TRADE: DESENVOLVIMENTO DE UMA APLICAÇÃO MOBILE PARA TROCA DE PRODUTOS USADOS

Rafael Lima dos REIS[1]
Vinicius Marques PERES[2]
Daniel Facciolo PIRES[3]

Resumo: Através da nova era digital várias atividades realizadas pelas pessoas ficaram mais fáceis, uma delas é a compra de produtos via internet, desde eletrônicos até móveis. Porém, com essa facilidade e o consumo exagerado, também surgiu o acúmulo de produtos que não possuem mais utilidade para aqueles que os adquiriu ou ficaram desatualizados, onde pôr fim são descartados ou ficam armazenados até estragarem. O presente artigo teve objetivo, ajudar as pessoas a se livrarem de produtos que não serão mais utilizados, através de um aplicativo para dispositivos móveis, onde o usuário poderá buscar por outros produtos que lhe interessem e trocar por outro que não utilize mais. Esse artigo metodologicamente apresenta também elementos e técnicas de desenvolvimento de aplicativos híbridos e todos os processos de documentação e desenvolvimento de um aplicativo móvel.

Palavras-chave: aplicativo, *mobile*, troca, produtos, usados, escambo.

Abstract: *Through the new digital age, many activities carried out by people have become easier, one of them is the purchase of products via the internet, from electronics to furniture. But with this ease and over-consumption, there has also arisen the accumulation of products that are no longer useful to those who have acquired them or have become outdated, where they end up being discarded or stored until they spoil. This article aims to help people get rid of products that will no longer be used through a mobile app where the user can search for other products that interest them and switch to another that they no longer use. This article also methodologically presents elements and techniques of hybrid application development and all the documentation and development processes of a mobile application.*

Keywords: *application, mobile, exchange, products, used, barter.*

1 Introdução

Uma das bases do capitalismo é o acúmulo interminável de bens onde a publicidade é usada como ferramenta para criar desejos nas pessoas de sempre adquirir novos itens, gerando assim um acúmulo desnecessário para a pessoa ou para qualquer integrante de sua família. Com isso, há um grande desperdício de espaço ocupado por esses itens. Sem saber o que fazer com estes, as pessoas tendem a deixá-los guardados até que estraguem ou se deteriorem, e consequentemente são jogados no lixo.

[1] Discente do curso de Bacharelado em Sistemas de Informação - Centro Universitário Municipal de Franca – rafaellimareis01@gmail.com
[2] Discente do curso de Bacharelado em Sistemas de Informação - Centro Universitário Municipal de Franca – viniciusmarquesperes@gmail.com
[3] Docente do curso de Bacharelado em Sistemas de Informação - Centro Universitário Municipal de Franca – daniel@facef.br

Outro fator que alavanca esse acúmulo de bens materiais é a crescente onda de novas tecnologias, com constantes mudanças no mercado, tornando assim, os produtos obsoletos com uma grande velocidade. Neste ponto entra a economia compartilhada, que ganhou força nos tempos de crise. Essa prática permitiu aos consumidores terem acesso a produtos de qualidade com um valor menor ou em troca por outros bens.

A economia compartilhada foi iniciada em meados de 90 nos Estados Unidos com o incentivo dos avanços tecnológicos da época, gerando assim uma redução dos custos das transações de "pessoa para pessoa". MELLO (2016, p.3) define a economia compartilhada como um sistema que possui sua base formada em cima do compartilhamento de recursos humanos, comércio e consumo compartilhado de bens e serviços. Essa ideia sugere negociações de compartilhamento, empréstimo, aluguel, doação e troca de bens.

Mesmo com a existência da economia compartilhada, hoje em dia há poucos métodos para realizar essas trocas. Existem algumas plataformas web destinadas a esse propósito, como "TomaLaDaCa[4]", "Troca Fácil[5]", mas que não chegaram a agradar os usuários, pois não apresentam um *layout* atrativo nem um ambiente organizado, devido a esses fatos os usuários acabavam se perdendo em busca de produtos ou que dificultaram alguns processos simples como: cadastro de usuário ou produto. O mais utilizado atualmente para tentar alavancar a economia compartilhada são as trocas realizadas através de grupos em redes sociais.

É pouco improvável negar que passamos por uma evolução nos últimos anos para a "era digital" onde todas as mudanças na sociedade e nas organizações são transmitidas para todos em pouco tempo. Com essa evolução a internet vem tomando a maior parte do tempo das pessoas, junto com o celular, que se torna uma poderosa ferramenta para facilitar o dia a dia de todos, nas tarefas de compras, pagamentos e troca de bens. Por isso a maior aposta do mercado são os aplicativos *mobile*.

Com a economia compartilhada, há algumas vantagens em aderir a ideia como: eliminar coisas desnecessárias, espaços extras na casa, novos bens sem custos monetários. Quem já aplica esse sistema utiliza normalmente grupos em suas redes sociais para realizar a troca de seus bens entre si.

[4] https://www.1001trocas.com.br/v01 acessado em 13/03/2018
[5] https://www.trocafacil.com.br acessado em 13/03/2018

Foram analisados alguns sistemas encontrados hoje no mercado, em base dessa análise foram levantados alguns itens que poderiam ter sido abordados pelas aplicações. Algumas questões são: nenhuma das aplicações possuem aplicativos para dispositivos móveis, onde segundo pesquisa realizada pelo IBGE em 2015 (IBGE, p.43) o acesso à internet era feito em 92% dos casos através do celular, ou seja, é bem mais fácil o usuário realizar suas trocas através de um aplicativo móvel do que em seu computador, também os *layouts* complexos que deixavam os usuários perdidos é um fator importante para o contato inicial com o aplicativo.

Neste contexto, o problema de pesquisa desse trabalho é responder se as tecnologias de desenvolvimento de aplicações para dispositivos móveis, bem como as metodologias e técnicas da engenharia de *software* disponíveis atualmente permitem o desenvolvimento de uma solução móvel para troca de usados.

O objetivo deste trabalho é o desenvolvimento de um *software* para dispositivos móveis, independente de plataforma, para apoiar a troca de usados utilizando tecnologias e metodologias da engenharia de *software*.

Os objetivos específicos são apresentados abaixo:

1. Criar documentação necessária do *software*;
2. Produzir código-fonte de acordo com padrões de projeto de *software*;
3. Testar a solução com potenciais usuários;
4. Apresentar o projeto na forma de um Canvas.

