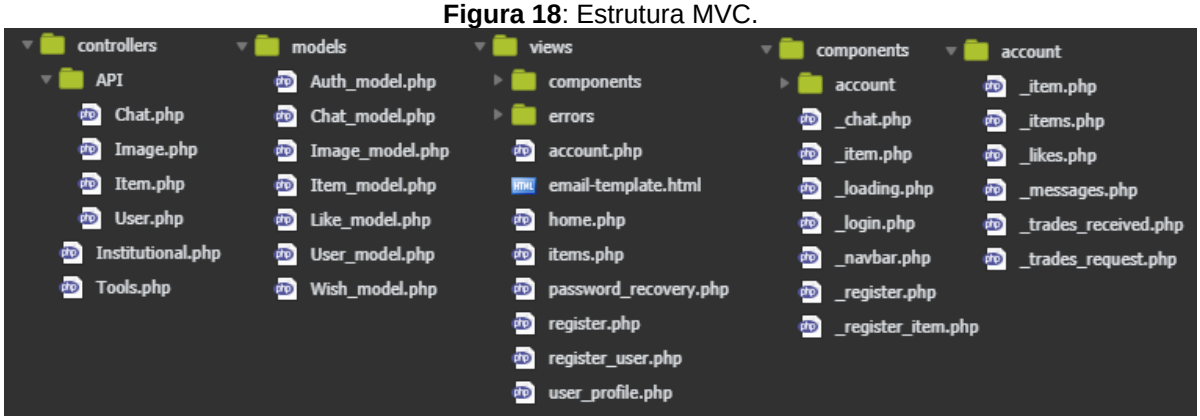
A Figura 17 mostra a pasta raiz do projeto: a estrutura do *framework Code Igniter*. Sobre ela, tanto a camada de negócio como a de visualização são codificadas, porém, seguindo o padrão *MVC* (Figura 18) e desacoplando o máximo possível com o padrão *RESTFul*.

**Figura 17**: Estrutura raiz.




**Fonte**: Autor, 2017.

**Figura 18**: Estrutura MVC.




**Fonte**: Autor, 2017.

### 4.4.1 Curtidas

As funções nas Figuras 19 e 20, *Controller* e *Model*, efetuam a inserção de um novo *like*. Para tal, ele precisa somente receber o identificador do item selecionado, validá-lo e mesclá-lo ao identificador do respectivo usuário. Lembrando que o identificador do usuário não precisa ser enviado nas requisições, pois ele se encontra em sua sessão de acesso e é resgatado após o método de verificação. Além dessa função, a regra de *like* possui *like_get()* – que lista as curtidas – e *like_del()* relativo ao "*deslike*".

**Figura 19**: Controller da inserção do *like*.

```php
public function like_post()
{
    $USER = $this->authentication->verify_authentication();

    $this->form_validation->set_rules('item_id', 'Item ID', 'trim|required|integer|max_length[10]|greater_than[-1]');

    if ($this->form_validation->run() === FALSE)
    {
        $error = $this->form_validation->error_array();
        $this->set_response($error, REST_Controller::HTTP_UNPROCESSABLE_ENTITY);
    }
    else
    {
        $response = $this->like_model->set_like($USER['id']);
        $response ? $this->set_response(null, REST_Controller::HTTP_CREATED) : $this->set_response(NULL, REST_Controller::HTTP_UNAUTHORIZED);
    }

}
```

**Fonte**: Autor, 2017.

**Figura 20**: Model da inserção do *like*.

```php
public function set_like($ID)
{
    $query = $this->db->get_where('likes', array("item_id" => $this->input->post('item_id'), "user_id" => $ID));
    if($query->num_rows() > 0)
    {
        return false;
    }
    $data = array
    (
        'user_id' => $ID,
        'item_id' => $this->input->post('item_id')
    );

    $this->db->insert('likes', $data);
    return true;
}
```

**Fonte**: Autor, 2017.

### 4.4.2 Trocas

A troca é iniciada pelo usuário que, após selecionar o item de outrem e escolher seu item a ofertar-lhe, conclui a ação. Feito isso, um registro com o *status* “pendente” é aberto e o requisitado é notificado. Ele pode aceitar a solicitação e marcar aquele registro como finalizado, ou recusá-la. As Figuras 21 e 22 demonstram o processo da criação do requisito de troca. Algumas verificações são feitas para não ocorrer trocas inválidas ou tentativas maliciosas que burlem a regra do sistema. Fora o método da figura 21, o *controller* também possui o método *trade_put()* que confirma positivamente o requisito de troca, *refuse_trade_post()* que recusa-o, *trade_delete()* que exclui o requisito de troca enquanto ele ainda não for finalizado e *trade_get()* que seleciona as trocas registradas.

**Figura 21**: Controller da inserção de trocas.

```php
# Do a request for trading.
public function trade_post()
{
    $USER = $this->authentication->verify_authentication();
    $ITEM_THEIRS = $this->authentication->verify_parameter('item_theirs');
    $ITEM_YOURS = $this->authentication->verify_parameter('item_yours');

    $response = $this->item_model->set_trade($ITEM_YOURS, $ITEM_THEIRS, $USER['id']);
    $response[0] ? $this->set_response(['status' => TRUE, 'message' => $response[1]], REST_Controller::HTTP_CREATED) :
                   $this->set_response(['status' => FALSE, 'message' => $response[1]], REST_Controller::HTTP_UNPROCESSABLE_ENTITY);
}
```

**Fonte**: Autor, 2017.

**Figura 22**: Model da inserção de trocas.

```php
public function set_trade($ITEM_YOURS, $ITEM_THEIRS, $ID)
{
    if($ITEM_YOURS === $ITEM_THEIRS)
    {
        return [false, 'You cannot trade the item by itself.'];
    }
    if(!$this->isValidItemId($ITEM_THEIRS) && $this->isValidItemId($ITEM_YOURS))
    {
        return [false, 'Only valid IDs are acceptable.'];
    }
    if($this->isMyOwnItem($ITEM_THEIRS, $ID))
    {
        return [false, 'You cannot trade an item which is already yours.'];
    }
    # Now is checking if the item is really yours.
    if(!$this->isMyOwnItem($ITEM_YOURS, $ID))
    {
        return [false, 'You cannot trade an item which is not yours.'];
    }

    if($this->isAlreadyRequested($ITEM_THEIRS, $ITEM_YOURS)[1])
    {
        $r = $this->isAlreadyRequested($ITEM_THEIRS, $ITEM_YOURS)[0];
        return [false, "You've already requested for this item before, its status is: ". $r];
    }
    if(!$this->isAlreadyTraded($ITEM_YOURS, $ITEM_THEIRS)[0])
    {
        return [false, $this->isAlreadyTraded($ITEM_YOURS, $ITEM_THEIRS)[1]];
    }

    $insert = "INSERT INTO trades VALUES (?, ?, NOW(), 'PENDING')";
    $this->db->query($insert, array(
        'item_theirs' => $ITEM_THEIRS,
        'item_yours' => $ITEM_YOURS));

    return [true, 'Successful request'];
}
```

**Fonte**: Autor, 2017.

### 4.4.3 Itens

Os itens cadastrados na plataforma podem ser pesquisados e listados por categoria, estado, cidade, título e estado de uso, conforme indica o código das figuras 23 e 24:

**Figura 23**: Controller da pesquisa de itens.

```php
# Get user item(s)
public function index_get()
{
    $ID = $this->get('id');
    $filterResult = $this->input->get('filter-result');

    if($filterResult != NULL)
    {

        $FILTER_PARAMS = array(
            'category' => $this->input->get('category'),
            'state' => $this->input->get('state'),
            'city' => $this->input->get('city'),
            'title' => $this->input->get('title')
        );

        $useState = $this->input->get('use-state');
        $useState == 'NOVO' || $useState == 'SEMI-NOVO' ? array_push($FILTER_PARAMS, "use_state", $useState) : NULL;

        $FILTER_PARAMS = array_filter($FILTER_PARAMS);

        $items = $this->item_model->get_filter_items($FILTER_PARAMS);
        $error = ['status' => FALSE, 'message' => 'No items were found'];
        $items ? $this->response($items, REST_Controller::HTTP_OK) : $this->response($error, REST_Controller::HTTP_NOT_FOUND);

    }
    elseif($ID == NULL)
    {
        $items = $this->item_model->get_item();
        $error = ['status' => FALSE, 'message' => 'No items were found'];
        $items ? $this->response($items, REST_Controller::HTTP_OK) : $this->response($error, REST_Controller::HTTP_NOT_FOUND);
    }
    else
    {
        $ID = (int) $ID;
        $ID <= 0 ? $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST) : $item = $this->item_model->get_item($ID, "id");

        $error = ['status' => FALSE, 'message' => 'Item could not be found'];
        !empty($item) ? $this->set_response($item, REST_Controller::HTTP_OK) : $this->set_response($error, REST_Controller::HTTP_NOT_FOUND);
    }
}
```

35:15 PHP Spaces: 4

**Fonte**: Autor, 2017.

**Figura 24**: Model da pesquisa de itens.

```php
# Get itens by paramteres
public function get_filter_items($FILTER_PARAMS)
{

    $this->db->select(' u.id as user_id, u.nickname, i.id as item_id, i.title');
    $this->db->from('items as i');
    $this->db->join('users as u', 'u.id = i.user_id', 'inner');
    foreach($FILTER_PARAMS as $key => $value)
    {
        $this->db->like($key, $value);
    }


    $query = $this->db->get()->result_array();
    $results = array();
    foreach($query as $value)
    {
       $images = $this->get_images($value['item_id']);
       $likes  = $this->count_likes($value['item_id']);
       $value['images'] = $images;
       $value['qt_likes'] = $likes[0]['qt_item_likes'];
       $results[] = $value;
    }
    return $results;

}
```

**Fonte**: Autor, 2017.

Os usuários podem cadastrar múltiplos itens, sendo um de cada vez. As funções das Figuras 25 e 26 demonstram a validação das informações do item, inserção no banco de dados e o registro de imagens que está detalhado no capitulo abaixo.

