Presente!: desenvolvimento de *Marketplace* para *delivery* de presentes

Ana Flavia Bezerra da Silva, Daniella Veronez Marques Barros, Elison Rodrigo Maciel Trindade, Henrique Camargo Jacó Hessel, Julia Siqueira Barros, Sushila Vieira Claro[1]

Orientador: Professor Leonardo Massayuki Takuno

Faculdade Impacta de Tecnologia
São Paulo, SP, Brasil
17 de dezembro de 2021

Resumo. O presente trabalho tem como objetivo trazer o relatório técnico, subsidiado em pesquisa de viabilidade mercadológica, do desenvolvimento de um aplicativo *mobile* de *marketplace* para *delivery* de presentes. Foi realizado, inicialmente, o mapeamento dos requisitos necessários para criação de uma solução, bem como análise de sua viabilidade. Optamos por realizar o desenvolvimento de um produto mínimo viável **(**MVP) com o *framework* Ionic em conjunto com o banco de dados MongoDB, voltado para a aplicação mobile.

Palavras-chaves: *Marketplace*. *Delivery*. Aplicativo *mobile*.

1. Introdução

Com a pandemia de COVID-19 e o isolamento social, o consumo através de ferramentas online, com respectiva entrega em domicílio, aumentou exponencialmente (31%, em comparando o ano de 2019 com o de 2020 – ROCHA, 2021; 38% quando comparando 2020 com 2018 – INOVASOCIAL, 2020). O comportamento de compra do consumidor, por conta de restrições físicas, mudou durante o período (39% passaram a encomendar produtos variados de forma online, 41% passou a utilizar serviços de entrega – em específico, de refeições, 68% consideraram como variável de alta importância a disponibilidade da entrega de suas compras; ao longo prazo, 74% planejam utilizar mais meios online para compras, sendo que 34% afirmam que utilizarão com certeza *marketplaces* digitais - EY PARTHENON, 2020). Outros relatórios também apontam o crescimento de vendas através de meios digitais (SEBRAE, 2020; ITAU, 2021) como um hábito que estava em crescimento moderado no período anterior à pandemia e que cresceu exponencialmente durante o isolamento.

Dentro das possibilidades do mercado, os membros do grupo decidiram empreender em uma incubadora de soluções digitais: a *Devlicious*, com a missão de captar as necessidades dos usuários e desenvolver soluções digitais para seus problemas.

Como primeiro item de portfólio, optamos por desenvolver, em nosso trabalho final do curso de Análise e Desenvolvimento de Sistemas, um *marketplace* que atendesse uma demanda ignorada no período da pandemia: o *delivery* de presentes. A ideia surgiu por conta das reclamações ouvidas de amigos e familiares dos membros, que diziam não poder mandar para determinado remetente, em ocasiões especiais, nenhum outro tipo de presente que não fosse comida, por ser a única opção disponibilizada nos aplicativos desse tipo (*iFood*, *Rappi*, *Uber Eats*).

[1] Os autores podem ser contatados, respectivamente, pelos seus correios eletrônicos: anaflaviabsilva@outlook.com, daniella.veronezbarros@gmail.com, roelisontrindade@gmail.com, henry_jaco@hotmail.com, juliasbarros@hotmail.com, sushilaqq@hotmail.com.

1.1. Apresentação da empresa *Devlicious*

A *Devlicious* (Figura 1), *startup* criada pelos membros deste grupo de trabalho, tem como principal intenção ser uma incubadora de soluções digitais. Nosso principal negócio é ouvir as reclamações e insatisfações dos usuários e pensar soluções para seus problemas; entendemos que uma boa ideia é trazer novas visões sobre um determinado problema, que simplifique sua resolução (a inspiração veio da série *Small Thing, Big Idea*, material original do TED[2]). Com isso, planejamos focar no desenvolvimento de aplicativos voltados para nichos não percebidos/explorados.

Sendo uma empresa estreante no mercado – e atuando nela como ofício secundário – o principal problema encontrado é não possuir ainda nenhum produto no portfólio que mostre o nosso potencial em inovação e a qualidade do time de trabalho. A partir disso, nos reunimos e discutimos diversas ideias que poderiam se materializar, nos ajudando a colocar nossa marca no mercado.

Figura 1 – Logotipo da empresa *Devlicious*. O contraste da letra cursiva (sensação de fluidez) com o ícone do monstrinho pixelizado (remetendo ao digital) resume a intenção da empresa, que é "desenrolar problemas através de bytes".

Fonte: elaborado pelos autores (2021).

1.2. Entraves no desenvolvimento do aplicativo Presente!

Em síntese, a declaração da adversidade que foi identificada é: o **problema** causado pela pandemia sanitária (isolamento) **afeta** o comércio de itens presenteáveis em diversas categorias de produtos **devido** a falta de um aplicativo/*marketplace* que ofereça a mesma comodidade da escolha e entrega diretamente ao remetente, como ocorre em aplicativos de entrega de refeições, por exemplo. Os **benefícios** do Presente! são:

- permitir presentear um remetente, independente da distância (a primeira versão atende apenas a cidade de São Paulo), de forma segura;
- possibilitar a seleção do presente em um ambiente que já filtre por categoria e por opções possíveis na região da entrega do presente;
- permitir pagamento direto pelo aplicativo;
- estimular pequenos comércios que possuam em seus estoques itens presenteáveis e que não foram contemplados pelos aplicativos de entrega de refeições;
- fazer a mediação direta com o entregador;
- atender um nicho de mercado surgido no período da pandemia (ainda não explorado) e que vai ao encontro das novas tendências de comportamento do consumidor.

[2] A série pode ser assistida aqui: https://bit.ly/3g6zt1R

Como explicitado anteriormente, a *Devlicious* é uma *startup* que tem como data de início o presente ano; assim, alguns problemas básicos dentro da nossa estrutura foram identificados:

- **Problema principal:** portfólio inexistente, o que não permite que os clientes possam contratar serviços com a segurança do trabalho que será feito;
- **Solução principal:** desenvolvimento de aplicativo que servirá para composição do portfólio da empresa, a fim de fortalecer sua imagem comercial.