Os procedimentos metodológicos estão assim organizados:

1. Revisão bibliográfica dos temas aderentes ao tema do trabalho;
2. Levantamento de requisitos e funcionalidades com potenciais usuários;
3. Construção do diagrama BPMN;
4. Criação de diagramas da UML;
5. Projeto e desenvolvimento do *software* utilizando Angular 4 e Ionic 3 e o paradigmas baseado em componentes.

Este trabalho justifica-se pelos resultados de uma pesquisa quantitativa realizada no início do trabalho para verificar se o uso de um *software* de troca de usados seria interessante. Com um total de 78 participantes, 98% responderam que utilizariam um

aplicativo deste tipo para realizar suas trocas, 51% dos participantes passam mais tempo na internet usando seus dispositivos móveis e que 59,5% recomendariam a aplicação a outras pessoas (REIS e PERES, 2018). Com isso, chegou-se a um resultado positivo sobre o desenvolvimento do aplicativo para dispositivos móveis, com um bom nível de aceitação e a possibilidade de se criar uma comunidade focada na economia compartilhada.

2 Referencial Teórico

A seguir serão discutidos alguns temas que foram abordados no decorrer desse projeto, como: gerência de projetos, tecnologias para desenvolvimento de aplicações híbridas, *e-commerce* e startup, a fim de apresentar alguns aspectos que englobam o desenvolvimento de uma aplicação.

2.1 E-commerce

O *e-commerce*, que em português significa comércio eletrônico, é uma categoria de comércio que realiza transações por meio de dispositivos e plataformas eletrônicas, como computadores e *smartphones*. Um exemplo comum dessa categoria é comprar ou vender produtos em lojas virtuais.

No início, o principal foco do *e-commerce* era para vender bens com valores modestos, como: livros e CDs. Atualmente, ele é utilizado para fins diversos, desde produtos como: roupas e alimentos, até produtos que possuem um valor de aquisição alto.

FELIPINI (2012) aponta a mudança na forma como as pessoas realizam compras e realizam transações. O autor aponta que essa mudança é uma excelente oportunidade para quem deseja ingressar no mundo do comércio eletrônico, pois é bem mais fácil se estabelecer em um setor que está em crescimento do que outro que está estabilizado. Ainda de acordo com (FELIPILI, 2012), há algumas razões para criar um empreendimento na internet, as quais seriam:

- **A força da novidade iguala as oportunidades:** o conhecimento relativo a esse ambiente de negócios, o chamado *know-how*, ainda não está plenamente definido. Na verdade, empresários e estudiosos do *e-commerce* ainda estão buscando conhecimento e obtendo aprendizado com erros e acertos durante seus caminhos. Este fato tem como função de nivelar, onde a distância de quem já está neste ambiente e quem está iniciando, é muito curta, sendo assim pequenos segmentos que, por ventura, possa a vir não interessar às grandes empresas ou não estejam suficientes maduros, podem representar uma excelente oportunidade de negócios para empreendedores com uma visão mais ampla;

- **O empreendimento pode ser implantado aos poucos e testado:** ao contrário de uma empresa tradicional, em que o início das operações ocorre somente com o empreendimento totalmente estruturado, um negócio no comércio eletrônico pode ser implementado em etapas, onde o investimento é diluído caucionado a facilidade nas correções de erros.
- **O momento é agora:** com a crescente de dispositivos tecnológicos e recursos fornecidos pelos mesmos, estima-se que as transações eletrônicas só tendem a crescer, ou seja, cada vez mais estará concentrado informações e oportunidades no mundo eletrônico (FELIPILI, 2012).

2.2 Aplicações Híbridas

Aplicações híbridas são aplicações que não são desenvolvidas para um dispositivo específico. Estas aplicações possuem características e funcionalidades que dependem de um conjunto de *softwares* para que sua implementação ocorra, como a câmera ou agenda, por exemplo. Outra característica das aplicações híbridas está no processo de desenvolvimento do projeto, que utiliza poucas linguagens e *frameworks*, portanto seu desenvolvimento ocorre de forma ágil e consequentemente possui um custo baixo (AGUIAR, 2017).

Resumidamente os aplicativos híbridos são aplicativos desenvolvidos de forma responsiva, ou seja, para funcionar para qualquer dispositivo.

2.3 Aplicações Nativas

Aplicações nativas são desenvolvidas para plataformas específicas, onde é possível explorar todas as funcionalidade do aparelho em questão. O *hardware* do aparelho também é utilizado sem nenhum problema, podendo assim explorar recursos mais avançados de interação com o usuário e um poder de processamento muito grande (MISZURA e DIVINO, 2012).

Com a obrigação de se desenvolver um aplicativo para cada plataforma surge algumas necessidades no decorrer do projeto, como o uso de ferramentas e linguagens específicas, através de *Software Development Kit* (SDKs) e *frameworks*, onde os projetos são vinculados diretamente nos ambientes e só podem ser executados nos dispositivos permitidos. Também é obrigatório que cada aplicativo siga as regras e padrões técnicos de interface e "*User Experience*" de cada plataforma, gerando assim um padrão entre todas as aplicações (REZENER e BRITTO, 2016)

Apesar de todas as vantagens de uma aplicação nativa, também há algumas desvantagens. Onde, para que seja desenvolvido um aplicativo para cada Sistema Operacional (SO) é necessário um tempo maior de desenvolvimento e também um orçamento mais alto para pagar os desenvolvedores. Outra questão a ser tratada é que para que o aplicativo funcione em mais de uma versão de S.O. é necessário o desenvolvimento e a adaptação para os componentes disponíveis em cada aparelho e versão (KAZAP, 2015).

2.4 Angular

Angular é um *framework* JavaScript (JS) baseado em TypeScript (TS), foi criado e é mantido pela Google, lançado em 2016, atualmente está na versão 4. Baseado em componentes que de acordo com a documentação do ANGULAR (2018), um componente é uma diretiva com configurações mais simples, que por sua vez são marcadores do Modelo de Objeto de Documento (DOM), tendo atributo, nome ou *Cascading Style Sheets* (CSS), que com o auxílio do compilador do Angular é inserido comportamentos específicos, ou até fazendo transformações no próprio DOM.