**Figura 25**: Controller do cadastro de item.

```php
# Create an item
public function index_post()
{
    $USER = $this->authentication->verify_authentication();
    $this->form_validation->set_rules('title', 'title', 'trim|required|min_length[2]|max_length[80]');
    $this->form_validation->set_rules('description', 'description', 'trim|required|min_length[2]|max_length[500]');
    $this->form_validation->set_rules('use_state', 'use state', 'trim|required|in_list[NOVO,USADO,SEMI-NOVO]');
    $this->form_validation->set_rules('category', 'category', 'trim|required|in_list[CATEGORIES]');
    if ($this->form_validation->run() === FALSE)
    {
        $error = $this->form_validation->error_array();
        $this->set_response($error, REST_Controller::HTTP_UNPROCESSABLE_ENTITY);
    }
    else
    {
        $response = $this->item_model->set_item($USER['id']);
        $item_ref = $this->db->insert_id();
        for($i=0; $i < count($_FILES['image']['name']) ; $i++) {
            $image = [
                'name' => $_FILES['image']['name'][$i],
                'type' => $_FILES['image']['type'][$i],
                'tmp_name' => $_FILES['image']['tmp_name'][$i],
                'error' => $_FILES['image']['error'][$i],
                'size' => $_FILES['image']['size'][$i]
            ];
            $image_alt = isset($this->input->post('image_alt')[$i]) ? $this->input->post('image_alt')[$i] : "No alt text to this image.";
            $response = $this->image_model->set_image($image, $image_alt, $item_ref);
        }
        $response ? $this->set_response(null, REST_Controller::HTTP_CREATED) : $this->set_response(NULL, REST_Controller::HTTP_UNAUTHORIZED);
    }

}
```

**Fonte**: Autor, 2017.

**Figura 26**: Model do cadastro de item.

```php
# Insert Item
public function set_item($USER_ID)
{
    $data = array
    (
        'title' => $this->input->post('title'),
        'description' => $this->input->post('description'),
        'use_state' => $this->input->post('use_state'),
        'category' => $this->input->post('category'),
        'user_id' => $USER_ID
    );

    return $this->db->insert('items', $data);
}
```

**Fonte**: Autor, 2017.

### 4.4.4 Imagens

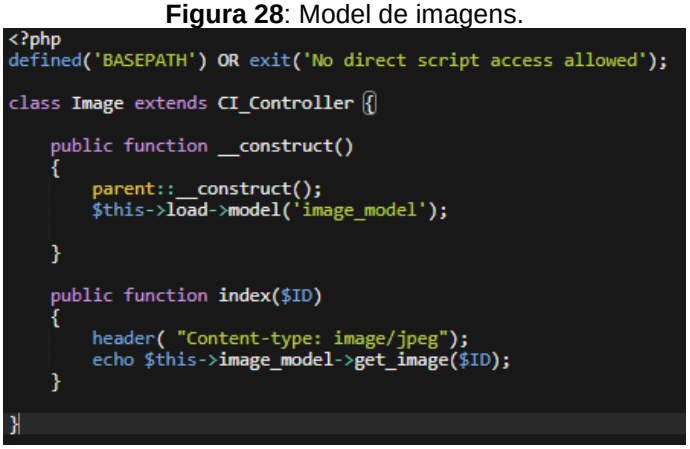
O gerenciamento de imagens – tanto de perfil como do item – foi arquitetado para manipulá-las como código binário salvo no banco de dados *(BLOB)*. Para fácil legibilidade e manutenção, foi criado Model e Controller (Figuras 27 e 28), para realizar os propósitos pertinentes a elas.

**Figura 27**: Controller de imagens.

```php
<?php
class Image_model extends CI_Model {

    public function __construct()
    {
        $this->load->database();
    }

    public function set_image($IMG_FILE, $ALT, $REF)
    {
        $imageName = time().'.jpg';
        if (move_uploaded_file($IMG_FILE['tmp_name'], $imageName)) {
            $data = array
            (
                'file' => file_get_contents($imageName),
                'alt' => $ALT,
                'item_id' => $REF
            );
            return $this->db->insert('images', $data);
            unlink($imageName);
        }
        else{
            echo false;
        }

    }

    public function get_image($ID)
    {
        $this->db->select('*');
        $this->db->from('images');
        $this->db->where('id', $ID);
        $image = $this->db->get()->row()->file;
        // header( "Content-type: image/jpeg");
        return $image;
    }

    public function get_references($REF)
    {
        $this->db->select('*');
        $this->db->from('images');
        $this->db->where('item_id', $REF);
        $references = $this->db->get()->result_array();
        return $references;
    }
}
?>
```

**Fonte**: Autor, 2017.

**Figura 28**: Model de imagens.



```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Image extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->load->model('image_model');

    }

    public function index($ID)
    {
        header( "Content-type: image/jpeg");
        echo $this->image_model->get_image($ID);
    }

}
```

**Fonte**: Autor, 2017.

### 4.4.5 Mensagens

As mensagens são agrupadas por conversas num chat. O usuário, quando logado, pode enviá-las a qualquer outro através do perfil ou de seu item alvo.

A Figura 29 indica o método que recebe a mensagem no controlador do chat. Em seguida, a 30 indica a inserção no banco de dados.

**Figura 29**: Controller do chat.

```
# Register an user
public function index_post()
{
    $USER = $this->authentication->verify_authentication();
    $this->form_validation->set_rules('receiver', 'receiver', 'trim|required|integer');
    $this->form_validation->set_rules('description', 'password', 'required|max_length[255]');

    if ($this->form_validation->run() === FALSE || $USER['id'] == $this->input->post('receiver'))
    {
        $error = $this->form_validation->error_array();
        $this->set_response($error, REST_Controller::HTTP_UNPROCESSABLE_ENTITY);
    }
    else
    {
    #   CREATED (201) being the HTTP response code:
        $created = $this->chat_model->set_chat($USER);
        $MID  = $this->db->insert_id();
        $this->set_response(['message' => 'Message sent successfully', 'created' => $created, 'id' => $MID], REST_Controller::HTTP_CREATED);
    }
}
```

**Fonte**: Autor, 2017.

**Figura 30**: Model do chat.

```
public function set_chat($sender)
{
    $date = date('Y-m-d H:i:s');
    $data = array
    (
        'receiver' => $this->input->post('receiver'),
        'description' => $this->input->post('description'),
        'sender' => $sender['id'],
        'created' => $date
    );

    $this->db->insert('messages', $data);

    $date = $this->out_date($date);

    $this->load->library('FCMNotify');
    $this->load->model('user_model');
    $this->fcmnotify->send(
        $this->user_model->get_fcm($this->input->post('receiver')),
        $sender['username'] . " enviou uma mensagem.",
        $this->input->post('description'),
        base_url() . "API/image/profile/" . $sender['id'],
        array('created' => $date, 'id' => $this->db->insert_id(), 'sender_id' => $sender['id'])
    );

    return $date;
}
```

**Fonte**: Autor, 2017.

Percebe-se nessa última figura o carregamento da classe “*fcmnotify*”: responsável por efetuar uma requisição ao servidor da Google, solicitando o envio da notificação *push* através do firebase (Figura 31). Todas as notificações no projeto seguem esse padrão. O front-end, como está descrito no próximo subcapítulo, possui um método à escuta dessas notificações.

**Figura 31**: Classe firebase para enviar notificações.

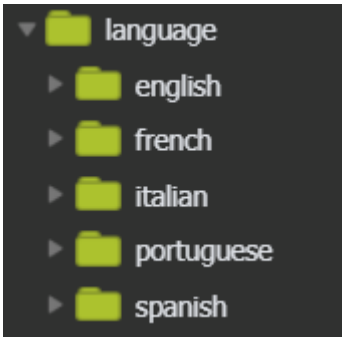
```php
<?php
    define('AUTH_KEY', "AAAAtFRpU1I:APA91bG5OqP7dwbJclOdS7JMyUFQJYhpkVGuFdNT85ILS13YHJkgE7HDV8HhZ
    class FCMNotify{

        public function send($tokenFcm, $title, $body, $icon, $data = null, $url = null){
            $data['title'] = $title;
            $data['body']  = $body;
            $data['sound'] = 'default';
            $data['icon'] = $icon;
            $data['action_click'] = $url;

            $fields = array(
                'to'           => $tokenFcm,
                'data'         => $data
            );
            $headers = array(
                'Authorization: key='.AUTH_KEY,
                'Content-Type: application/json'
            );
            $ch = curl_init();
            curl_setopt($ch,CURLOPT_URL, 'https://fcm.googleapis.com/fcm/send');
            curl_setopt($ch,CURLOPT_POST, true );
            curl_setopt($ch,CURLOPT_HTTPHEADER, $headers);
            curl_setopt($ch,CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch,CURLOPT_SSL_VERIFYPEER, false);
            curl_setopt($ch,CURLOPT_POSTFIELDS, json_encode($fields));
            $result = curl_exec($ch );
            curl_close( $ch );
        }
    }
?>
```

**Fonte**: Autor, 2017.

### 4.4.6 Internacionalização

Foram escolhidos cinco idiomas para a plataforma, sendo eles Português Brasileiro, Inglês, Italiano, Espanhol e Francês (Figura 32). Caso o navegador do usuário não esteja em nenhuma dessas linguagens, a página será carregada em Inglês.