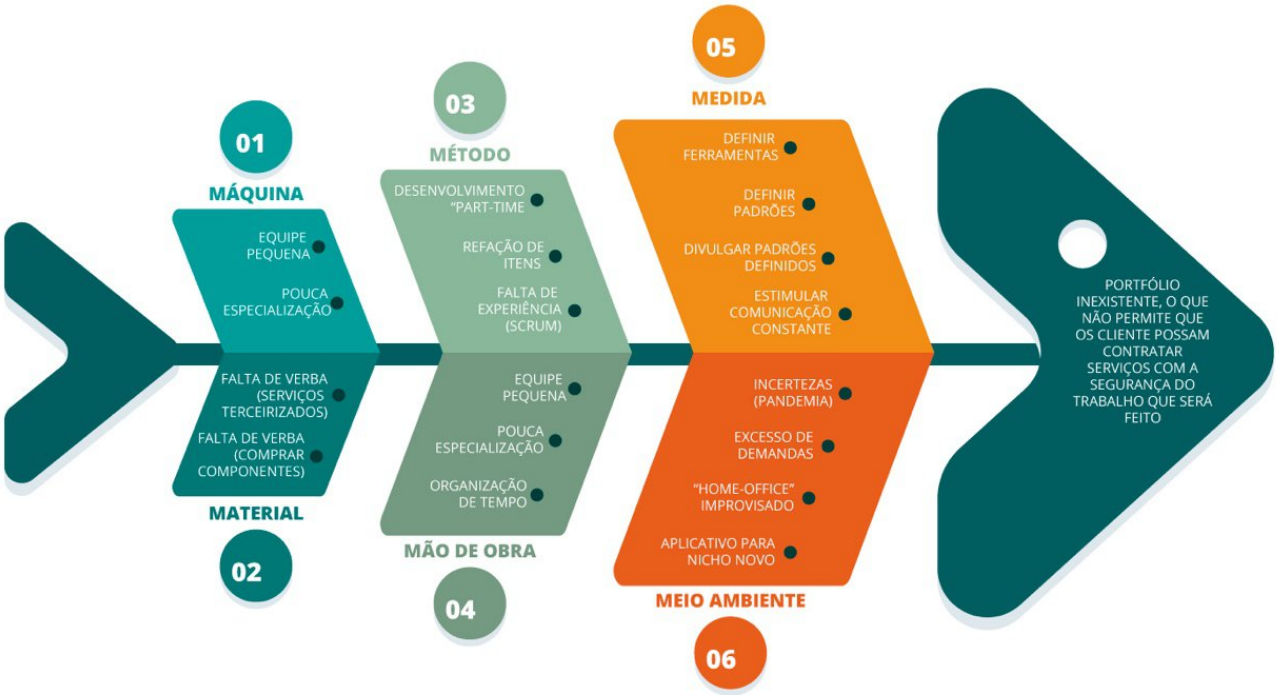
Dentro do problema principal, temos ainda as seguintes situações:

- equipe pequena, com pouca especialização;
- desenvolvimento de aplicativos *part-time*[3] (todos os membros do grupo trabalham de 6-10 horas por dia, de segunda a sexta, o que permite pouco tempo de dedicação ao projeto);
- pouca experiência com a metodologia SCRUM;
- dificuldade na organização do tempo;
- dificuldade na definição das ferramentas e padrões do aplicativo e, por consequência, atraso da divulgação destes entre a equipe;
- comunicação realizada majoritariamente por aplicativos de conversa;
- incerteza geral na vida pessoal por conta da pandemia;
- excesso de demandas de diversas áreas (casa, trabalho, faculdade)
- ambiente de trabalho improvisado, por conta da pandemia;
- incerteza nos rumos do aplicativo, por este ser direcionado a um novo nicho;
- refação de diversos itens para adequação à ideia proposta;
- pouquíssimo fluxo de caixa para aquisição de serviços terceirizados ou componentes.

Ainda que haja bastante documentação de aplicativos pioneiros no ramo de *delivery*, a complexidade dos documentos acabaria se tornando um problema para desenvolvermos nosso produto se tomarmos como base essas informações. Por conta disso, as discussões sobre o desenvolvimento do aplicativo “Presente!” têm trazido diversas fontes, ferramentas e ideias para sua execução. Concomitantemente, os membros do grupo realizaram cursos referentes às áreas em deficiência e revisaram os conteúdos vistos durante a graduação. A fim de explicar melhor a situação, desenvolvemos o diagrama[4] na Figura 2:

[3] O trabalho *part-time* (que pode ser traduzido também como “em tempo parcial”) é uma forma de atividade em que a jornada de trabalho semanal é menor que a de um emprego em tempo integral (*full-time*).
[4] “O Diagrama de Ishikawa, também conhecido como ‘Diagrama de Causa e Efeito’ ou ‘Espinha de Peixe’, permite estruturar hierarquicamente as causas de determinado problema ou oportunidade de melhoria.” (ESALQ/USP, s/d, s/p).

Figura 2 – Diagrama de Causa e Efeito (Diagrama de Ishikawa) da *Devlicious*.



Fonte: adaptado de *Freepik*/ elaborado pelos autores (2021).

1.3. *Stakeholders* e restrições

Como *stakeholders* do projeto, temos os seguintes atores:

- **Usuário:** toda pessoa que se cadastrar no aplicativo. Divide-se em três papéis:
  - **Comprador (Cliente):** após o usuário se cadastrar no aplicativo, como padrão ele recebe o papel "Cliente", podendo consultar dados públicos da loja (*status*, endereços, avaliações, produtos), buscar e filtrar produtos, consultar a situação do pedido, rastrear entregas, realizar pagamentos e avaliar pedido (Loja e Entregador).
    - **Lojista (Loja):** quando o Cliente solicita a alteração para o papel "Loja", ele pode enviar a documentação referente a empresa (CNPJ, comprovante de endereço etc.). A Loja herda todas as funcionalidades de Cliente, sendo adicionada a capacidade de cadastrar/alterar/excluir produtos, consultar listagens financeiras, avisar quando o pedido estiver pronto para retirada e aceitar ou negar o cancelamento do pedido.
  - **Entregador:** quando o Cliente solicita a alteração para o papel "Entregador", ele pode enviar a documentação referente a empresa (CNH, antecedentes criminais etc.). O Entregador não herda as funcionalidades de Cliente (podendo alterar entre os dois papéis). Neste papel são adicionadas as funcionalidades de receber pedido de entrega, coletar pedido de entregar e atualizar *status* de entrega.
- **Remetente:** o Remetente não é um usuário diretamente cadastrado no aplicativo. Seus dados (nome, endereço, telefone) devem ser inseridos pelo Cliente no fechamento do pedido.

- ***Devlicious*:** empresa responsável pelo desenvolvimento do aplicativo, tem como papel "Administrador", com todas as possibilidades de CRUD em todas as partes do sistema.

Definido os atores do sistema, colocamos as seguintes restrições:

- **Financeira:** o grupo não possui verba para contratação de mais mão-de-obra, tampouco para compra de serviços terceirizados ou componentes;
- **Legal:** é preciso atentar ao Código de Defesa do Consumidor, bem como para a Lei Geral de Proteção de Dados Pessoais, uma vez que o aplicativo guardará dados sensíveis (tanto pessoais quanto de pagamentos);
- **Técnica:** o programa precisa ser desenvolvido com as ferramentas que o grupo têm maior facilidade/conhecimento;
- **Humana:** o programa deve ter um *layout* intuitivo a fim de diminuir a curva de aprendizado do aplicativo;
- **Tecnológica:** o programa deve ser desenvolvido com retrocompatibilidade para atender sistemas operacionais mais antigos.

2. "Mesmo distante, presente." - Solução proposta

Como trazido na primeira parte deste relatório, a pandemia e o isolamento social trouxeram novas formas de consumir, fazendo com que o mercado de entregas crescesse de forma intensa. Entretanto, algumas áreas não foram contempladas, como, por exemplo, a que tratamos em nosso projeto de aplicativo: a entrega de presentes. Em pesquisa informal e não-documentada, o grupo conversou com diversos segmentos sociais, profissionais e etários sobre a ideia de desenvolver um *marketplace*, nos moldes do *Ifood*, focado na possibilidade de manter as relações afetivas mesmo à distância.

Assim, o primeiro passo em direção à solução do problema foi desenvolver o nome e identidade visual do aplicativo. Por contarmos em nossa equipe com um membro com formação em Publicidade e Propaganda, tornou-se relativamente fácil o desenrolar desta atividade.