WAHLIN (2017) aponta alguns benefícios de se utilizar Angular e TypeScript:

- **Consistência:** Um dos fatos mais importantes no Angular é que a estrutura geral é baseada em componentes e serviços. Os componentes por sua vez começam da mesma maneira (com seus "*imports*", "*decorator*" e "*class*"), mesmo tendo coisas adicionais a estrutura geral sempre é a mesma. Outra área de consistência são os serviços, mesmo existindo várias opções (*factories*, *services* e *providers*) com Angular, há a possibilidade de "injetar" em seu construtor sem haver diferença e dificuldade entre cada um;
- **Produtividade:** Com a consistência também temos a produtividade, com as classes dentro dos módulos do ECMAScript 6 (ES6), ele se organiza e transforma em um código auto responsável, onde, os dados de entrada passam somente pelas propriedades de entrada e as saídas pelas propriedades de saída;
- **Modularidade:** No Angular existem os módulos que fornecem uma maneira de organizar todas as funcionalidades da aplicação, dividindo assim tudo em recursos que podem ser reutilizados. Há também outros benefícios, como por exemplo o "*lazy loading*", onde um ou mais recursos são carregados em segundo plano ou de acordo com a demanda da aplicação;
- **Captura de Erros Precoce:** Como o Angular utiliza o TypeScript, ele acaba ganhando algumas vantagens, como possibilidade de conexão com arquivos de extensão do formato “.tst” ou diretamente com “.js”, sem ter qualquer tipo de perda de

desempenho. Outra vantagem do TS é o suporte dos principais recursos do ECMAScript 2015, ECMAScript 2016 e ECMAScript 2017. Também há o suporte para tipos (primitivos, interfaces e outros tipos personalizados).

Além dessas vantagens do TypeScript, o Angular também foi desenvolvido com incentivos aos testes. Tornando assim todo o ambiente com suporte aos testes de unidade de ponta a ponta com Karma e Jasmine.

2.5 Ionic

De acordo com a Documentação do IONIC (2018), esse é um *Software Development Kits* (SDK) que permite desenvolver aplicativos para dispositivos móveis com um desempenho quase similar aos aplicativos "nativos" (Java para Android e Objective-c ou Swift para iOS), usando apenas linguagens web (*Hypertext Markup Language* ou em português Linguagem de Marcação de Hipertexto (HTML), Cascading Style Sheets ou em português Folha de Estilo em Cascatas (CSS) e seus derivados e Angular).

Em seu artigo, CÉSAR (2017) aponta alguns pontos positivos do Ionic:

- Estrutura do projeto, como cada componente é separado por pastas deixando assim mais fácil de encontrar os itens desejados;
- Organização das dependências através do @ngModule, onde, ao invés de declarar em vários lugares da aplicação só é necessário declarar em um só lugar. Deixando assim o código muito mais fácil de dar manutenção;
- Ionic CLI, um gerador de componentes, com isso pode ser desenvolvido (*pages*, *providers*, *tabs*, *pipes*, *components* e diretivas). Além disso todos esses componentes serão organizados de acordo com a estrutura padrão dos projetos Ionic.

O suporte ao ES6 é outra vantagem do Ionic, mesmo que nem os *browsers* não suporta a sintaxe, no momento em que a aplicação é compilada também é gerada uma versão para o ECMAScript 5 (ES5).

2.6 Cordova

APACHE (2015) é uma estrutura de desenvolvimento móvel. Onde é permitido usar tecnologias da web (HTML, CSS e JavaScript) para desenvolvimento, onde é utilizado as *Application Programming Interface* ou em português Interface de Programação de Aplicativos (APIs) de ligação compatíveis com cada dispositivo para acessar os recursos de *hardware* como sensor, *Global Positioning System* ou em português Sistema de Posicionamento Global (GPS), agenda, entre outros.

Em seu artigo JHEYSON (2017) com a ajuda do Cordova, afirma que os desenvolvedores híbridos têm acesso a quase todos os recursos nativos dos aparelhos, como câmera, sensor, lista de contatos, dados dos celulares através das APIs que o Apache Cordova disponibiliza. Quando se utiliza alguma linguagem para aplicações híbridas, é fundamental a utilização de plugins para a comunicação com o *hardware* dos dispositivos. Com isso o desenvolvedor pode instalar só o que é necessário para sua aplicação. Assim, pode-se considerar que o Ionic funciona em cima do Cordova, enquanto o Ionic fica responsável pela interface gráfica e regras de negócio da aplicação, o Cordova fica responsável pela comunicação com o *hardware* e empacotamento do sistema para vários dispositivos móveis.

3 Empreendendo a Solução Proposta

Esta seção apresenta um breve conceito sobre empreendedorismo, *startup* e o modelo Canvas.

3.1 Empreendedorismo

Empreendedorismo é a forma de se fazer algo novo ou diferente de todos os que existem no mercado, é o termo referente a busca de outras oportunidades através da criatividade e inovação. Sendo que uma das formas mais utilizadas atualmente, relacionado com um empreendedor, é o início de uma nova empresa ou um novo produto (QUINTANILHA, 2015).

Em seu artigo, PADILHA (2011 apud HOSKISSON 2008) diz que a essência do empreendedorismo é saber encontrar e ter a coragem de explorar as oportunidades, principalmente as que ninguém vê ou às quais ninguém reconhece uma possível chance de se tornar um grande negócio. Podendo assim ligar o empreendedorismo com a satisfação das necessidade e força para enfrentar crises, explorando oportunidades com criatividade.

3.2 Startup

Uma *startup* é uma empresa em fase inicial, geralmente no início da implementação e organização dos processos internos. Pode até ser uma empresa que ainda não iniciaram a venda de seu produto ou serviço, mas que já esteja funcionando, ou pelo menos em fase de instalação (ALEXANDRE, 2016).

LUNA (2012 apud HERMANSON 2011) apresenta um ponto de vista semelhante onde diz que *startups* são empresas pequenas, em fase inicial, com sua principal