**Figura 32**: Estrutura da internacionalização.




**Fonte**: Autor, 2017.

Como a imagem acima indica, para cada idioma é necessária uma pasta e dentro da pasta pode-se criar arquivos que possuam termos-chave e sua respectiva tradução:

**Figura 33**: Arquivo de configuração de idioma.

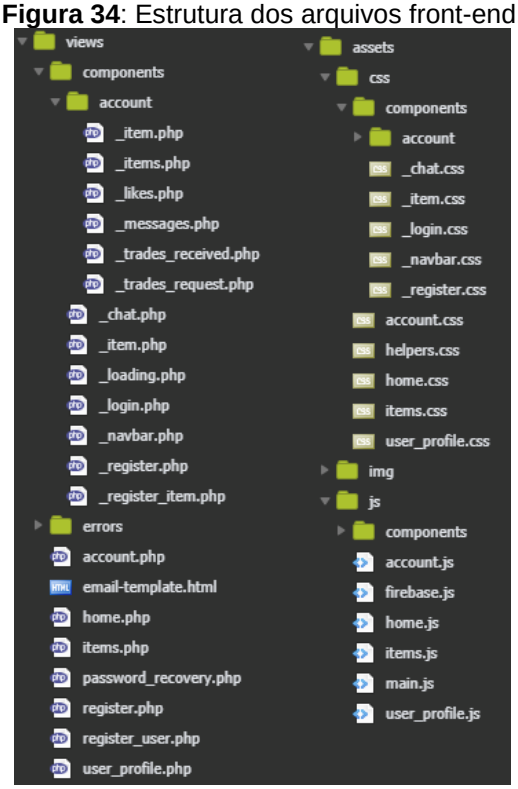
```php
<?php

$lang['main_nav_item_1'] = 'Items';
$lang['main_nav_item_2'] = 'How it works';
$lang['main_nav_item_3'] = 'About';
$lang['main_nav_item_4'] = 'Get Started';
$lang['main_nav_item_5'] = 'Log in';

?>
```

**Fonte**: Autor, 2017.

## 4.5 FRONT-END

Codificação realizada na camada de visualização, controlando *inputs*, *outputs* e a disposição do *layout*. A Figura 34 mostra a estrutura da pasta *view* responsável pelas telas e a *assets* que cuida dos arquivos dinâmicos que estão dispostos publicamente.

**Figura 34**: Estrutura dos arquivos front-end.




**Fonte**: Autor, 2017.

### 4.5.1 JavaScript

Todas as requisições ao servidor utilizaram o *AJAX* do *JQuery*. O *vue* permitiu a criação de instâncias que agrupam métodos, dados e um vínculo a algum elemento

*HTML*. Dessa forma, o código fica organizado por componentes, cada um possuindo as suas especificações para o funcionamento. Como mostra a Figura 35, a chave "*el*" armazena o "*id*" do elemento *HTML* que sofrerá modificações, o "*data*" define o "*model*" responsável pelos dados de *input* e *output* e "*methods*" armazena as funções necessárias.

**Figura 35**: Estrutura do objeto Vue.

```
function bindData(){
    var itemsController = new Vue({
      el: '#itemsController',
      data: {
        show404: false,
        itemsCount: 0,
        item: {
            title: null,
            authorName: null,
            authorImage: null,
            authorLink: null,
            category: null,
            useState: null,
            likes: null,
            description: null
        },
        items: []
      },
        methods:{
            searchBySomething: function(){
```

**Fonte**: Autor, 2017.

Os métodos são responsáveis por realizar a chamada ao *end-point* e popular o modelo de dados, retornado em *JSON*, que será exibido no respectivo elemento *HTML* vinculado:

**Figura 36**: Métodos do objeto Vue.

```
methods:{
    searchBySomething: function(){
        var query = "/API/item?filter-result=true"
        var title = $('#search-by-title').val();
        title.length ? query = query + "&title=" + title.replace(/\s/g,'') : null;
        var category = $('.search-by-category .active.selected span').text();
        category.length ? query = query + "&category=" + category : null;
        var useState = $('.search-by-use-state .active.selected span').text();
        useState.length ? query = query + "&use-state=" + useState : null;
        var state = $('#search-by-state').val();
        state.length ? query = query + "&state=" + state.replace(/\s/g,'') : null;
        var city = $('#search-by-city').val();
        city.length ? query = query + "&city=" + city.replace(/\s/g,'') : null;
        this.populateController(query)
    },
    populateController: function(path){
        var that = this;
        path = path || "/API/item";
        this.items = [];
        $.ajax({
            method: "GET",
            url: path,
            complete: function(jqXHR, textStatus){
                switch (jqXHR.status) {
                    case 200:
                        jqXHR.responseJSON.map((e) => { that.itemBuilder(e) });
                        that.itemsCount = jqXHR.responseJSON.length;
                        break;
                    case 404:
                        that.show404 = true;
                        that.itemsCount = 0;
                        console.log('404');
                        break;
                    default:
                        console.log("Other error")
                }
            }
        })
    },
```

**Fonte**: Autor, 2017.

A aplicação do *firebase* necessitou de algumas configurações. Ao entrar no sistema, o usuário é requisitado pelo uso de notificações, e, caso ele permita, um *token* – que identifica a máquina para onde a notificação deverá ser enviada – será retornado para ser salvo junto no servidor (Figura 37). Foi necessário também um observador que fique atento às novas notificações para assim exibi-las na tela do dispositivo, independente de o site estar fechado (Figura 38).

**Figura 37**: Configuração do firebase messaging.

```
// Initialize Firebase
var config = {
    apiKey: "AIzaSyDoT3ObbiBKe459nJIMJsMt2cLhbpmjF4s",
    authDomain: "trocaqui-54785.firebaseapp.com",
    databaseURL: "https://trocaqui-54785.firebaseio.com",
    projectId: "trocaqui-54785",
    storageBucket: "trocaqui-54785.appspot.com",
    messagingSenderId: "774510302034"
};

firebase.initializeApp(config);
const messaging = firebase.messaging();

$(function(){
    messaging.getToken().then(function (token){
        if(token === null){
            $('#notification-modal').modal('open');
            $('.decide-notification').click(function(){
                messaging.requestPermission().then(function(){
                    console.log("Notificação permitida.")
                    messaging.getToken().then(function(token){
                        saveToken(token);
                    })
                }).catch(function(err) {
                  console.log('Notificação não permitida.', err);
                });
            });
        }
    }).catch(function(err){
        console.log('Notificação não permitida.', err);
    });

});
```

**Fonte**: Autor, 2017.

**Figura 38**: Configuração do observador de notificações.

```
messaging.setBackgroundMessageHandler(function(payload){
    const title= payload.data.title;
    const options = {
        body: payload.data.status,
        icon: payload.data.icon,
        sound: payload.data.sound
    }
    return self.registration.showNotification(title, options);
});
```

**Fonte**: Autor, 2017.

Contudo, essas notificações são capturadas somente em *foreground* (primeiro plano), mas necessitando também do *background* (segundo plano). E, para isso, foi preciso um arquivo, na raiz da pasta, que manipule um *service worker* (Figura 39).

**Figura 39**: Service Worker do Firebase.

```
importScripts('https://www.gstatic.com/firebasejs/3.9.0/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/3.9.0/firebase-messaging.js');
importScripts('idb-keyval.js');

var config = {
    apiKey: "AIzaSyDoT3ObbiBKe459nJIMJsMt2cLhbpmjF4s",
    authDomain: "trocaqui-54785.firebaseapp.com",
    databaseURL: "https://trocaqui-54785.firebaseio.com",
    projectId: "trocaqui-54785",
    storageBucket: "trocaqui-54785.appspot.com",
    messagingSenderId: "774510302034"
};

firebase.initializeApp(config);

var messaging = firebase.messaging();

messaging.setBackgroundMessageHandler(function(payload){
    idbKeyval.set('newMessage', JSON.stringify(payload.data));
    console.log(payload);
    const title= payload.data.title;
    const options = {
        body: payload.data.body,
        icon: payload.data.icon
    }
    return self.registration.showNotification(title, options);
});
```

**Fonte**: Autor, 2017.

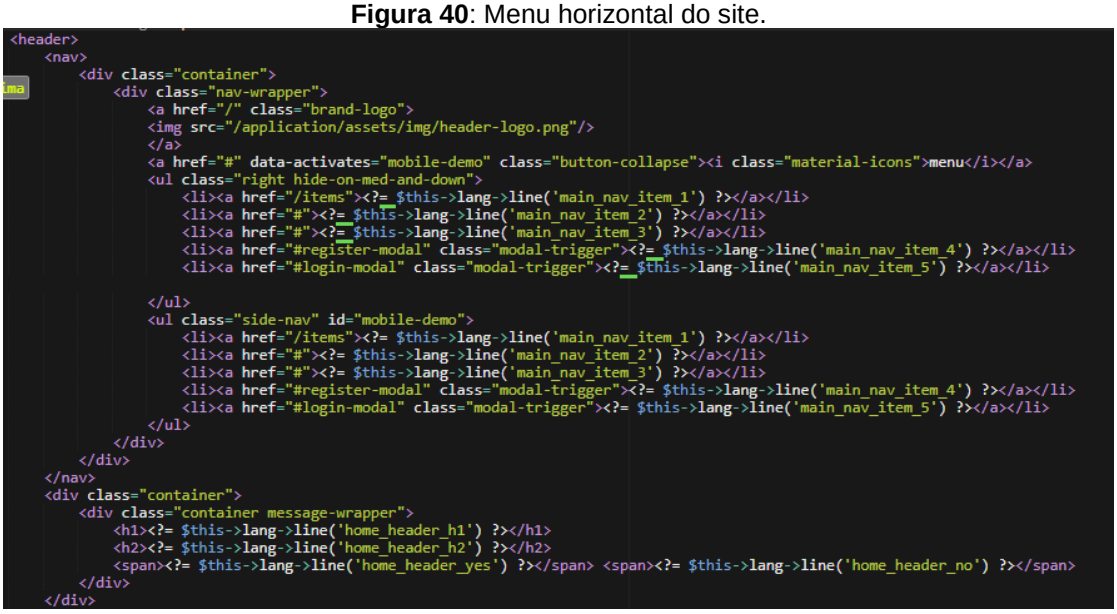
Um ponto interessante é que esse arquivo não possui acesso ao escopo global do *JS*. Portanto, ao receber uma mensagem, por exemplo, não seria possível atualizar a programação do chat instantaneamente. Para resolver isso, utilizou-se o banco de dados *front-end* para que o *service worker* insira a mensagem recebida em uma tabela. Por outro lado, o escopo global verifica a cada 1 segundo se existe uma mensagem nova no banco de dados. Dessa forma, é possível atualizar as mensagens em *background.*

Devido ao uso do *Materialize*, alguns *scripts* também foram usados para atender aos requisitos do *framework*. Fornecem objetos prontos que possibilitam a sobrescrita dos parâmetros das ferramentas, por exemplo, o calendário usado no cadastro: por definição vem em inglês. Sobrescreveu-se então mediante o idioma do usuário.