Por se tratar de *ritos afetivos à distância*, optamos pelo nome "**Presente!",** com a exclamação no final, trazendo, em uma palavra, um amplo universo semântico: vigente, existente, regalo, mimo, lembrança, oferecimento, zeloso, guardado, recordado.

Nome definido, passamos à criação do mote/*slogan* para o aplicativo, pensando no aumento da atratividade do público-alvo: "**Mesmo distante, presente.".** A frase consegue sintetizar o objetivo de nosso aplicativo, uma vez que pontua a questão da distância (isolamento social), da presença (mesmo que distante) e do regalo.

Sabendo que o *design* visual de um aplicativo é grande parte responsável pelo interesse do usuário, definimos algumas diretrizes para o desenvolvimento do logotipo (Figura 3). Foram usadas as fontes *Dancing_Script* para a *combination mark*[5] e *Roboto Condensed* para o *slogan* e fonte do aplicativo (escolha feita por esta ser a fonte oficial do sistema Android; na versão para IOS, deverá ser utilizada a fonte Helvética, que é a oficial deste sistema). Como cores base, decidimos utilizar tons de laranja (por ser uma cor que transmite a sensação de alegria e renovação) e cinza-escuro, para equilibrar a paleta cromática.

[5] Também é conhecido como "Combinação de texto e símbolo", embora a forma em inglês seja mais disseminada: "O nome é bastante autoexplicativo, mas os logotipos que combinam texto e símbolo incorporam tanto imagens como palavras em seu design. Os logotipos de texto e símbolo são compostos de qualquer combinação de imagens e palavras que você escolher[...]". (TAYLOR-BRANDS, s/d, s/p)

Figura 3 – Logotipo e ícone do aplicativo “Presente!”.

Fonte: elaborado pelos autores (2021).

Em síntese, a ideia principal é proporcionar um *marketplace* onde o cliente possa escolher, em razão da região de abrangência do remetente, um presente (dentre os ofertados pelo comércio daquela região), que será entregue diretamente ao presenteado, através dos entregadores parceiros do aplicativo.

Por conta do tamanho e da complexidade do desenvolvimento do aplicativo, bem como do curto prazo e das dificuldades expostas no item 1.2, a equipe decidiu apresentar, como resultado deste trabalho, um **produto mínimo viável** (*MVP, Minimum Viable Product*[6]); assim, serão elencados, juntos com suas funcionalidades, os seguintes artefatos:

- Tela de *login*;
- Tela de cadastro de usuário;
- Tela principal (*home*);
- Tela de busca;
- Tela de produtos;
- Tela de pedido.

Dessa forma, todas as informações deste relatório (DER, requisitos, regras de negócio, funcionalidades etc.) contemplam apenas o escopo definido para o MVP, listado acima.

2.1. Requisitos do sistema

Como pontuado anteriormente, para este trabalho desenvolveremos apenas um MVP, alguns requisitos (Quadro 1) são necessários para o seu desenvolvimento.

Quadro 1 – Requisitos.

Requisito	Descrição
SSS0001	O Sistema DEVE permitir cadastro de novo usuário.
SSS0002	O Sistema DEVE validar dados de login e senha.

[6] “Produto Mínimo Viável ou MVP é uma técnica de desenvolvimento em que um novo produto é introduzido no mercado com características básicas, mas suficientes para chamar a atenção dos consumidores. O produto final é lançado no mercado somente após obter *feedback* suficiente dos usuários iniciais do produto.”. Tradução livre de THE ECONOMIC TIMES, s/d.

Requisito	Descrição
SSS0003	O Sistema DEVE, após a autenticação de login, direcionar o usuário para tela inicial.
SSS0004	O Sistema DEVE permitir busca de itens.
SSS0005	O Sistema DEVE permitir filtragem de itens (preço, distância, formas de pagamento, categorias).
SSS0006	O Sistema DEVE mostrar ao usuário listagem dos itens selecionados, endereço de entrega e pagamento antes da confirmação do pedido.
SSS0007	O Sistema DEVE permitir cadastro de produtos em uma categoria (por exemplo: Amor, Amizade, Agradecimento, Aniversário, Boa Sorte, Desculpas, Melhoras, Casamento, Crianças, Condolências, Picante).

Fonte: elaborado pelos autores (2021).

2.2. Regras de negócio

Considerando o escopo definido para o MVP, listamos a seguir as regras de negócio (Quadro 2) referente ao seu desenvolvimento.

Quadro 2 – Regras de negócio.

Regra	Descrição
RN0001	Todos os usuários são cadastrados, inicialmente, no papel de CLIENTE.
RN0002	Um usuário pode alterar seu papel para LOJA após enviar os documentos necessários e ter o cadastro alterado.
RN0003	Um usuário pode alterar seu papel para ENTREGADOR após enviar os documentos necessários e ter o cadastro alterado.
RN0004	Um CLIENTE pode selecionar um ou mais produtos da mesma LOJA no mesmo pedido.
RN0005	Um CLIENTE deve poder consultar a listagem dos produtos selecionados no pedido antes de fechar o pedido.
RN0006	Um CLIENTE deve poder selecionar a forma de pagamento de um pedido.
RN0007	Um CLIENTE deve poder validar e emitir um pedido.

Fonte: elaborado pelos autores (2021).

2.3. Descrição das funcionalidades

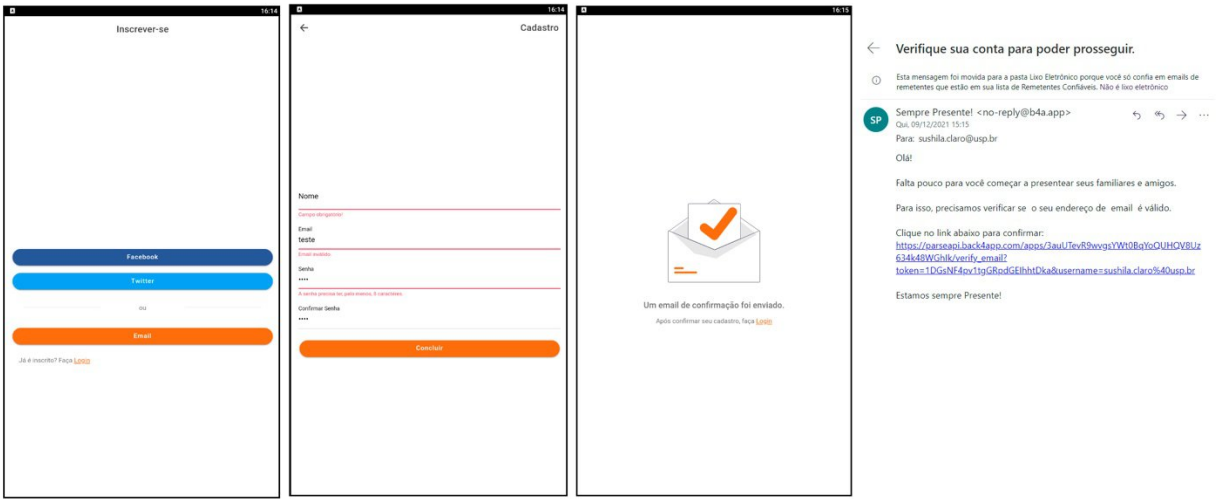
As funcionalidades desenvolvidas para apresentação do MVP, como mencionado anteriormente, são as seguintes: **tela de *login***, **tela de cadastro de usuário**, **tela principal (*home*)**, **tela de busca**, **tela de produtos** e **tela de pedido**. Detalharemos, a seguir, cada um dos tópicos.