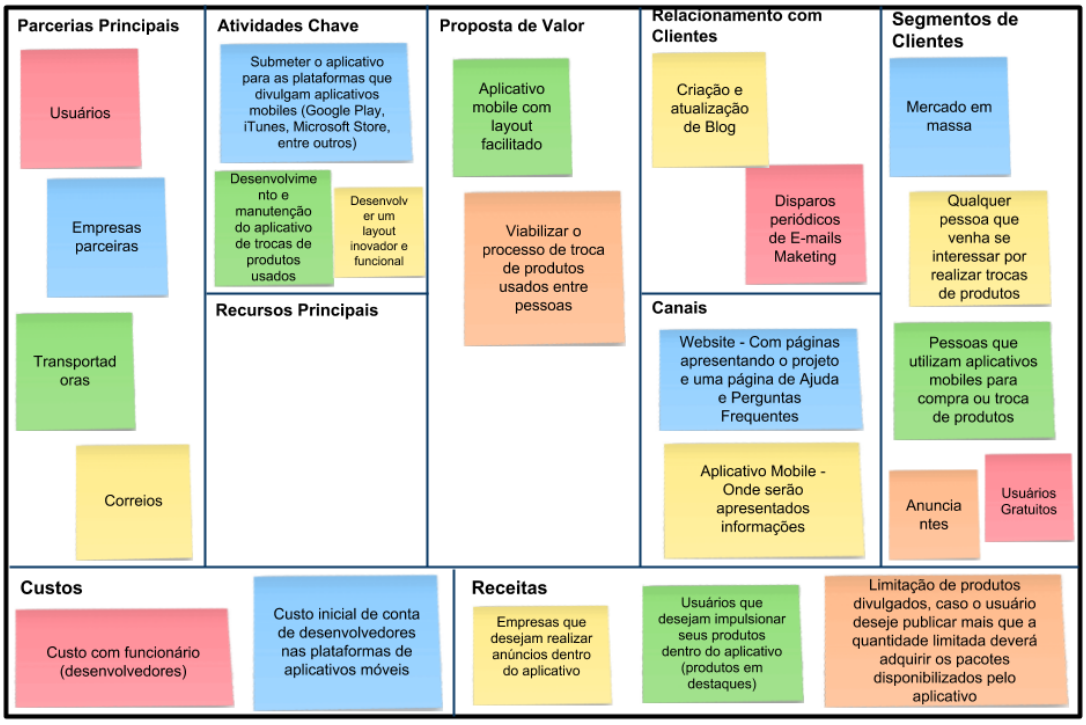
meta o estudo e desenvolvimento de ideias inovadoras, com um baixo custo de manutenção e uma grande possibilidade de gerar lucros rapidamente.

3.3 Modelo Canvas

O modelo Canvas (*Business Model Generation*) é uma metodologia desenvolvida por Alex Osterwalder, durante sua tese de doutorado. Dividido em 9 componentes essenciais para todas as organizações, em formato de um quadro, para permitir que o negócio seja analisado de forma completa, deixando bem mais fácil as possíveis alterações para diversas ideias ou situações (MOTA, 2018).

Com essa metodologia, qualquer empresa pode criar modelos de negócio de suas empresas ou até recriar de outras. Esses componentes do modelo Canvas cobrem as 4 áreas de um negócio, que são: clientes, oferta, infraestrutura e viabilidade financeira. Criado em meados de 2000 o Canvas já possui várias empresas que utilizam essa metodologia como *International Business Machines* (IBM), Ericsson, governo canadense, entre outros.

Figura 1 – Modelo Canvas



Fonte: Os autores

Os 9 componentes do modelo Canvas são descritos a seguir:

1) Segmentação do Cliente: é nesse bloco onde é definido o segmento de mercado onde o projeto irá afetar. Deve ser definido os principais clientes para o projeto.

2) Oferta de Valor: é onde a empresa define quais serão os principais benefícios que o cliente terá utilizando seu produto. É neste momento em que a empresa pode refletir e descobrir seus diferenciais diante dos concorrentes.

3) Canais: É definido por onde a empresa vai trabalhar o *marketing* do produto diante de seus clientes.

4) Relacionamento: É onde a empresa descreve sua estratégia para evitar que seus clientes troquem seu produto pela concorrência.

5) Fontes de Renda: O bloco de receita é onde a empresa determina o meio de cobrança do produto. Alguns exemplos são: venda, assinatura, licença, entre outros.

6) Recursos Chave: É listado os recursos fundamentais para que a empresa tenha sucesso na produção do produto.

7) Atividades Chave: Junto com os recursos chave, as atividades são as ações que devem ser feitas de forma constante para que o negócio funcione de maneira adequada.

8) Parcerias Chave: São os itens/atividades terceirizadas, ou seja, qualquer tipo de atividade ou matéria-prima fornecida por outra empresa deve ser listado neste bloco.

9) Estrutura de Custo: É onde a empresa define todos os principais custos com maior peso financeiro derivado de operações do negócio.

A Figura 1 do modelo Canvas desenvolvido para o projeto, contendo cada um dos nove componentes é discutido em documento disponível no repositório do projeto (REIS e PERES, 2018).

4 Definição dos Processos da Aplicação *Mobile*

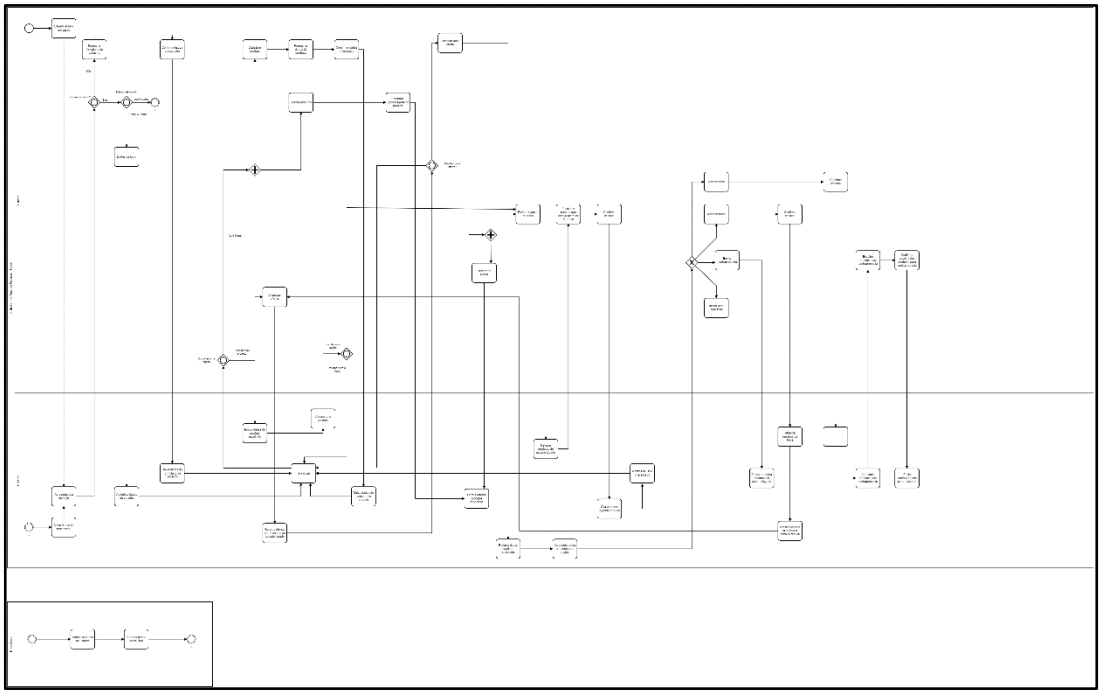
Nos tópicos a seguir serão tratadas as definições de todos os processos de documentação do projeto desenvolvido, que visa auxiliar no entendimento e visão dos processos presentes na aplicação e possíveis manutenção e melhorias que o projeto possa vir a sofrer.