### 4.5.2 HTML

Foi codificado nos padrões da versão cinco juntamente com os auxílios do *Materialize* para formalizar seções como o menu:

**Figura 40**: Menu horizontal do site.



```
<header>
    <nav>
        <div class="container">
            <div class="nav-wrapper">
                <a href="/" class="brand-logo">
                <img src="/application/assets/img/header-logo.png"/>
                </a>
                <a href="#" data-activates="mobile-demo" class="button-collapse"><i class="material-icons">menu</i></a>
                <ul class="right hide-on-med-and-down">
                    <li><a href="/items"><?= $this->lang->line('main_nav_item_1') ?></a></li>
                    <li><a href="#"><?= $this->lang->line('main_nav_item_2') ?></a></li>
                    <li><a href="#"><?= $this->lang->line('main_nav_item_3') ?></a></li>
                    <li><a href="#register-modal" class="modal-trigger"><?= $this->lang->line('main_nav_item_4') ?></a></li>
                    <li><a href="#login-modal" class="modal-trigger"><?= $this->lang->line('main_nav_item_5') ?></a></li>

                </ul>
                <ul class="side-nav" id="mobile-demo">
                    <li><a href="/items"><?= $this->lang->line('main_nav_item_1') ?></a></li>
                    <li><a href="#"><?= $this->lang->line('main_nav_item_2') ?></a></li>
                    <li><a href="#"><?= $this->lang->line('main_nav_item_3') ?></a></li>
                    <li><a href="#register-modal" class="modal-trigger"><?= $this->lang->line('main_nav_item_4') ?></a></li>
                    <li><a href="#login-modal" class="modal-trigger"><?= $this->lang->line('main_nav_item_5') ?></a></li>
                </ul>
            </div>
        </div>
    </nav>
    <div class="container">
        <div class="container message-wrapper">
            <h1><?= $this->lang->line('home_header_h1') ?></h1>
            <h2><?= $this->lang->line('home_header_h2') ?></h2>
            <span><?= $this->lang->line('home_header_yes') ?></span> <span><?= $this->lang->line('home_header_no') ?></span>
        </div>
    </div>
```

**Fonte**: Autor, 2017.

Pode-se perceber que na imagem acima estão sendo usados códigos *PHP* no conteúdo dos elementos, ao invés de textos. Isto é efeito da internacionalização supracitada: usa-se código e, mediante a linguagem do dispositivo do usuário, obtêm-se o texto traduzido.

Usou-se bastante *modal* – pequena tela sobreposta sobre o site – para tarefas como *login*, cadastro, troca, mensagens, carregamentos e visualização de itens. Dessa forma, ao poupar o carregamento de uma nova página para tal ação, facilitou-se a navegação. A Figura 41 mostra o código de aplicação de modal nos padrões da *framework*.

Para poder realizar o objetivo de tornar o projeto adaptável aos diversos padrões de layout, foi utilizado o esquema de *grids* que obedece a diferentes resoluções através de classes (Figura 42). Onde as classes não foram suficientes, fizeram-se códigos em *CSS*: estão exemplificados no próximo capítulo.

**Figura 41**: Exemplo de modal utilizado para o cadastro.

```
<div id="register-modal" class="modal bottom-sheet">
    <div class="modal-content">
        <h4 class="register-title"><?= $this->lang->line('home_modal_register_title') ?></h4>
        <p class="register-text-body"><?= $this->lang->line('home_modal_register_body') ?></p>
        <div>
            <div class="row">
                <div class="input-field col s12 m4 ">
                    <input id="username" type="text" class="validate" required>
                    <label for="username" data-error="wrong" data-success="right"><?= $this->lang->line('home_modal_register_name') ?></label>
                </div>
                <div class="input-field col s12 m4">
                    <input id="nickname" type="text" class="validate" required>
                    <label for="nickname" data-error="wrong" data-success="right"><?= $this->lang->line('home_modal_register_nickname') ?></label>
                </div>
                <div class="input-field date-field col s12 m4">
                    <input type="text" id="birth" class="datepicker" required>
                    <!--<label for="birth" data-error="wrong" data-success="right">Aniversário</label>-->
                </div>
            </div>
            <div class="row">
                <div class="input-field col s6 m4">
                    <input id="emailuser" type="email" class="validate" required>
                    <label for="emailuser" data-error="wrong" data-success="right"><?= $this->lang->line('home_modal_register_email') ?></label>
                </div>
                <div class="input-field col s6 m4">
                    <input id="password" type="password" class="validate">
                    <label for="password" data-error="wrong" data-success="right"><?= $this->lang->line('home_modal_register_password') ?></label>
                </div>
                <div class="input-field col s6 m4">
                    <input id="country" type="text" class="validate" placeholder="<?= $this->lang->line('home_modal_register_country') ?>" required>
                    <!--<label for="country" data-error="wrong" data-success="right">Pais</label>-->
                </div>
            </div>
```

**Fonte**: Autor, 2017.

**Figura 42**: Exemplo de código adaptável utilizado na página inicial.

```
<div class="col s12 m6 l3">
    <div class="content">
        <div class="wrapper-step">
            <h4>01</h4>
        </div>
        <img src="/application/assets/img/how-it-works-computer.svg"></img>
        <span><?= $this->lang->line('home_how_works_step_1') ?></span>
    </div>
</div>
<div class="col s12 m6 l3">
    <div class="content">
        <div class="wrapper-step">
            <h4>02</h4>
        </div>
        <img src="/application/assets/img/how-it-works-ux.svg"></img>
        <span><?= $this->lang->line('home_how_works_step_2') ?></span>
    </div>
</div>
<div class="col s12 m6 l3">
    <div class="content">
        <div class="wrapper-step">
            <h4>03</h4>
        </div>
        <img src="/application/assets/img/how-it-works-iron.svg"></img>
        <span><?= $this->lang->line('home_how_works_step_3') ?></span>
    </div>
</div>
<div class="col s12 m6 l3">
    <div class="content">
        <div class="wrapper-step">
            <h4>04</h4>
        </div>
        <img src="/application/assets/img/how-it-works-confetti.svg"></img>
        <span><?= $this->lang->line('home_how_works_step_4') ?></span>
    </div>
</div>
```

**Fonte**: Autor, 2017.

A Figura 43 mostra um código que utiliza recursos do *Vue* para dinamizar os elementos ao controlar as ações efetuadas pelo usuário e exibir as informações dos

modelos. O “{{}}” é usado para exibir o valor dos atributos que estão no *javascript* e o prefixo “v” indica funções do framework *javascript*, como na imagem há o *“v-bind”.*

**Figura 43**: Exemplo de código *HTML* com *Vue* utilizado para exibir o item.

```
<div class="modal-infomations">
    <div class="modal-content">
        <h4 class="modal-title">{{item.title}}</h4>
        <div class="sub-informations">
            <span>{{item.category}}</span>
            <span>{{item.useState}}</span>
            <span>{{item.likes}} curtida (s)</span>
        </div>
        <p class="description">
            {{item.description}}
        </p>
        <div class="author-view">
            <a href="#!user">
                <div v-bind:style='{ backgroundImage: "url(" + item.authorImage + ")", }'
                     class="author-image"></div>
                <span class="author-name">{{item.authorName}}</span>
            </a>
        </div>
    </div>
```

**Fonte**: Autor, 2017.

E, por fim, o *modal* das mensagens, que utiliza os dados dinâmicos fornecidos pelo modelo de dados pré-definido no vue:

**Figura 44**: Exemplo de código *HTML* com *Vue* utilizado para exibir o item.

```
<div class="modal-content">
    <div class="row valign-wrapper" id="chatheader">
        <div class="col s1">
            <a href="#!" class="modal-action modal-close"><i class="material-icons white-text small">keyboard_arrow_left</i></a>
        </div>
        <div class="col s2">
            <img class="circle" width="50px" v-bind:src="target_user.picture"/>
        </div>
        <div class="col s9">
            <h4 class="login-title">{{target_user.username}}</h4>
            <a href="#">{{target_user.nickname}}</a>
        </div>
    </div>
    <div class="row" id="chatbox">
        <div class="row message-row" v-for="chatMessage in chatMessages">
            <div class="col m6 offset-m6" v-if="chatMessage.sent_by_sender == '1' " >
                <div class="my-message">
                    <p>{{chatMessage.description}}</p>
                    <span class="my-message-date message-date">{{chatMessage.created}}</span>
                </div>
            </div>
            <div class="col m6" v-else >
                <div class="target-message">
                    <p>{{chatMessage.description}}</p>
                    <span class="message-date">{{chatMessage.created}}</span>
                </div>
            </div>
        </div>
    </div>
    <div class="row valign-wrapper" id="sendmessage">
        <div class="col s11">
            <div id="messagebox" tabindex="1" v-on:keyup.enter="sendMessage()" contenteditable></div>
        </div>
        <div id="send-button" class="col s1" v-on:click="sendMessage()">
            <a class=""><i class="material-icons white-text">send</i></a>
        </div>
    </div>
</div>
```

**Fonte**: Autor, 2017.

### 4.5.3 CSS

O diferencial foi o uso das *media queries* para suprir as faltas necessárias para a concretização do layout totalmente adaptável:

**Figura 45**: Exemplo de CSS utilizado para o layout adaptável.

```css
@media only screen and (max-width: 992px) {
    header,
    main,
    footer {
        margin-left: 0;
    }
    .side-bar-logo h3 {
        right: 25px;
    }
    .mobile-user-view{
        display: block;
    }
    #slide-out{
        position: fixed;
    }
    .user-view .user-infomations {
        position: absolute;
        left: 93px;
        top: 5px;
    }
    .mobile-nav-bar{
        display: block;
    }
    .content-items-wrapper .item-results{
        margin-left: 0px;
    }
}
```

**Fonte**: Autor, 2017.

De resto, a escrita do código foi voltada à padronização do sistema em um único esquema de *design* e sobrescritas em algumas classes do *Materialize*.