2.3.1. Cadastro de usuário

Para utilizar o aplicativo, o usuário precisa realizar um cadastro inicial, informando seus dados pessoais (nome, CPF, endereço, CEP), bem como *e-mail* e inserir uma senha. Caso o usuário insira um valor inválido para o *e-mail* (seja por não conter "@" ou já estar cadastrado), o sistema emite um aviso informando qual a situação do erro (o mesmo ocorre com a senha, que deve seguir alguns padrões de segurança para ser validada).

Após o cadastro, o usuário recebe um *e-mail* de validação de seu endereço na caixa postal eletrônica (caso não valide a mensagem recebida, o usuário não consegue fazer *login* no aplicativo). O processo pode ser observado na Figura 4.

Figura 4 – Processo de cadastro do aplicativo "Presente!".

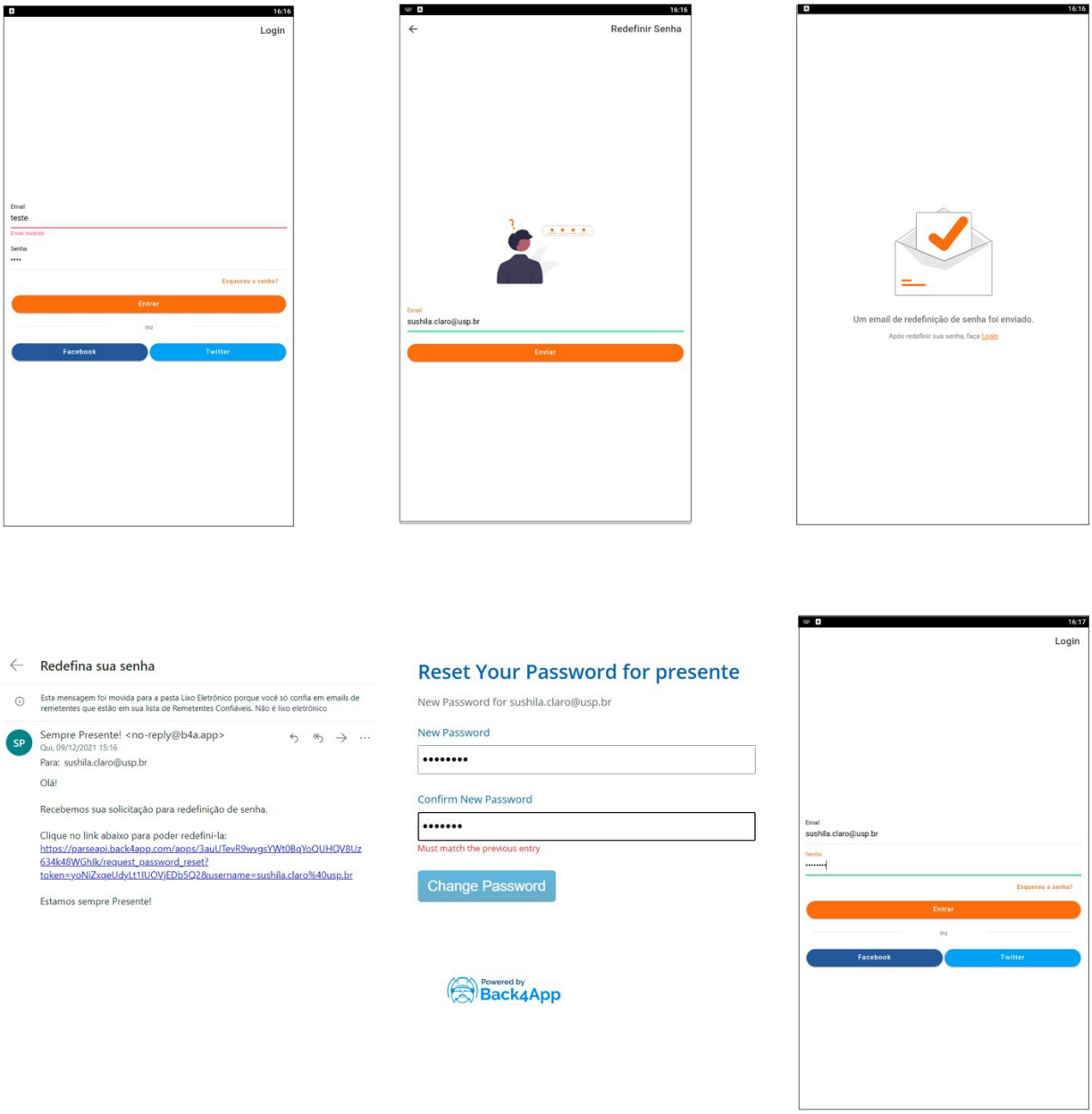


Fonte: elaborado pelos autores (2021).

2.3.2. Login

A tela de *login* serve para que os usuários que já estão cadastrados consigam entrar em seus respectivos perfis e utilizar o programa. O *login* pode ser realizado inserindo o *e-mail* e senha cadastrados. Quando o usuário coloca um *e-mail* não válido (seja por não conter "@", não estar cadastrado ou não ter sido validado), o sistema emite um aviso informando qual a situação do erro (o mesmo ocorre com a senha). O processo pode ser observado na Figura 5.

Figura 5 – Processo de login do aplicativo “Presente!”.

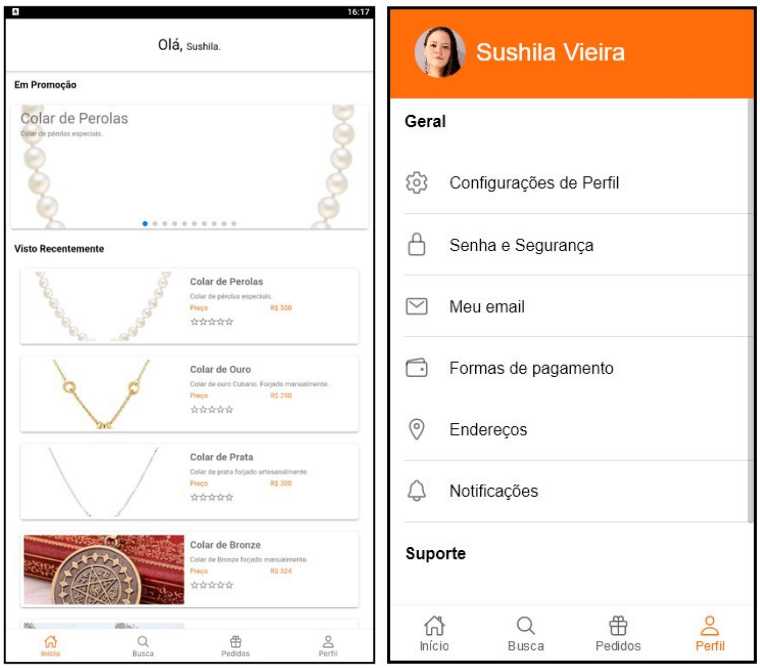


Fonte: elaborado pelos autores (2021).