O repositório das documentações (BPMN, UML, Diagrama de Caso de Uso, Diagrama de atividade, Diagrama de estados, Diagramas de sequência, MER e DER) deste projeto é encontrado no (REIS e PERES, 2018).

4.1 BPMN

O *Business Process Modeling Notation* (BPMN) é uma notação que representa por meio de diagramas os processos de negócio. A modelagem é importante para esclarecer e tornar os processos visualmente, facilitando tanto em análise de melhorias quanto de automação dos processos (ARANTES, 2014).

Figura 2 – BPMN



Fonte: Os autores

O BPMN deste projeto é encontrado no repositório do projeto (REIS e PERES, 2018).

4.2 UML

O *Unified Modeling Language* ou Linguagem de Modelagem Unificada (UML) é uma linguagem de modelagem que visa auxiliar os analistas de *software* a definir os requisitos, dinâmica dos processos, estrutura lógica, o comportamento do software. Essas características são determinadas antes mesmo do *software* entrar em desenvolvimento (FREITAS, 2008).

4.3 Diagrama de Caso de Uso

O diagrama de Caso de Uso é responsável por apresentar os processos do sistema do ponto de vista do usuário, descrevendo as principais funções do sistema (RIBEIRO, 2012).

Esse diagrama é composto por quatro partes, sendo elas:

- O cenário: responsável pelas sequências de eventos;
- Os atores: que representa o tipo de usuários;
- O Caso de Uso: representando a funcionalidade ou tarefa;
- E a comunicação que liga um ator à um Caso de Uso.

4.4 Diagrama de Atividade

O diagrama de atividade tem como objetivo principal, utilizando o ponto de vista funcional do sistema, especificar o funcionamento do software. Tem semelhanças a um fluxograma, com isso fornece uma maneira mais lógica e menos abstrata do sistema em seu nível micro ou macro (VENTURA, 2016).

4.5 Diagrama de Estados ou Diagrama de Máquina de Estados

O diagrama de estados, também conhecido como diagrama de máquina de estados, tem diversos tipos de usabilidade, sendo as principais ilustrar cenários de caso de uso em um contexto de negócios, descrever como os estados de um objeto e seu tempo de vida, apresentar o comportamento geral de uma máquina de estado, entre outros (LUCIDCHART, 2018).

4.6 Diagrama de Sequência

Diagrama de sequência é responsável por descrever como, e em qual ordem, grupos de objetos trabalham, são conhecidos como diagramas de eventos. Esse diagrama ilustra os detalhes de um Caso de Uso, modelando a lógica de um processo, operação ou função (LUCIDCHART, 2018).

4.7 Modelo Entidade Relacionamento (MER)

O Modelo Entidade Relacionamento, é um modelo que visa descrever entidades envolvidas em um modelo de negócio, com seus atributos e relacionamentos. De maneira geral, esse modelo representa a estrutura que o banco de dados possuirá (RODRIGUES, 2014).

4.8 Diagrama Entidade Relacionamento (DER)

O Diagrama Entidade Relacionamento é a representação gráfica e principal ferramenta do Modelo Entidade Relacionamento, muitas vezes utilizado na criação ou análise do banco de dados relacionais (RODRIGUES, 2014).

5 Resultados

Nesta seção detalha-se o desenvolvimento de toda a aplicação, desde o *front-end*, que foi o aplicativo *mobile* para os usuários, até a parte do *back-end*, que foi utilizado pela aplicação para se comunicar com o banco de dados.

5.1 Telas da Aplicação

Após todas as fases de planejamento e desenvolvimento do projeto, terminamos a primeira versão da aplicação. A seguir serão apresentadas algumas telas desenvolvidas no decorrer do projeto.

5.1.1 Login

A tela inicial da aplicação no primeiro acesso é a tela de *login*, que pode ser vista na Figura 3, onde o usuário deve colocar seu "*login*" e senha que foi cadastrado, também é possível acessar o aplicativo com o Facebook, ou até criar um novo cadastro.

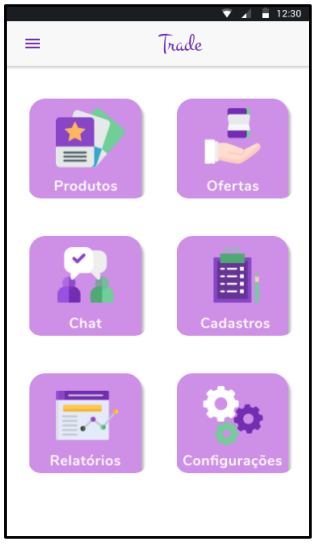
Figura 3 - Tela de login da aplicação

Fonte: Os autores

5.1.2 Menu

Após realizar o login, o usuário deve acessar o menu e selecionar uma das opções disponíveis, entre elas uma lista de produtos disponíveis para troca. Outra opção é visualizar todas as ofertas que fez a outros usuários ou que recebeu, também é possível visualizar *chats*, um pequeno relatório sobre as trocas já realizadas na plataforma e por último é possível selecionar um submenu chamado "cadastros", onde pode visualizar seu perfil ou cadastrar/editar um novo produto. Essas opções são ilustradas na Figura 4.

Figura 4 - Menu do aplicativo



Fonte: Os autores

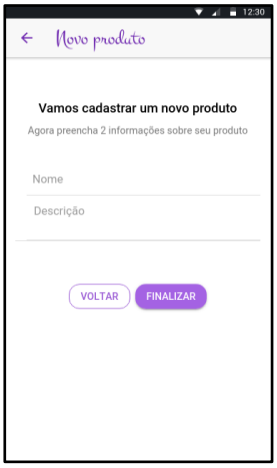
5.1.3 Cadastro

Uma das opções citadas é a área dos seus produtos, onde o usuário pode editar um produto que ele já havia criado ou cadastrar um novo em alguns passos. Para criar um novo produto, primeiro ele deve colocar algumas imagens do produto novo, e preencher as informações para o novo produto ser criado (Figuras 5 e 6), com isso já vai estar liberado para outras pessoas fazerem ofertas.