### 4.5.4 REST

Ele foi implementado no padrão dito "*RESTful*". Logo, os dados enviados e recebidos pela aplicação estão em formato *JSON*, conforme exemplifica a Figura 46.

**Figura 46**: Exemplo de JSON utilizado para o item.


```json
[
    {
        "user_id": "91",
        "nickname": "Velit.",
        "phone": null,
        "profile_image": "http://lorempixel.com/640/480/?89331",
        "item_id": "91",
        "title": "Miss",
        "description": "Iure labore vitae ut optio suscipit est impedit illum. Ut explicabo fuga a odio tenetur dicta.",
        "use_state": "USADO",
        "category": "ELETRONICO",
        "active": "1",
        "images": [],
        "qt_likes": "0"
    }
]
```


**Fonte**: Autor, 2017.

Os end-point utilizados foram definidos pelas rotas (endereços) da Figura 47 e foram separados dos endereços da página em si através do diretório *API*.

**Figura 47**: Exemplo de rota utilizado no projeto.

```
# image routes operation in controller:
$route['API/image/(:num)'] = 'API/image/index/id/$1';
$route['API/image/register'] = 'API/image/register';
# item routes operation in controller:
$route['API/item/(:num)'] = 'API/item/index/id/$1';
$route['API/item/(:num)'] = 'API/item/index/id/$1';
$route['API/item/trade/(:num)'] = 'API/item/trade/id/$1';
$route['API/item/(:num)/trade/(:num)'] = 'API/item/trade/item_yours/$1/item_theirs/$2';
$route['API/item/(:num)/refuse/trade/(:num)'] = 'API/item/refuse_trade/item_yours/$1/item_theirs/$2';
# user routes operation in controller:
$route['API/user/login'] = 'API/user/authenticate';
$route['API/user/(:num)'] = 'API/user/index/id/$1';
$route['API/user/(:num)/wish'] = 'API/user/wish/id/$1';
$route['API/user/(:num)/wish/item/(:num)'] = 'API/user/wish/user_id/$1/item_id/$2';
$route['API/user/(:num)/like'] = 'API/user/like/id/$1';
$route['API/user/(:num)/like/item/(:num)'] = 'API/user/like/user_id/$1/item_id/$2';
$route['API/user/password-recovery'] = 'API/user/password_recovery';
```

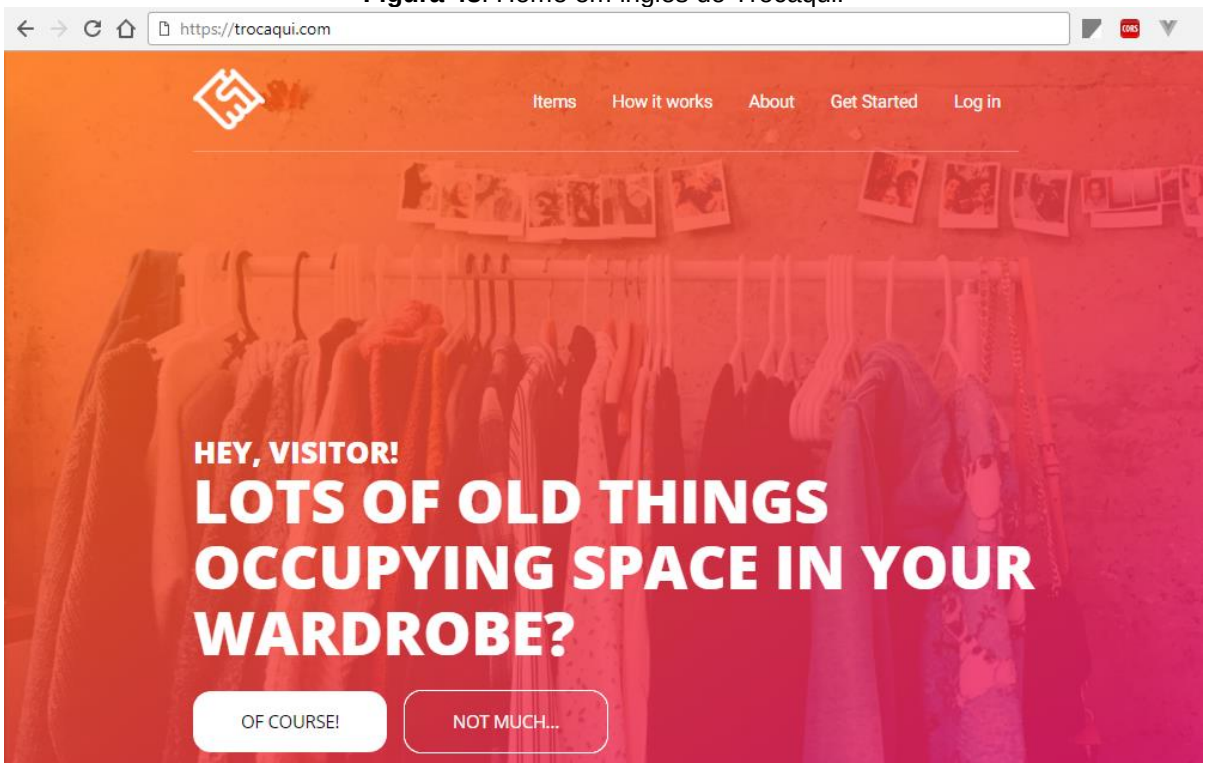
**Fonte**: Autor, 2017.

# 5 RESULTADO

As metodologias empregadas atenderam suficientemente aos objetivos do projeto – atenderão também aos planos futuros – e isso foi atestado no decorrer do desenvolvimento, quando não puderam ser aplicadas diversas funcionalidades de potencial, devido à limitação do escopo e ao prazo.

No estado da arte foram estudadas algumas plataformas existentes no mercado, objetivando a coleta de melhorias e diferenciais possíveis à aplicação no presente estudo. Foi percebido que a maioria dos *sites* existentes trabalha com venda e negociações de moeda. O Trocaqui manteve sua característica de escambo, dando suporte não somente ao Brasil – como a plataforma Trocas Online, a única que lida com troca sem o envolvimento de moedas encontrada nas duas primeiras páginas do *Google*, porém em Portugal – mas outros países em que se falam as línguas em questão, embora nem todos os idiomas estejam completos; caso nenhum seja identificado, o idioma por padrão será inglês, realizando, contudo, a pesquisa de objetos de acordo com a localidade do usuário. Na Figura 48, pode-se ver o diferencial: a página está em automaticamente inglês sem afetar o endereço.

**Figura 48**: Home em inglês do Trocaqui.




**Fonte**: Autor, 2017.

Caso o idioma do navegador esteja em francês, o *site* será carregado neste idioma, e assim por diante.

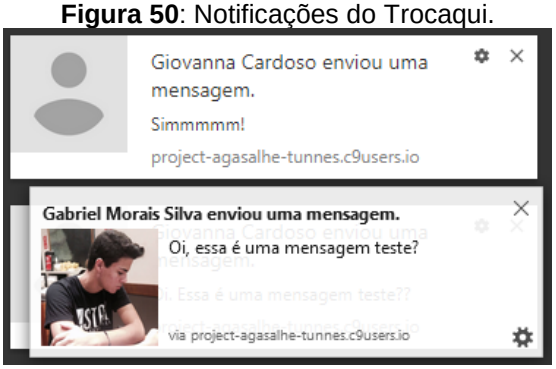
Outro ponto positivo é a adaptação do site em outras resoluções – sem adicionar o prefixo “m” de *site mobile* – que foi alcançada com as técnicas de *front-end* selecionadas:

**Figura 49**: Home responsiva do Trocaqui.


**Fonte**: Autor, 2017.

O domínio do site possui certificado *SSL*, diferente do principal concorrente Trocas Online. Sendo assim, foi possível utilizar a *API* de notificações do *firebase* que exige um endereço seguro. A Figura 50 mostra um exemplo de notificação implementado no *Google Chrome* e *Firefox*. Com ele, o usuário pode recebê-la estando ou não com o site aberto; no caso do *Chrome* ela aparece também com o navegador fechado, *Firefox* e *Opera* o faz somente em *mobile*.