2.3.3. Principal (*Home*)

É a tela principal do aplicativo onde serão apresentados para o usuário, logo após sucesso de *login*, promoções e sugestões de compra, bem como os ícones de *home* (Figura 6, à esquerda), busca e perfil (Figura 6, à direita).

Figura 6 – Tela principal (*home*) e tela de perfil do aplicativo “Presente!”.

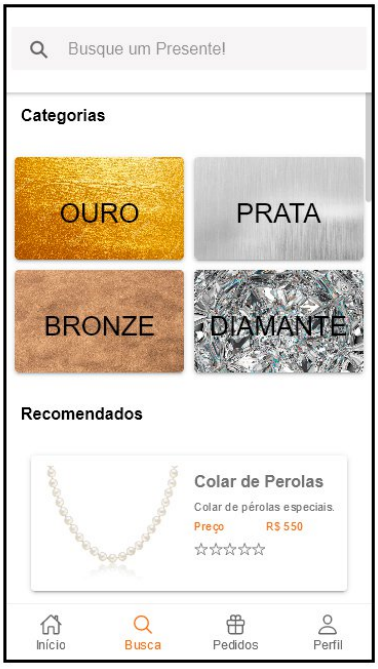


Fonte: elaborado pelos autores (2021).

2.3.4. Busca de Produtos

A tela de busca (Figura 7) possui apenas um espaço para inserção dos termos a serem buscados. O aplicativo exige que seja colocado, no mínimo, três caracteres para iniciar o processo de busca. Caso a palavra escolhida seja menor que o valor mínimo, o sistema emite um aviso informando qual a situação do erro. O limite de caracteres para busca é de 64, contando espaços em branco.

Figura 7 – Tela de busca de produtos no aplicativo “Presente!”.

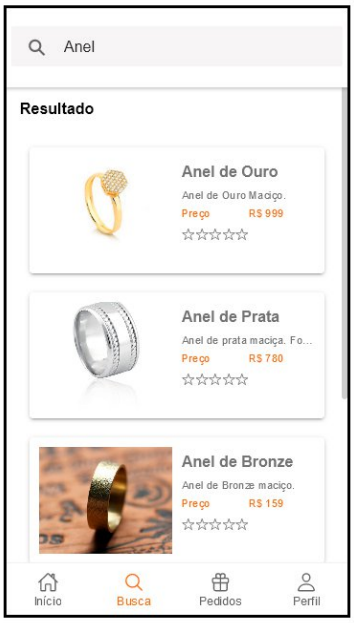


Fonte: elaborado pelos autores (2021).

2.3.5. Listagem de Produtos

Após rodar a busca, o sistema devolve, em listagem com fotos (Figura 8), os produtos que contenham os termos pesquisados, por ordem de relevância: termo contido no nome, termo contido na descrição; é possível também ordenar os resultados por preço (maior valor, menor valor).

Figura 8 – Listagem de resultados da busca no aplicativo “Presente!”.



Fonte: elaborado pelos autores (2021).

2.3.6. Produto

Ao selecionar o produto, o usuário consultar (Figura 9) mais detalhes dele (fotos, descrição, nome e avaliação do vendedor, avaliação do produto). Também é possível selecionar a quantidade de itens a serem inseridos no carrinho de compra.

Figura 9 – Detalhes de produto no aplicativo “Presente!”.

Fonte: elaborado pelos autores (2021).

2.3.7. Carrinho

O usuário pode visualizar, ao selecionar o ícone do “carrinho”, quais e quantos produtos foram selecionados, podendo diminuir ou aumentar a quantidade, bem como remover itens. Após, finaliza a seleção e segue para a concretização do pedido (Figura 10).

Figura 10 – Tela do carrinho no aplicativo “Presente!”.

Fonte: elaborado pelos autores (2021).

2.3.8. Pedido

Ao clicar em “Fechar pedido”, é solicitado que o cliente confirme os dados do remetente, bem como insira as informações para o pagamento do pedido. A seguir, é apresentada a listagem de dados do pedido (produtos, remetente, forma de pagamento) para que o cliente valide e efetive o pedido. Na Figura 11, apresentamos a tela de pedidos.

Figura 11 – Tela de pedido no aplicativo “Presente!”.

Fonte: elaborado pelos autores (2021).

3. Projeto, análise e implementação

A solução apresentada foi uma aplicação *mobile*, elaborada utilizando *frameworks* para o desenvolvimento de *front-end* e serviços de *BAAS* (*Backend as a Service*) para o do *back-end*.

Para o *front-end*, utilizamos o *framework Ionic* em conjunto com AngularJS e Capacitor (*bridge*). No *back-end*, optamos pelo *Parse Platform*. Já o banco de dados utilizado foi o *MongoDB* (*NoSQL*) e a aplicação foi hospedada na plataforma *Back4App*, que é um *MBAAS*[7] (*Mobile Backend As A Service*).

3.1. Arquitetura, módulos e subsistemas

A arquitetura do sistema é monolítica e está divida em três partes: ***front-end***, representado pelo aplicativo (*web*/*mobile*); ***back-end***, que está representado pela plataforma *Parse*; e o **banco de dados**, que está representado pelo *MongoDB*.

Importante frisar que a camada de *back-end* utiliza uma plataforma *BaaS* (*Backend as a Service*), onde toda interação, tanto com o *front-end* quanto com o armazenamento de dados, é gerida pelo serviço do *Back4App*.

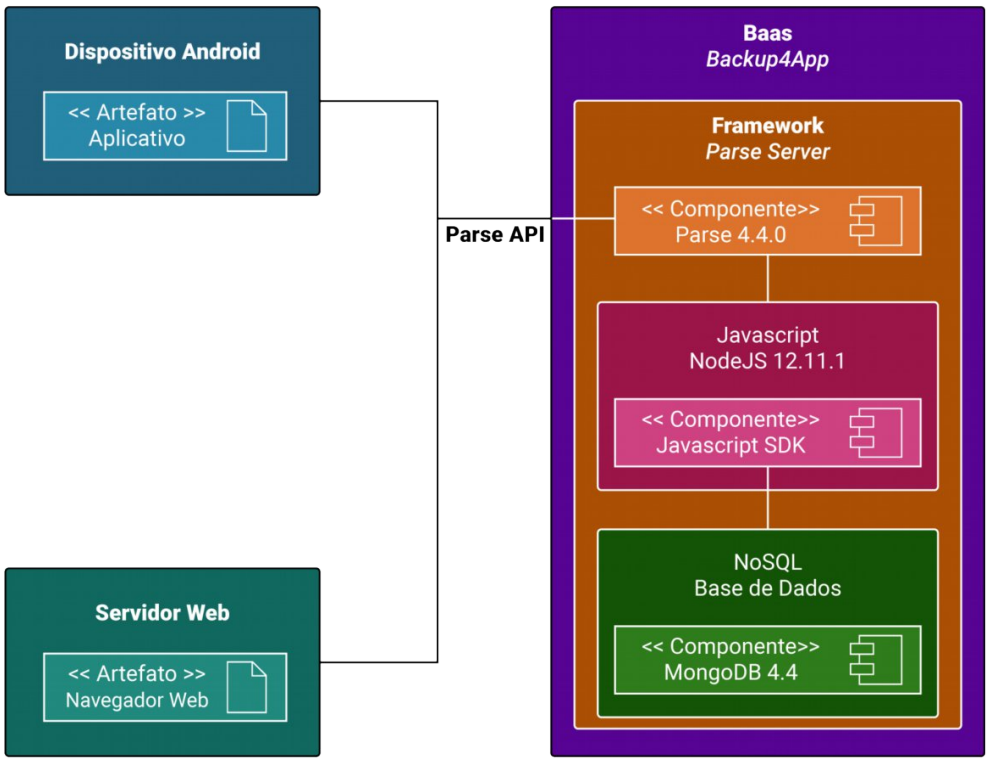
A seguir, breve descrição das tecnologias utilizadas:

- *Javascript/Typescript:* a linguagem principal utilizada tanto no *front-end* como no *back-end*;
- O *front-end:* desenvolvido com as tecnologias *web HTML*, *CSS*, *Javascript*. *Angular* e *Ionic*;
- Utilizamos o *NodeJS* como gerenciador de pacotes e base para implementação do *Angular* e do *Parse SDK*;
- O banco de dados utilizado é o *MongoDB*;
- A aplicação *back-end* está hospedada no *Back4App*;
- Utilização de bibliotecas e *frameworks Javascript*;
- O *Capacitor* é o responsável por fazer a ponte (*bridge*) entre as tecnologias *web* e os códigos nativos;
- *Parse Platform*: plataforma responsável por todos os componentes e desenvolvimento do *back-end*.

Para facilitar a compreensão da arquitetura do programa, trazemos o diagrama arquitetural na Figura 12:

7 “[...] é uma plataforma que automatiza o desenvolvimento de *back-end* e cuida da infraestrutura de nuvem. [...] terceirizará as responsabilidades inerentes a manutenção e gerenciamento de servidores para um terceiro e para focar no desenvolvimento do *front-end*.” (BACK4APP, 2021a, s/p)

Figura 12 – Diagrama arquitetural.



Fonte: elaborado pelos autores (2021).

3.1.1. Módulos

Dentro do escopo definido para o *MVP*, separamos a codificação em quatro módulos: **cadastro de usuário**, ***login* de usuário**, **busca de produtos** e **pedidos**. A seguir, listagem com exemplos dos códigos (módulo de *login* – Listagem 1; módulo de cadastro – Listagem 2 e módulo de busca – Listagem 3) utilizados no *Ionic*.

Listagem 1 – Exemplo de código responsável pelo módulo de *login* (*Ionic*).

```
72  login() {
73     (async () => {
74        try {
75           const user = await Parse.User.logIn(this.email, this.password);
76           const currentUser = Parse.User.current();
77           console.log('Logged in user', user);
78           this.router.navigate(['tabs']);
79
80        } catch (error) {
81           console.error('Error while logging in user', error);
82        }
83     }
```

Fonte: elaborado pelos autores (2021).

Listagem 2 – Exemplo do código responsável por lidar com o cadastro de usuário, bem como tratar eventuais erros (*Ionic*).

```
cadastrar() {
   (async () => {
      const user = new Parse.User();
      user.set('name', this.name);
      user.set('username', this.username);
      user.set('email', this.username);
      user.set('password', this.password);

      try {
         if (this.password !== this.confirmPassword) {
            this.password = '';
            this.confirmPassword = '';
            this.exibirAlertaSenha();
            throw new Error('As senhas não coincidem. Tente novamente');
         }

         const userResult = await user.signUp();
         Parse.User.logOut();
         this.router.navigate(['email-enviado']);
         console.log('User signed up. Please verify your e-mail');

      } catch (error) {
         console.error('Error while signing up user', error);
      }
}
```

Fonte: elaborado pelos autores (2021).

Listagem 3 – Exemplo de código responsável pelo módulo de busca (*front-end*, à esquerda e *back-end*, à direita).

```
<ion-list>
      <ion-item lines="none" *ngFor="let item of produtos | filter: busca">
         <ion-card>
            <ion-card-content class="ion-no-padding">
            <ion-grid fixed>
               <ion-row>
                  <ion-col size="6">
                     <ion-thumbnail>
                        <ion-img size="large" [src]="item.imagem">
                        </ion-img>
                     </ion-thumbnail>
                  </ion-col>
                  <ion-col size="6">
                     <ion-label>{{item.nome}}</ion-label>
                     <ion-text color="">
                        <p>{{item.descricao}}</p>
                     </ion-text>
                     <ion-grid class="preco" class="ion-no-padding" fixed>
                        <ion-row>
                           <ion-col  size="6">
                              <ion-text color="primary">
                                 <p>Preço</p>
                              </ion-text>
                           </ion-col>
                           <ion-col size="6">
                              <ion-text color="primary">
                                 <p>R$ {{item.preco}}</p>
                              </ion-text>
                           </ion-col>
                        </ion-row>
                     </ion-grid>
                  <ion-icon size="" name="star-outline"></ion-icon>
                  <ion-icon size="" name="star-outline"></ion-icon>
                  <ion-icon size="" name="star-outline"></ion-icon>
                  <ion-icon size="" name="star-outline"></ion-icon>
                  <ion-icon size="" name="star-outline"></ion-icon>
                  </ion-col>
               </ion-row>
            </ion-grid>
         </ion-card-content>
      </ion-card>
   </ion-item>
</ion-list>
```

```
data: any;
produtos: any;
busca: string;

constructor() {
   Parse.initialize(environment.APP_ID, environment.JS_KEY);
   Parse.serverURL = 'https://parseapi.back4app.com';

   const queryProdutos = (async () => {
      const produto = Parse.Object.extend('Produto');
      const query = new Parse.Query(produto);
      query.select('nome_produto', 'preco', 'descricao', 'imagem');


      try {
         const array = [];
         const results = await query.find();
         this.data = results;

         for (const object of results) {
            // Access the Parse Object attributes using the .GET method
            const nomeProduto = object.get('nome_produto');
            const precoProduto = object.get('preco');
            const descricaoProduto = object.get('descricao');
            const imagem = object.get('imagem');
            const imagemURL = imagem.url();

            array.push({
               imagem: imagemURL,
               nome: nomeProduto,
               descricao: descricaoProduto,
               preco: precoProduto,
            });

            this.produtos = array;
         }

      } catch (error) {
         console.error('Error while fetching Produto', error);
      }
   })();
}
```