Figura 5 - Cadastro de produto

Fonte: Os autores

Figura 6 – 2ª fase de cadastro de produtos



Fonte: Os autores

5.1.4 Oferta de Troca

A principal função da aplicação, são as ofertas de troca que podem ser realizadas por qualquer usuário cadastrado, basta apenas escolher um produto que deseja e ir para a tela de finalizar oferta (Figura 7), onde o usuário poderá escolher um ou mais de seus produtos para oferecer para o dono do produto que ele deseja.

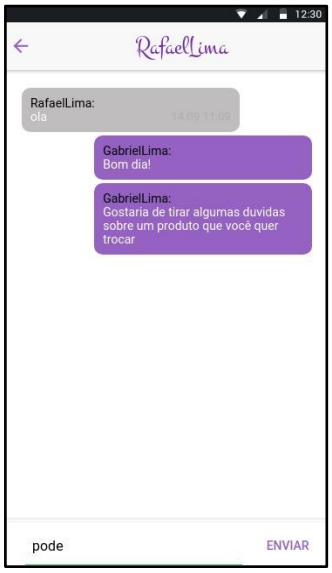
Figura 7 - Tela de produtos para oferecer a troca

Fonte: Os autores

5.1.5 Chat

Conforme ilustra a Figura 8, essa seção de *chat* é destinada para troca de mensagens a fim de esclarecer possíveis dúvidas que os usuários possam ter sobre os produtos.

Figura 8 - Tela de chat para comunicação entre usuários



Fonte: Os autores

5.2 *Back-end*

Para evitar ao máximo que a aplicação execute tarefas pesadas como consultas em um banco de dados e processamento de regras de negócio, decidimos desenvolver uma Api Rest (*Representational State Transfer*), onde o aplicativo terá apenas que passar alguns

parâmetros e acessar as *Uniform Resource Locator* ou em português Localizador Padrão de Recursos (URL) com os métodos de requisição HTTP (*GET*, *UPDATE*, *DELETE*, *POST*).

Inicialmente, para a parte do *back-end* foi usado a linguagem Node, que é uma plataforma *open source* que roda códigos em JavaScript do lado do servidor e conta com vários pacotes auxiliares para se desenvolver uma API. Dois pacotes utilizados para desenvolver a estrutura inicial da aplicação foram:

1) *Express*: Utilizado para montar a base da API, como as rotas, *middlewares* de segurança e erros, e porta do servidor.

2) *Consign*: Foi utilizado para realizar todo o mapeamento dos arquivos da API, para que fosse criado uma ordem de carregamento, evitando assim qualquer erro de sincronização dos arquivos.

A Figura 9 ilustra como foi utilizado esses dois pacotes, no método "*load*" (linhas 1 a 6 e 20 a 22) onde carregamos todos os arquivos necessários em uma determinada ordem com o Consign e em "*app.listen*" (linha 11 a 13), onde é definido a porta onde a API estará disponível com o Express.

Figura 9 - Exemplo com Consign e Express

```
1  load()
2    .include('./configs/db.js')
3    .then('./api/configs/middleware.js')
4    .then('./api/routers')
5    .then('./api/configs/info.js')
6    .into(app)
7
8  // start server
9  const port = process.env.PORT || 8000;
10
11 const server = app.listen(port, () => {
12   console.log(`Server start in port ${process.env.PORT}`);
13 });
14 console.log(server);
15 // dependencias socket
16 const io = require('socket.io')(server);
17
18 io.app = app;
19 // carregando modulos
20 load()
21   .include('socket')
22   .into(io)
23
```

Fonte: Os autores

Outra funcionalidade do pacote Express é a possibilidade de se criar os *endpoints* que a API necessita. Dentro dessas rotas pode se obter parâmetros passados e

direcioná-los para outra parte do código, com isso temos uma aplicação mais organizada e de fácil manutenção.

Figura 10 - Exemplo de rotas utilizando o Express

```
1   app.route('/products')
2     .get(auth.authenticate, async (req, res, next) => {
3       try {
4         res.locals = await product.all(req.user);
5         next();
6       } catch (e) {
7         throw e.message;
8       }
9     })
```

Fonte: Os autores

Esse exemplo de roteamento pode ser observado na Figura 10, onde pode ser definido a URL "/products" (linha 1 a 9) e todos os métodos que essa URL disponibiliza. Nessas rotas também podem ser definidas funções auxiliares conhecidas como "middlewares" onde todas as rotas deverão passar. Na Figura 10 é utilizada a função de "auth.authenticate" (linha 2), onde fica obrigatório que todos que acessarem essa URL deverão passar o parâmetro "Authorization" que é uma chave única que cada usuário possui.

A segunda parte do *back-end* é sobre a comunicação da API com o banco de dados (PostgreSQL) escolhido pelo time de desenvolvimento. A biblioteca utilizada para fazer essa comunicação foi o *Sequelize* um *Object Relational Mapper* (ORM) feito para *back-end* desenvolvido em JavaScript.

Um ORM é uma técnica de mapeamento objeto relacional, onde é feita a relação dos objetos com os dados salvos em banco. Essa técnica vem se tornando bastante conhecida, pois, além de aumentar a produtividade, ele deixa bem abstrato as consultas SQL.

Na Figura 11, é possível observar a definição da tabela de produtos, onde é definido o campo "id" como chave primária e auto incremento (linha 3 a 6), e os campos "nome" e "id_user" como obrigatórios (linha 8 a 21). Caso a biblioteca não receba um desses campos obrigatórios, ele automaticamente retorna uma mensagem de aviso e não permite a inserção da tupla na tabela. Também é possível observar todos os relacionamentos com outras tabelas, definindo o tipo de relacionamento e suas respectivas chaves estrangeira.