**Figura 50**: Notificações do Trocaqui.

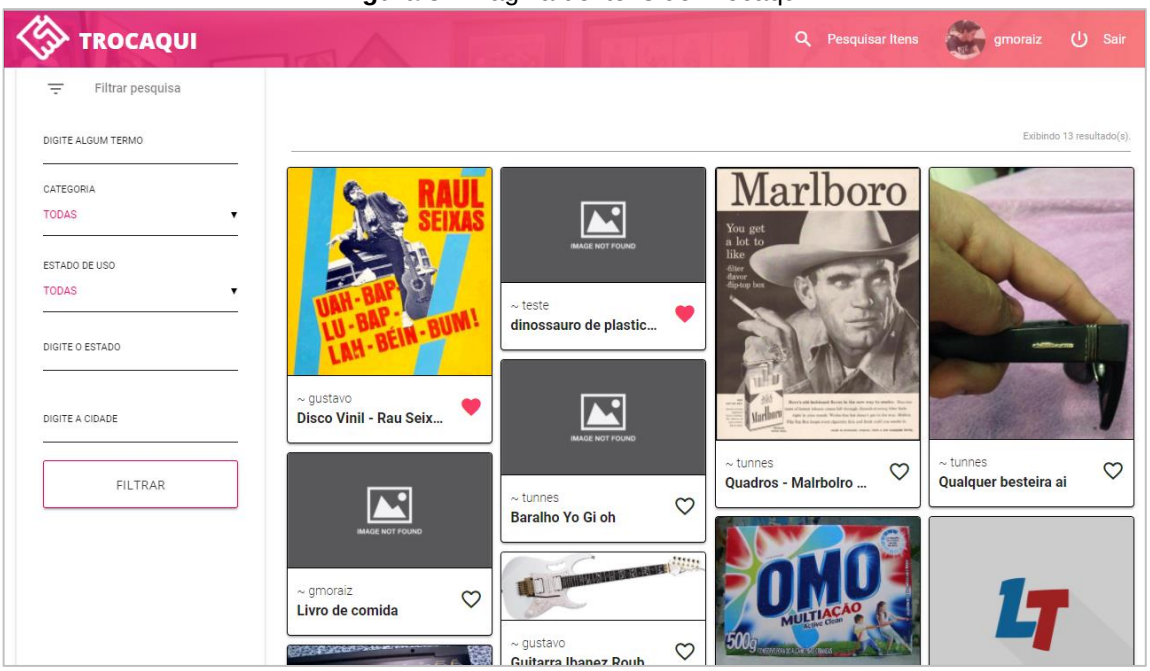



**Fonte**: Autor, 2017.

## 5.1 PERCURSO DE NAVEGAÇÃO

Para realizar uma troca, o usuário deve estar autenticado e selecionar um item na página de itens:

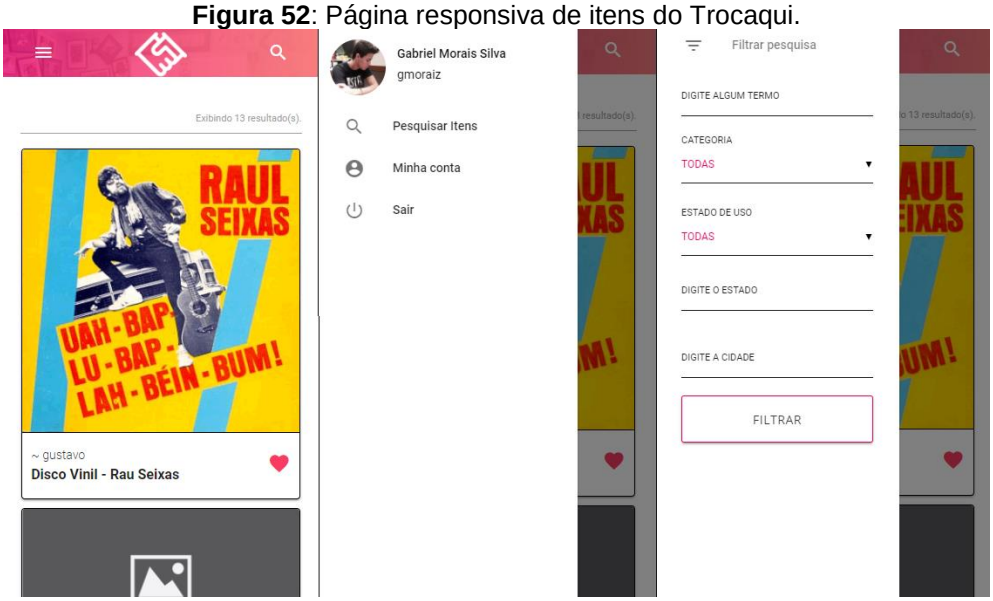
**Figura 51**: Página de itens do Trocaqui.




**Fonte**: Autor, 2017.

E, segue em versão responsiva, a exibição da página de item com três possibilidades:

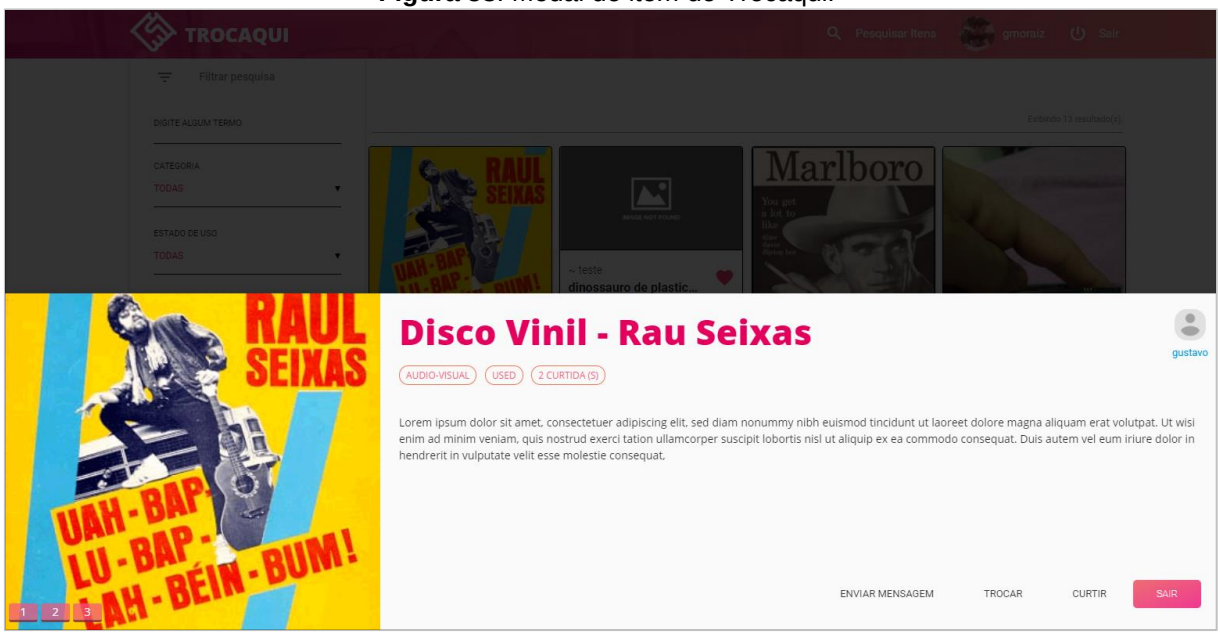
**Figura 52**: Página responsiva de itens do Trocaqui.




**Fonte**: Autor, 2017.

Após selecionado, um modal será aberto expandindo o item (Figura 53). Caso não haja autenticação, a tela será a mesma, porém sem as opções de trocar, curtir e enviar mensagem.

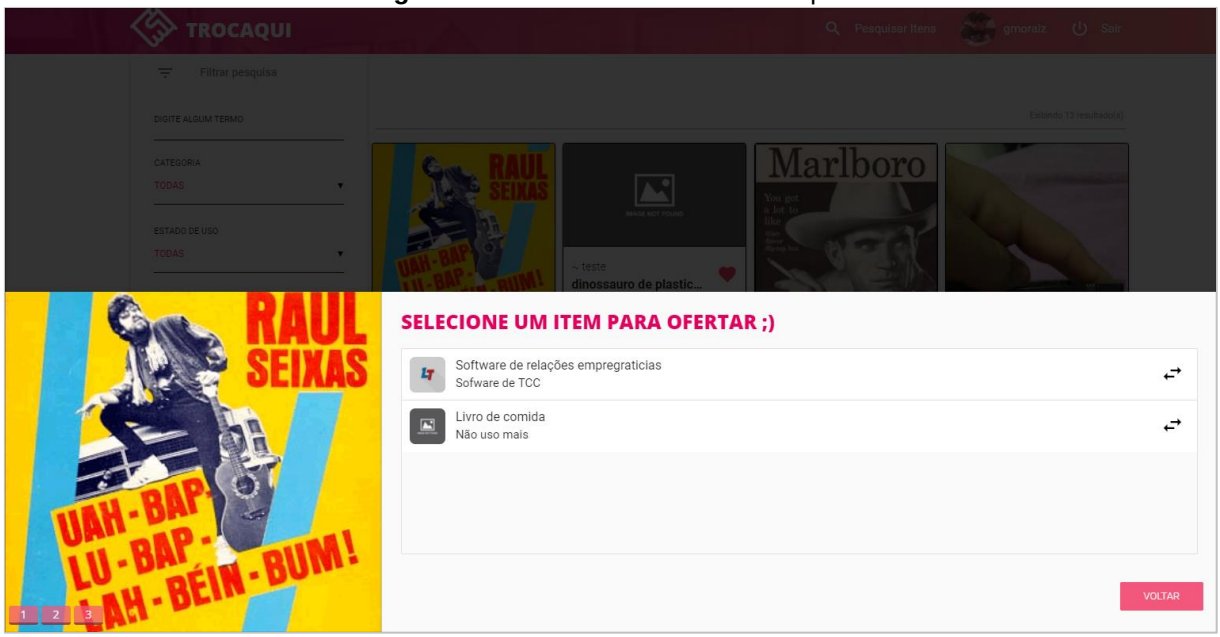
**Figura 53**: Modal do item do Trocaqui.




**Fonte**: Autor, 2017.

E então, ao clicar em trocar, outra tela surgirá (Figura 54), para que ele selecione um item seu em oferta ao desejado.