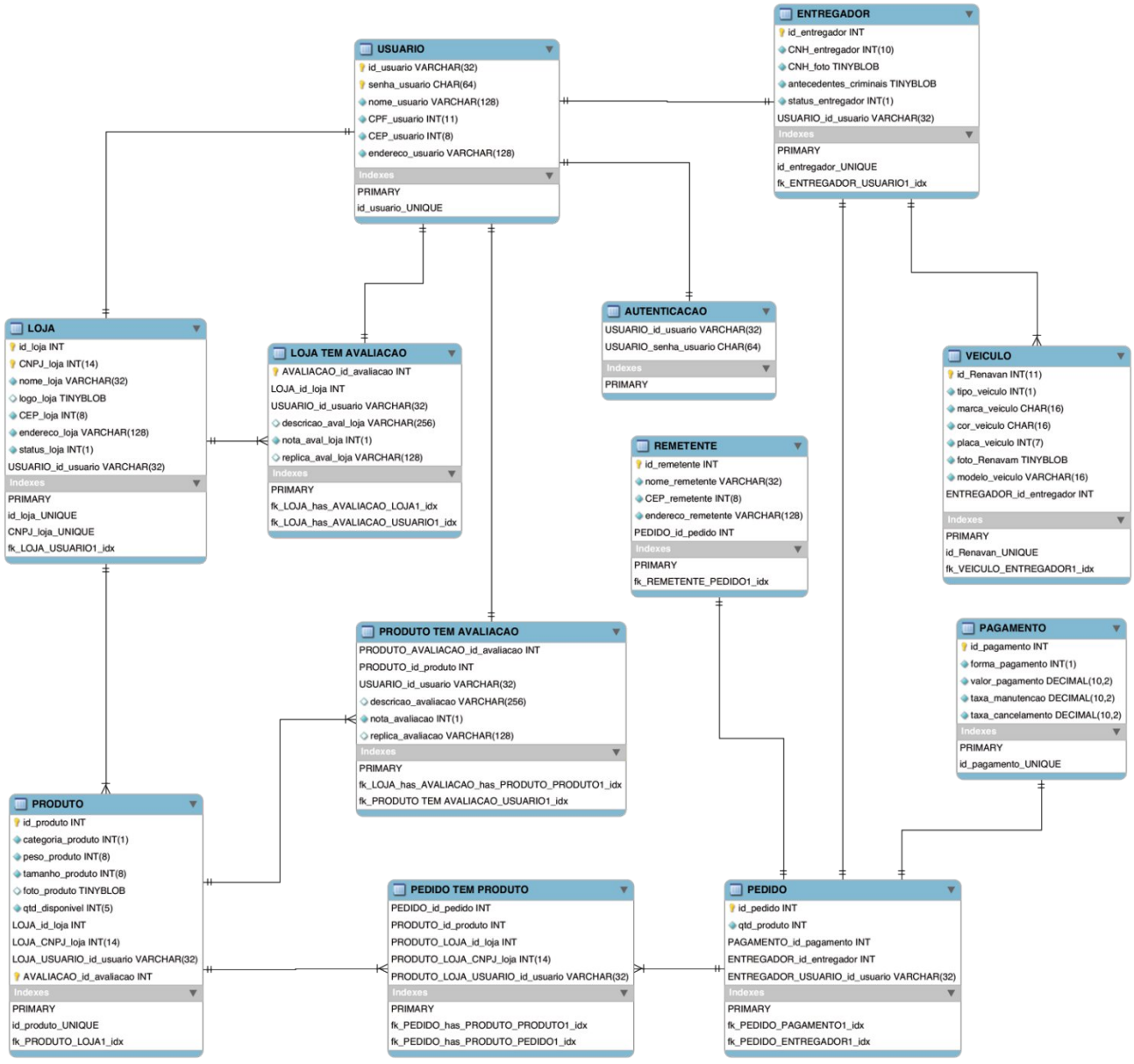
Fonte: elaborado pelos autores (2021).

3.2. Projeto de banco de dados

Ainda que o banco de dados selecionado seja o *MongoDB*, consideramos a estrutura do *Back4app*[8], que é híbrida, de forma que apresentamos (Figura 13) o esquema de entidade-relacionamento do programa a seguir.

Observamos que, neste caso, estamos trazendo o aplicativo por completo e não apenas o escopo do *MVP*.

Figura 13 – EER (esquema entidade-relacionamento) “Presente!”.



Fonte: elaborado pelos autores (2021).

[8] *Back4app* é uma das soluções de *Backend as a Service* que “[...] dá suporte na utilização de uma planilha como solução de banco de dados oferecendo uma maneira mais fácil para os desenvolvedores criarem, atualizarem e sincronizar seus dados de aplicativos [...] permitindo que seus usuários importem ou exportem seu JSON.” (BACK4APP, 2021b, s/p).

3.3. Tecnologias utilizadas

Para a concepção do *MVP* do aplicativo “Presente!”, utilizamos como base, para o *front-end*, o *framework Ionic* (baseado em Angular), que é *open source* e serve para desenvolvimento de aplicativos móveis multiplataforma (*web* e *mobile*).

Para o *back-end*, utilizamos o *Parse* juntamente com o *Back4App*, soluções que permitem o desenvolvimento de aplicativos de modo mais rápido, uma vez que viabilizam a utilização de soluções pré-prontas para o desenvolvimento de *back-end*.

No banco de dados, optamos pela utilização do *MongoDB*, um banco *NoSQL* (não-relacional e orientado a documentos), mas que, utilizado em conjunto com o *Back4App*, permite criar relacionamentos através de ponteiros (caracterizando-se, portanto, como um banco de dados híbrido), além de trabalhar com documentos esquematizados similares ao *JSON*[9].

O *layout* do aplicativo foi desenvolvido no *Figma*, ferramenta que permite criar protótipos de tela de aplicativos para verificar legibilidade e usabilidade.

O projeto foi hospedado no *GitHub*, para facilitar o fluxo de trabalho da equipe e pode ser acessado pelo link: https://github.com/devliciousofficial/presente

A seguir, listamos as tecnologias citadas (Quadro 3), bem como suas utilizações dentro do escopo definido para o MVP.

Quadro 3 – Tecnologias utilizadas.

Tecnologia	**Descrição**	**Utilização**
ANGULAR	“[...] é um *framework JavaScript* que simplifica não apenas a construção da interface de usuário, mas também o desenvolvimento de aplicações *client-side* diferenciadas, sejam elas para a *web*, *mobile* ou *desktop*.” (DEVMEDIA, 2021a, s/p)	Desenvolvimento de *front-end*.
BACK4APP	O *Back4App* é um *MBaaS* flexível, escalável e fácil de usar, baseado na plataforma *Parse*. É uma plataforma totalmente gerenciada, com provisionamento e dimensionamento automatizados do aplicativo *Parse Server*. Oferece migração de aplicativos, ferramentas de gerenciamento baseadas na *Web*, *backup* e recuperação, monitoramento e alerta 24/7, e suporte especializado. O *Back4App* permite que os desenvolvedores personalizem e otimizem cada aplicativo separadamente, para máxima flexibilidade. (BACK4APP, 2021b, s/p)	Hospedagem e gerenciamento de banco de dados.

[9] “[...] é um arquivo que contém uma série de dados estruturados em formato texto e é utilizado para transferir informações entre sistemas [...] é uma notação para a transferência de dados que segue um padrão específico [...] utilizada em diferentes linguagens de programação e sistemas.” Entretanto, esses dados precisam estar estruturados “[...] por meio de uma coleção de pares com nome e valor ou ser uma lista ordenada de valores. Seus elementos devem conter: chave [...] e valor [...]” (SOUZA, 2020, s/p).