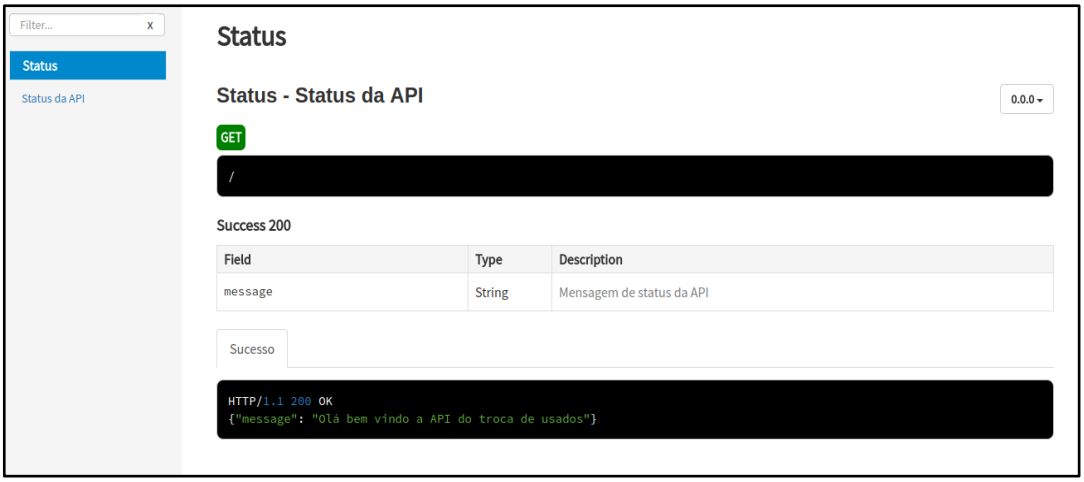
Figura 11 - Exemplo de *Models* para o Sequelize

```
module.exports = (sequelize, dataType) => {
  const Product = sequelize.define('tb_products', {
    id: {
      type: dataType.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    name: {
      type: dataType.STRING(30),
      validate: {
        notEmpty: {
          msg: 'Campo obrigatório'
        }
      }
    },
    description: {
      type: dataType.TEXT
    },
    id_user: {
      type: dataType.INTEGER,
      allowNull: false
    }
  },
  {
    timestamp: true,
    paranoid: true
  }
  );

  Product.associate = (models) => {
    Product.belongsTo(models.tb_users, {foreignKey: 'id_user'});
    Product.hasMany(models.tb_images, { as: 'images', foreignKey: 'id_product', onDelete: 'cascade' });

```

Fonte: Os autores

Por fim, para facilitar o uso da API foi desenvolvida uma documentação de todos os *endpoints* disponíveis e com exemplos de como deve ser utilizado. Essa documentação está disponível em um servidor, é disponibilizado um menu com todas as requisições e exemplos de uso como ilustrado na Figura 12.

Figura 12 - Imagem da documentação da API



Fonte: Os autores

5.3 *Front-end*

A segunda parte do sistema é a aplicação móvel, desenvolvida no *framework* Ionic que utiliza a linguagem TypeScript para que seja desenvolvido o aplicativo em uma linguagem e compilado para 3 sistemas operacionais diferentes: Android, iOS e Windows Phone.

O *framework* vem com várias funcionalidades para facilitar o desenvolvimento da aplicação e a divisão entre todas as tarefas, deixando assim um código muito mais organizado. Um desses exemplos é toda a comunicação que o aplicativo *mobile* faz com a API. A Figura 13 contém as linhas 9 a 11, onde é definido um método *GET* para buscar informações dos produtos a serem trocados.

Figura 13 - Imagem de um *provider* da aplicação *mobile*

```
1   constructor(
2     public http: HttpClient,
3     private configs: ConfigsApi,
4     private fileTransfer: FileTransfer
5   ) {
6     this.httpOptions = this.configs.getHeaders();
7   }
8
9   listProducts(): Observable<any>{
10    return this.http.get(`${this.url}products`, this.httpOptions);
11  }
```

Fonte: Os autores

Como foi ilustrado na Figura 6, há arquivos separados por funcionalidades, onde os responsáveis por acessar essas *providers* que consultam a API, são as P*ages*. Cada página é dividida em 3 arquivos principais (.ts, .html, .css), onde o .html é responsável por exibir a página para o usuário, .css é onde fica todo o estilo da página e efeitos, e o .ts é onde a lógica fica, e ainda o acesso ao *hardware* do dispositivo e aos *providers*. Por ser construído em módulos, o *framework* Ionic necessita apenas da injeção das dependências que a página vai precisar em seu funcionamento. A Figura 14, da linha 1 a 5, são definidos todos os componentes que a página vai usar, na linha 12 a 18 é definida a função que utiliza o *provider* para buscar as informações que serão exibidas para os usuários.

Outra parte que foi trabalhada no projeto foi a reutilização de código, onde com a ajuda do Ionic, foi possível criar componentes que podem ser inseridos em várias páginas diferentes, evitando assim que o código seja duplicado em várias páginas. Na Figura 15, é definido um simples componente que têm como objetivo criar um *slider* de imagens (será mostrado mais à frente a tela em que esse *slider* é utilizado). Com isso, qualquer página que precisar de um *slider*, deve somente injetar a dependência e utilizar o componente, sem ter que reescrever o mesmo novamente.

Figura 14 - *Page* com injeção de dependências e consulta aos *providers*

```
constructor(
    public navCtrl: NavController,
    public navParams: NavParams,
    private product: ProductProvider
  ) {
  }

  ionViewWillEnter() {
    this.findAllProducts();
  }

  findAllProducts() {
    this.product.listProducts().subscribe(response => {
      this.products = response.data;
    }, err => {
      console.log(err);
    });
  }
```

Fonte: Os autores

Figura 15 - Definição de um componente

```
@Component({
  selector: 'slider',
  templateUrl: 'slider.html'
})
export class SliderComponent {
  @Input() images;
  @ViewChild(Slides) slides: Slides;

  constructor() {
  }
```

Fonte: Os autores

6 Conclusão

Em geral finalizamos o objetivo principal do trabalho, produzir um aplicativo móvel para mais de uma plataforma (Android, iOS e Windows Phone), com um *design* simples e intuitivo para auxiliar qualquer usuário em encontrar um produto que deseje realizar alguma troca por outro que ele possua, para auxiliar a crescente onda da chamada “Economia Compartilhada”.

Este trabalho buscou unir todo o conhecimento adquirido no decorrer do curso, sobre desenvolvimento híbrido com o *framework* Ionic, aspectos relacionados a Banco de Dados, Engenharia de *Software* e metodologias ágeis de projetos, para proporcionar ao usuário maior facilidade na troca de usados e velocidade na negociação com outras pessoas.

Esse aplicativo contém algumas funcionalidades básicas, tendo em mente algumas implementações futuras para melhorar ainda mais a usabilidade do aplicativo. Entre essas implementações está o uso do GPS do aparelho para mostrar em tempo real os produtos mais próximos dele e também um sistema de pontuação para que os mais bem qualificados estejam sempre no topo, dando assim mais visibilidade para seus produtos.

Referências

AGUIAR, Anderson. **Aplicativo Nativo ou Aplicativo Híbrido: Qual a melhor opção?**. 2017. Disponível em: <https://secaoweb.com.br/blog/aplicativo-nativo-vs-aplicativo-hibrido>. Acesso em: 25 maio 2018.

ALEXANDRE, Hugo. **De forma simples o que é uma Startup?.** 2016. Disponível em: <https://www.hugoalexandretrindade.com/de-forma-simples-o-que-e-uma-startup>. Acesso em: 04 jun. 2018.