**Figura 54**: Modal de oferta do Trocaqui.




**Fonte**: Autor, 2017.

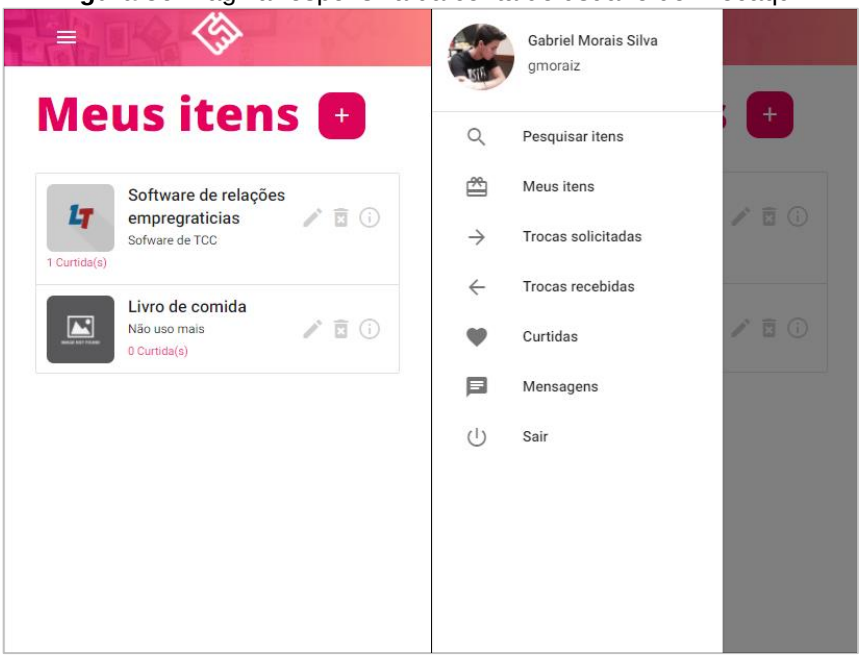
O dono do item será notificado e ambos poderão manter a negociação por outros meios de contato. Em seu painel (Figura 55), ele poderá selecionar a opção de trocas recebidas para responder a solicitação.

**Figura 55**: Página da conta do usuário do Trocaqui.


**Fonte**: Autor, 2017.

Esta página, em sua versão responsiva fica da seguinte forma:

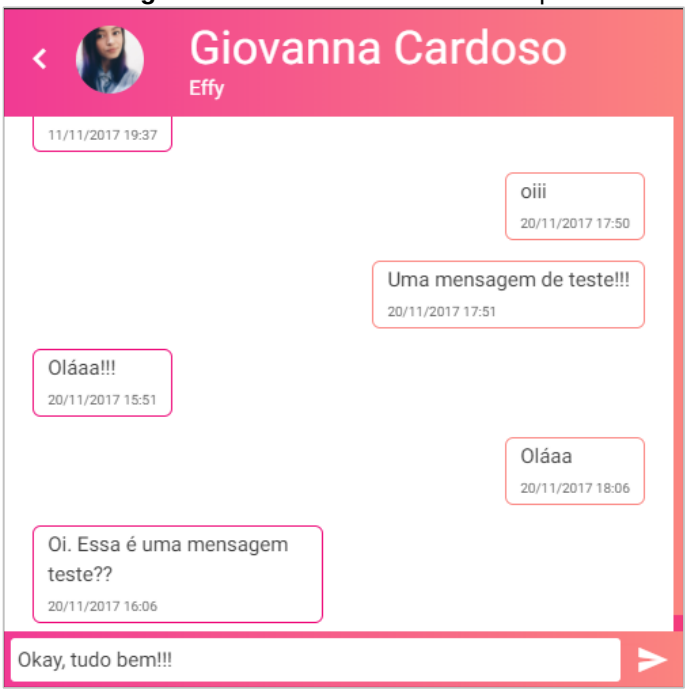
**Figura 56**: Página responsiva da conta do usuário do Trocaqui.




**Fonte**: Autor, 2017.

Então, a continuidade da negociação pode ser realizada através da conversação através do chat implementado (Figura 57). Estando autenticado, o usuário poderá enviar mensagens para qualquer usuário.

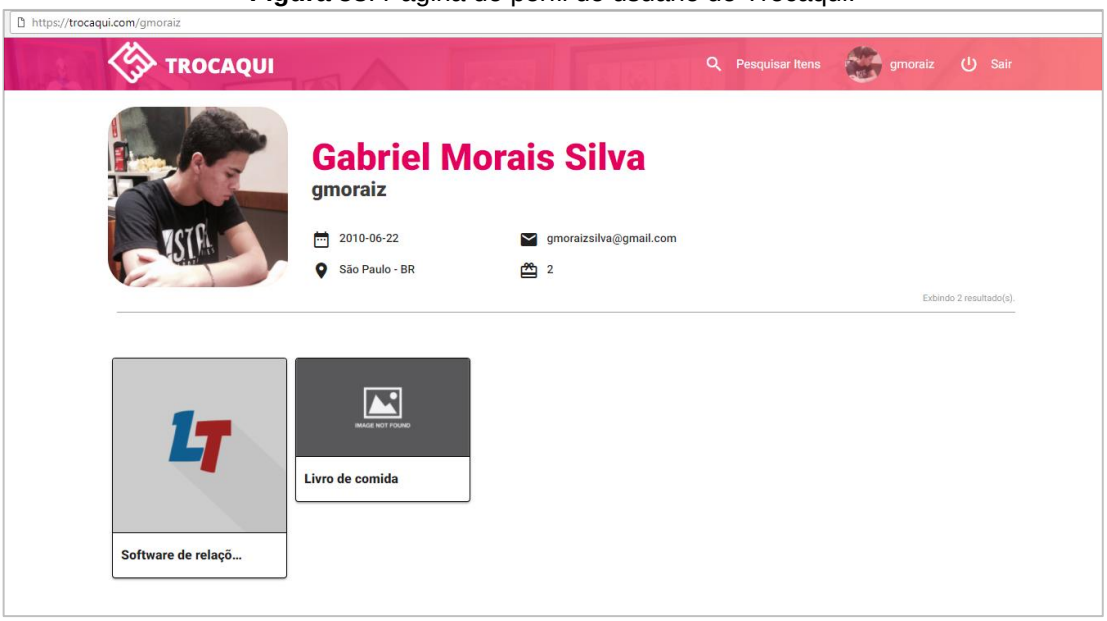
**Figura 57**: Modal do chat do Trocaqui.




**Fonte**: Autor, 2017.

Caso haja clique sobre a imagem do usuário ou seu *nickname*, o respectivo perfil será aberto em uma nova janela na *url* principal "/" nome de usuário (Figura 58).

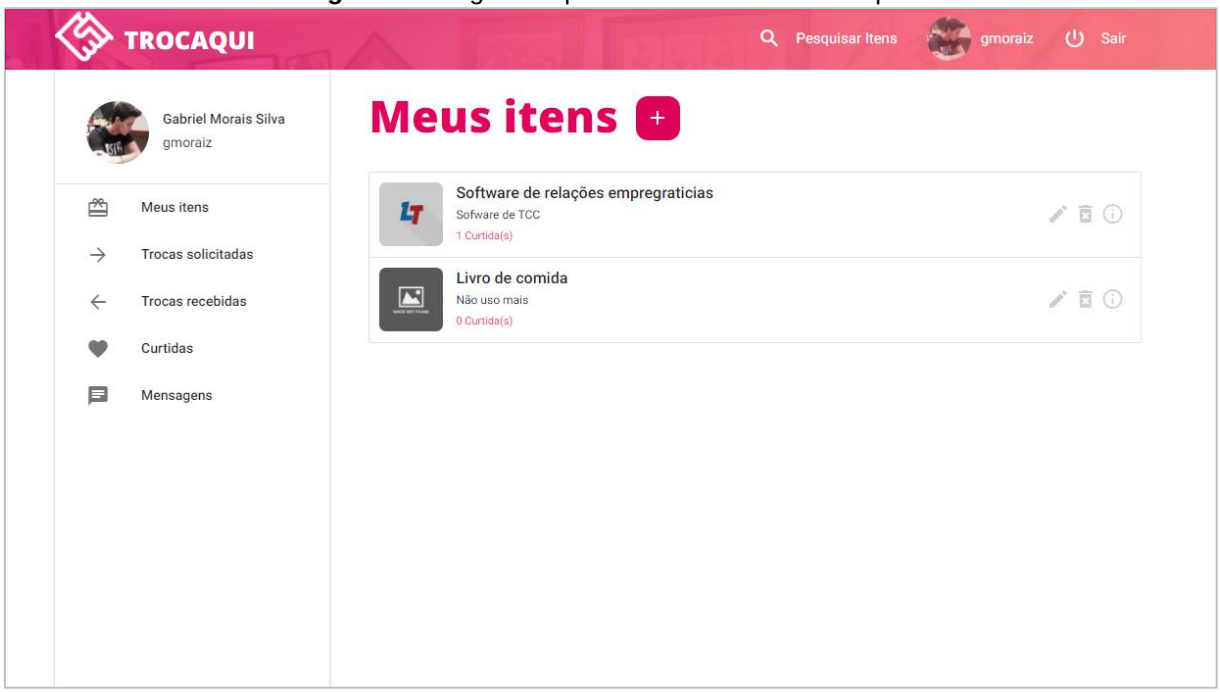
**Figura 58**: Página do perfil do usuário do Trocaqui.




**Fonte**: Autor, 2017.

Assim que autenticado, o usuário será redirecionado para o perfil de sua conta (Figuras 55 e 56) e a tela de seus itens será a primeira a ser carregada:

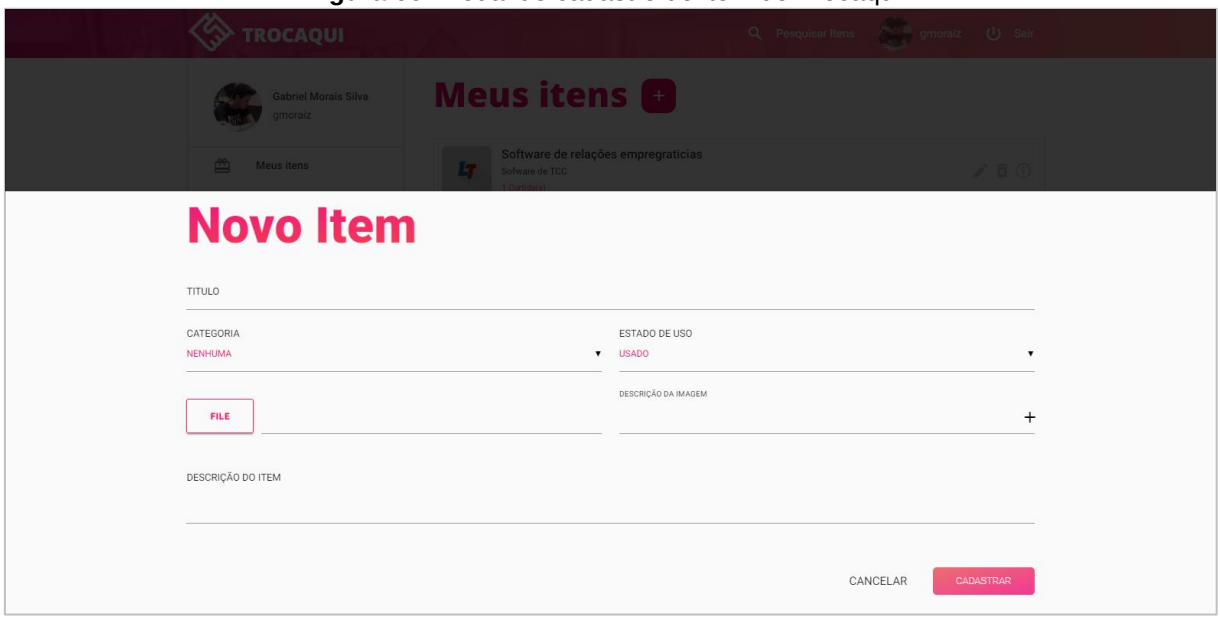
**Figura 59**: Página do perfil do usuário do Trocaqui.