Tecnologia	Descrição	Utilização
CAPACITOR	*Capacitor* é um *runtime* nativo de código aberto que facilita a construção de aplicativos *web* modernos que são executados nativamente no iOS, Android e na *web* (*PWA*). Representando a próxima evolução dos aplicativos híbridos, o *Capacitor* usa uma abordagem de contêiner nativo moderna para equipes que desejam desenvolver aplicativos utilizando tecnologias *web* (*JavaScript*, *HTML* e *CSS*) sem sacrificar o acesso total aos *SDKs* nativos quando necessário.	Integração entre código *web* e as plataformas nativas.
IONIC	"[...] é um *framework open source* para desenvolvimento de aplicativos móveis multiplataforma. Para isso, possibilita a implementação do *app* utilizando tecnologias comumente empregadas na construção do *front-end* de soluções [...]" (DEVMEDIA, 2021b, s/p)	Desenvolvimento de componentes *front-end*.
MONGODB	"[...] *software* de banco de dados orientado a documentos livre, de código aberto e multiplataforma [...] usa documentos semelhantes a *JSON* com esquemas [...] dados podem ser aninhados em hierarquias complexas e continuar a ser indexáveis e fáceis de buscar. (WIKIPEDIA, 2021, s/p). No BACK4APP, há um recurso de ponteiros que pode ser utilizado para criar relacionamentos entre os dados.	Banco de dados.
PARSE	"O *Parse* é uma estrutura de código aberto amplamente usada para o desenvolvimento de *back-ends* de aplicativos. A estrutura ajuda os desenvolvedores a acelerar o desenvolvimento de aplicativos em uma extensão considerável." (BACK4APP, 2021, s/p) Além disso, reduz o esforço necessário para desenvolver um aplicativo e "[...] permite que os usuários aproveitem o desenvolvimento de seus aplicativos sem complicações. Os desenvolvedores não precisam contar com a ajuda de engenheiros de *back-end* para criar aplicativos de alta qualidade." (BACK4APP, 2021c, s/p)	Desenvolvimento de *back-end*.

Fonte: elaborado pelos autores (2021).

4. Considerações finais

As melhores ideias surgem das propostas mais simples. A equipe *Devlicious*, durante a pandemia, conseguiu identificar um problema que parecia ser comum aos familiares e amigos e sugeriu, como visto neste relatório, uma solução – que não adjetivaremos de "simples", pois seria amenizar as dificuldades que tivemos ao longo do caminho – que entende ser capaz de resolver a questão levantada no início deste trabalho.

Ao definirmos o escopo do *MVP*, a estratégia para realização do mesmo se tornou mais realista. Utilizar, de fato, o "dividir para conquistar" foi a melhor tática para que pudéssemos definir e ir produzir um aplicativo cuja entrega fosse possível, levando em conta o curto prazo para sua realização e a concomitância com as outras atividades acadêmicas, pessoais e profissionais.

Durante o percurso da realização deste projeto, foi interessante não apenas revisar conceitos vistos desde os primeiros dias de aula, mas ir atrás de tecnologias novas, conseguindo trabalhar com essas por conta da excelente base que tivemos durante nossa graduação – deixamos nossos agradecimentos e profundo respeito a todos os professores que passaram por nós, com seus intensos conhecimentos técnicos e sua infinita paciência pedagógica.

Assim, entregamos o que consideramos ser o melhor possível dentro das margens que nos foram dadas, satisfeitos e imensamente contentes de conseguir atingir a nossa proposta.

Para aqueles que quiserem verificar o projeto em sua totalidade, este pode ser acessado pelo link: https://github.com/devliciousofficial/presente.

Para acessar a nossa apresentação: https://bit.ly/31HXDLZ.

Referências

BACK4APP. **O que é Backend as a Service?** Disponível em https://bit.ly/3nO1n7x Acesso em 17.09.2021a.

_____. **What is Back4app?** Disponível em https://bit.ly/3tRskZ4 Acesso em 17.09.2021b.

_____. **What is Parse?** Disponível em https://bit.ly/3zo8eqr Acesso em 17.09.2021c.

DEVMEDIA. **Guia Completo de Ionic:** Criando aplicativos Mobile Multiplataforma. Disponível em https://bit.ly/3nMOpa4 Acesso em 17.09.2021b.

_____. **Guia Completo de Angular:** Primeiros Passos no Angular. Disponível em https://bit.ly/3CoIFaH Acesso em 17.09.2021a.

ESALQ/USP. Diagrama de Ishikawa. **Projetos de Gestão pela Qualidade da ESALQ** – Ferramentas. S/d. Disponível em https://bit.ly/3CJ65I6 Acesso em 15.08.2021.

EY PARTHENON. Consumo e Pandemia: As mudanças de hábitos e padrões de comportamento provocados pelo coronavírus. **VEJA INSIGHTS**. 2020. Disponível em https://bit.ly/3CMjKij Acesso em 14.08.2021.

INOVASOCIAL. **O isolamento social transformou o consumo dos brasileiros?** 2020. Disponível em https://bit.ly/3m5reai Acesso em 14.08.2021.

ITAÚ. **Análise do Comportamento de Consumo 2020.** 2021. Disponível em https://bit.ly/3m1j3LV Acesso em 14.08.2021.

MELO, D. O que é SQL? **Tecnoblog**. 2021. Disponível em https://bit.ly/3tXH60f Acesso em 17.09.2021.

ROCHA, R. Itaú mapeia drivers do consumo em 2021. **Meio&Mensagem.** 2021. Disponível em https://bit.ly/3fZbqC3 Acesso em 14.08.2021.

SEBRAE. **Conheça novos padrões de consumo e tendências do mercado pós-pandemia.** 2020. Disponível em https://bit.ly/2VUjfC9 Acesso em 14.08.2021.

SOUZA, I. Afinal, o que é JSON e para que ele serve? **Rock Content**. 2020. Disponível em https://bit.ly/3u6QpLH Acesso em 17.09.2021.

TAYLOR-BRANDS. Os 9 tipos de logotipos e como os usar. **Tailor Brands.** S/d. Disponível em https://bit.ly/3o2uVP1 Acesso em 15.08.2021.

THE ECONOMIC TIMES. What is Minimum Viable Product? **India Times.** Disponível em https://bit.ly/2Z1tsOk Acesso em 14.10.2021.

WIKIPEDIA. **MongoDB**. Disponível em https://bit.ly/3lC0Hzv Acesso em 17.09.2021.

Agradecimentos

Agradecemos aos nossos familiares, que foram o nosso suporte para a conclusão de mais uma etapa em nossa vida. Aos nossos companheiros e amigos que aguentaram todo esse "*papo de nerd de TI*" pelos últimos anos: vocês foram incríveis. E ao nosso grupo – *Devlicious* – por todo o "*corre*" e pelas boas risadas!

Agradecemos os colegas de curso, que se tornaram grandes amigos durante o período da pandemia, sendo o apoio – uns dos outros – para chegarmos juntos ao final. Kouki, Ogyan, Fabião e Lucas Campos, estamos juntos!

Aos professores, em especial (mas não exclusivamente!) ao Rafael Will, José Pacheco, Marco Jeunon e Leonardo Takuno: a presença de vocês iluminou nossos caminhos e deixou as aulas muito melhores! Obrigado, obrigado, obrigado!

Ao Professor Leonardo Augusto Taniguti Mantovani, que nos deixou cedo demais. Não esquecemos, em nenhum momento, dos seus ensinamentos: seguimos sempre procurando o que o cliente precisa, não necessariamente o que ele quer. Obrigado por tudo.

Até mais, e obrigado pelos peixes!