ANGULAR. **Understanding Components**. Version 1.7.4. [S.l.]. 2018. Disponível em: <https://docs.angularjs.org/guide/component>. Acesso em: 26 abr. 2018.

APACHE, Software. **Guide Overview**. 2015. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 26 abr. 2018.

ARANTES, Rhaíssa Nogueira. **Artigo Introdução ao Business Process Modeling Notation (BPMN)**. 2014. Disponível em: <https://www.devmedia.com.br/introducao-ao-business-process-modeling-notation-bpmn/29892>. Acesso em: 12 set. 2018.

CÉSAR, Bruno. **Republic: Aplicação *mobile* para divulgar e procurar vagas em repúblicas universitárias**. 2017. 49 p. Artigo (Sistemas de Informação) - Uberlândia, [S.l.], 2017. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/19599/1/RepublicRepublicAplicacaoMobile.pdf>. Acesso em: 26 abr. 2018.

FEAUSP, **REGE**, 2016. Disponível em: <http://www.spell.org.br/documentos/ver/43214/economia-compartilhada-e-consumo-colaborativo>. Acesso em: 20 mar. 2018.

FELIPINI, Dailton. **EMPREENDEDORISMO NA INTERNET: Como agarrar esta nova oportunidade de negócios**. 3 Edição. ed. [S.l.]: LeBooks Livraria de Ebooks, 2012. 25 p. Disponível em: <http://www.ebooksbrasil.org/adobeebook/empreendedorismo.pdf>. Acesso em: 03 set. 2018

FREITAS, Márcio Nogueira. **Artigo UML** 2008. Disponível em: <https://www.devmedia.com.br/uml/8579>. Acesso em: 12 set. 2018.

IBGE (Org.) - Instituto Brasileiro de Geografia e Estatística. Acesso à Internet e à Televisão e Posse de Telefone Móvel Celular para Uso Pessoal. Rio de Janeiro: [s.n.], 2016. 90 p. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv99054.pdf>. Acesso em: 04 jun. 2018.

IONIC, Framework. **Core Concepts**. Version 3.0. [S.l.]. 2018. Disponível em: <https://ionicframework.com/docs/intro/concepts>. Acesso em: 28 abr. 2018.

JHEYSON, Allan. **Desenvolvimento de Aplicativos Híbridos com o Ionic Framework**. 2017. 16 p. Artigo (Sistemas de Informação) - Escola Regional de Informática do Piauí, [S.l.], 2017. Disponível em: <http://www.eripi.com.br/2017/images/anais/minicursos/13.pdf>. Acesso em: 28 abr. 2018.

KAZAP. **Aplicativo Nativo vs. Web App: o que são, as diferenças e qual escolher.** 2015. Disponível em: <http://www.kazap.com.br/aplicativo-nativo-vs-web-apps/>. Acesso em: 12 set. 2018.

LUCIDCHART. **Artigo Diagrama de máquina de estados 2018**. Disponível em: < https://www.lucidchart.com/pages/pt/diagrama-de-máquina-de-estados>. Acesso em: 12 set. 2018.

LUNA, Polyana. **Empreendedorismo Start Up: um Estudo de Caso em uma Empresa de Tecnologia no Estado do Pará**. 2012. 16 f. Artigo (Simpósio de Excelência em Gestão e Tecnologia) - SEgT, [S.l.], 2012. Disponível em: <https://www.aedb.br/seget/arquivos/artigos12/30616273.pdf>. Acesso em: 04 jun. 2018.

MELLO, Lisilene; PETRINI, Maira; CLARISSA, Ana. **Economia compartilhada e consumo colaborativo: o que estamos pesquisando?**. 2016. 08 p. Artigo (Gestão e Sustentabilidade) -

MISZURA, Jan; DIVINO, Gilcimar. **Desenvolvimento em Smartphones - Aplicativos Nativos e Web**. 2012. 8 p. Artigo (Sistemas de Informação) – PUC Goiás, [S.l.], 2012. Disponível em: <http://www.cpgls.pucgoias.edu.br/7mostra/Artigos/AGRARIAS%20EXATAS%20E%20DA%20TERRA/Desenvolvimento%20em%20Smartphones%20-%20Aplicativos%20Nativos%20e%20Web.pdf>. Acesso em: 19 set. 2018.

MOTA, Gleison. **CANVAS: O que é e para que serve?**. Disponível em: <http://www.administradores.com.br/artigos/empreendedorismo/canvas-o-que-e-e-para-que-serve/109236/>. Acesso em: 17 set. 2018.

PADILHA, Telma. **A IMPORTANCIA DO EMPREENDEDORISMO COMO ESTRATÉGIA DE NEGÓCIO**. 2011. 62 f. Artigo (Engenharia de Produção) - UNIMEP, [S.l.], 2011. Disponível em: <http://www.unisalesiano.edu.br/biblioteca/monografias/53972.pdf>. Acesso em: 04 jun. 2018.

QUINTANILHA, Pedro. **O Que é Ser Empreendedor**. 2015. Disponível em: <https://mentalidadeempreendedora.com.br/desenvolvendo-a-mentalidade-empreendedora/o-que-e-ser-empreendedor/>. Acesso em: 04 jun. 2018.

REIS, Rafael; PERES, Vinicius. **Repositório contendo documentações, códigos e pesquisas realizadas para o projeto**. Disponível em: <https://github.com/viniiciusmarques/app-trade-documentation>. Acesso em: 18 set. 2018

REZENER, Beatriz; BRITTO, João. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataforma**. 2016. 63 p. Monografia (Engenharia de Software) - Faculdade UnB Gama - FGA, Universidade de Brasília - UnB, [S.l.], 2016. Disponível em: <https://fga.unb.br/articles/0001/5113/Beatriz_Joao_TCC_Aplicativos_M_veis.pdf>. Acesso em: 19 set. 2018.

RIBEIRO, Leandro. **Artigo O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML** 2012. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 12 set. 2018.

RODRIGUES, Joes. **Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)**. 2014. Disponível em: < https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Acesso em: 19 set. 2018.

VENTURA, Plínio. **Artigo Entendendo o Diagrama de Atividades da UML 201**6. Disponível em: <https://www.ateomomento.com.br/uml-diagrama-de-atividades>. Acesso em: 12 set. 2018.

WAHLIN, Dan. **5 Key Benefits of Angular and TypeScript**. 2017. Disponível em: <https://blog.codewithdan.com/5-key-benefits-of-angular-and-typescript>. Acesso em: 28 abr. 2018.