**Fonte**: Autor, 2017.

Para inserir um novo item, basta clicar sobre o ícone com um “+”. Um *modal* será aberto para que os dados sejam preenchidos:

**Figura 60**: Modal do cadastro de item do Trocaqui.




**Fonte**: Autor, 2017.

## 5.2 QUADRO DE COMPARAÇÃO

E por fim, um quadro comparativo, que visa demonstrar o destaque do presente trabalho perante os demais.

**Tabela 11**: Comparação entre o Trocaqui e concorrentes.

| Plataforma | SSL | Responsivo | Internacionalização | Somente Troca |
|---|---|---|---|---|
| OLX | Sim | Não | Não | Não |
| Enjoei | Sim | Sim | Não | Não |
| Swapping | Não | Não | Não | Não |
| Pra que dinheiro? | Sim | Sim | Não | Não |
| Troca Troca Brasil | Não | Não | Não | Não |
| Trocas Online | Não | Sim | Não | Sim |
| **Trocaqui** | **Sim** | **Sim** | **Sim** | **Sim** |

**Fonte**: Autor, 2017.

Os resultados são satisfatórios, porém, não definitivos; mudanças são necessárias futuramente, principalmente na disposição dos elementos, acessibilidade e internacionalização. Trocaqui possui a vantagem de ser o único que trata somente de troca, possui internacionalização, que, diferente da localização (sites direcionados especificamente para regiões, onde os endereços são afetados), visa a globalização dos recursos da plataforma mediante a programação, é responsivo (semelhante ao Trocas Online, que, apesar de atuar somente em Portugal, é o seu único rival direto) e possui certificado *SSL* (estando uns passos à frente deste último).

# 6 CONCLUSÃO

Desde as primeiras cogitações pertinentes a este estudo, problemas e dificuldades foram lançados, ideias negligenciadas e, no presente tema, hipóteses refletidas em busca de um bem comum, estando condizente com as diretrizes acadêmicas, a realidade de cada integrante da equipe e o aparato de cunho técnico-computacional.

Trocaqui é fruto disso. Os objetivos apresentados na introdução foram engajamento para o trabalho dos subsequentes capítulos. O problema a ser resolvido foi peculiar a uma realidade proveniente do atual momento em que nos encontramos. Portanto, o foco a todo momento foi desenvolver um sistema abrangente que tivesse todos os internautas como usuários, com o auxílio da internacionalização. Esta ainda precisa ser muito explorada, pois, apesar do suporte a 5 idiomas, apenas três operam em todo sistema: Inglês, Português e Espanhol. Mas o importante aqui é que a estrutura foi montada com legibilidade e estabilidade, facilitando futuras manutenções de idioma e localidade (como o fuso-horário).

Tendo em mente a grande gama de usuários, acessibilidade também é importante, e este tópico não foi suficientemente implementado, porém, é necessário e faz parte da continuação do sistema, juntamente à aplicação de um sistema de *long-polling* que atenda o esquema de notificações e mensagens para os usuários que não aceitarem as notificações *desktop* ou utilizem navegadores que não a suportem.

Alguns típicos recursos de mídia social, também cogitados inicialmente, foram implementados com aptidão, pois conquistam a amizade de pessoas já familiarizadas com outros canais; além de proporcionar um certo engajamento entre eles. O recurso de comentários, não optado, fica como proposta ulterior. Contudo, um *chat* próprio, que foi implementado aos moldes das “mensagens diretas” do *twitter*, sacia as necessidades de comunicação entre os usuários, afinal, oferecer-lhes uma plataforma inteiramente voltada a trocas e dispensar um meio de conversação e negociação – sem precisar se locomover para outro veículo – seria uma certeza de insucesso.

A atenção à adaptação do site em diversos dispositivos é relativa à abrangência de usuários. Muitos podem ser os dispositivos, resoluções e navegadores. Houve sucesso na implementação do *design* responsivo, as pessoas poderão usar o sistema em seus sistemas móveis. Entretanto, isso não dirime o intuito de futuramente produzir aplicativos *mobile* para *iOS* e *Android*, com o auxílio do *framework Ionic*.

Em suma, a premissa do presente trabalho – junto aos objetivos específicos – foi alcançada e encontra-se aberta a atualizações. Um trabalho penoso, corrido e virtuoso por conceder à equipe experiência, conhecimento e a porta de entrada ao registro acadêmico. Aos internautas, cabe a plataforma, pronta a atender posteriores intuitos de troca de uma maneira nova, exclusiva e dinâmica.

# REFERÊNCIAS


ALMEIDA, A. P. S. **Design Responsivo**: Técnicas, frameworks e ferramentas. 2014, p. 12. Disponível em: <http://bsi.uniriotec.br/tcc/textos/201412Almeida.pdf>. Acesso em 22 de set. 2017

BALDUÍNO, P. **Dominando JavaScript com jQuery**. 2014, seção 5.

BELL, P; BEER, B. **Introdução ao GitHub**: Um guia que não é técnico. 2015, p. 13.

BOTSMAN, R; ROGERS, R. **O Que É Meu É Seu**: Como o Consumo Colaborativo Vai Mudar o Nosso Mundo.

CASTELLS, M. **A galáxia internet**: reflexões sobre a internet, negócios e a sociedade. 2001, p. 109.

CHACON, S; STRAUB, B. **Pro Git**, 2º ed, 2014. Disponível em: <https://git-scm.com/book/pt-br/v2>. Acesso em 18 de set. de 2017.

CASTRO, S. B. de.; MANARA, E. M. **Desenvolvimento de Software II**: Introdução ao Desenvolvimento Web com HTML, CSS, JAVASCRIPT e PHP. 2014, p. 61, 95

**Enjoei**. Disponível em: <https://www.enjoei.com.br>. Acesso em 29 de ago. 2017

**_____**. Disponível em: <https://www.yafue.com.ar>. Acesso em 29 de ago. 2017

FERNANDO, L; EDUARDO, V. **Consumo excessivo e adicção na pós-modernidade**: uma interpretação psicanalítica. 2004, p. 426.

FISCHER, G. D. **As trajetórias e características do youtube e globo media center/globo vídeos**: um olhar comunicacional sobre as lógicas operativas de websites de vídeos para compreender a constituição do caráter midiático da web. 2008, p. 43-44.

GAMMA, ERICH. **Padrões de Projetos**: Soluções reutilizáveis de software orientado a objetos. 2000, p.20.

IANNI, O. **Teorias da globalização**. 4. ed. Rio de Janeiro: Civilização Brasileira, 1997.

INCAU, C. **Vue.js**: Construa aplicações incríveis. 2017

LOPES, S. **A Web Mobile**: Programe para um mundo de muitos dispositivos. 1º edição, 2013, p. 29.

MARTINS, A. **Autoridade Certificadora para Acesso Seguro**. 02 de março de 2001, p .1.

**Materialize**. Disponível em: <http://materializecss.com>. Acesso em 22 de set. 2017

MILANI, A. **MySQL**: Guia do programador. Novatec, 2006, p. 22

ANTUNES, JONATHAN LAMIM. **CodeIgniter**: Produtividade na criação de aplicações web em PHP. 2016, Seção 6, 11, 41.

NIEDERAUER, J. **Desenvolvendo Websites com PHP**. 2º edição, novatec, 2011, pag. 23

**OLX**. Disponível em: <http://www.olx.com.br>. Acesso em 29 de ago. 2017

**Pra Que Dinheiro**. Disponível em: <https://www.praquedinheiro.com.br>. Acesso em 26 de ago. 2017

SILVA, M. S. **Ajax com JQuery**: Requisições AJAX com a simplicidade de jQuery. 2009, p. 64, 65.

**_____**. **Introdução à JQuery**. Disponível em: <http://www.linhadecodigo.com.br/artigo/2068/introducao-a-jquery.aspx>. Acesso em 22 de set. 2017.

**Swapping**. Disponível em: <http://swapping.com.br>. Acesso em 26 de ago. 2017

**Troca Troca Brasil**. Disponível em: <http://www.trocatrocabrasil.com.br>. Acesso em 26 de ago. 2017

**Trocas Online**. Disponível em: <http://www.trocasonline.com>. Acesso em 02 de set. 2017

XAVIER, Otávio Calaça. **Semantic Web Services based on RESTful**: A Case Study in Online Social Networks. 2011. 137 f. Dissertação (Mestrado em Ciências Exatas e da Terra - Ciências da Computação) - Universidade Federal de Goiás, Goiânia, 2011. Disponível em: <http://repositorio.bc.ufg.br/tede/handle/tde/515#preview-link0> Acesso em 19 de set. de 2017.

ZEMEL, T. **Web Design Responsivo**: Páginas adaptáveis para todos os dispositivos. 25 de março 2015.