Revista do *Fórum Gerencial*, v. 1, n. 2, 2021
© Centro Universitário de Patos de Minas
https://revistas.unipam.edu.br/index.php/forumgerencial

APLICATIVO DE DELIVERY PARA EMPRESAS DO RAMO DA ALIMENTAÇÃO[1]

Kevem Pedro Pereira Mesquita Martins Lima[2]
Juliana Lilis da Silva[3]

RESUMO: O presente estudo trata de uma aplicação *web* e outra *mobile* desenvolvidas para a empresa de *delivery* Me Petisque situada em Patos de Minas. A empresa utiliza o aplicativo WhatsApp e o *marketplace* iFood para atender aos seus clientes, o que dificulta o atendimento e o aumento de lucros. Nesse sentido, o objetivo do desenvolvimento das aplicações *web* e *mobile* foi entregar à empresa Me Petisque uma plataforma que permitisse um melhor atendimento aos clientes, aumentando o número de atendimentos e, consequentemente, aumentando os lucros da empresa. No desenvolvimento foi utilizado o framework NestJS na parte do *backend* da aplicação, a biblioteca React Native na aplicação *mobile* e a biblioteca ReactJS na aplicação *web*. A linguagem utilizada foi o JavaScript e o banco de dados escolhido para armazenamento dos dados foi o *PostgreSQL*. Para facilitar e agilizar o desenvolvimento, a metodologia utilizada seguiu as premissas do *Scrum*, sendo utilizados alguns artefatos e eventos do mesmo. Como resultado, a empresa de *delivery* em questão tem em mãos uma ferramenta que pode impulsionar seu negócio.
PALAVRAS-CHAVE: *Delivery*; Aplicações Híbridas; JavaScript.

ABSTRACT: The present study it's a *web* and a *mobile* application developed to the *delivery* company Me Petisque located in Patos de Minas. The company use the WhatsApp application and the iFood marketplace to serve its customers, which makes it difficult to service and increase profits. In this sense, the objective of developing *web* and *mobile* applications was to provide the company Me Petisque a platform that allows better service to customers, increasing the number of services and, consequently, increasing the company's profits. In development was used the framework, NestJS in the *backend* part of the application, library React Native in the *mobile* application and library ReactJS in the *web* application, the language used was JavaScript and the database chosen for data storage was *PostgreSQL*. To facilitate and speed up development, the methodology used followed the premises of Scrum, using some artifacts and events from it. As a result, the *delivery* company in question has a tool in hand that can boost your business.
KEYWORDS: *Delivery*; Hybrid Applications; JavaScript.

1 INTRODUÇÃO

Em decorrência da mudança no estilo de vida da população nas últimas décadas, ocorreu um aumento significativo da alimentação fora do domicílio. Alguns

[1] Artigo apresentado como avaliação parcial da disciplina de Estágio Supervisionado II, para obtenção de título de bacharel em Sistemas de Informação, pelo Centro Universitário de Patos de Minas – UNIPAM.

[2] Estudante de Graduação do Curso de Sistemas de Informação do UNIPAM. E-mail: kevempedro@unipam.edu.br.

[3] Orientadora, Mestre em Ciência da Computação pela UFU e professora do UNIPAM. E-mail: juliana@unipam.edu.br.

fatores são responsáveis, como serviços fornecidos pelos *fast foods*, *self services* e *deliveries* (LEAL, 2010).

No mercado de *delivery* é de suma importância que os donos de estabelecimentos apostem na tecnologia para aumentar suas receitas e fidelizar seus clientes. Como pode ser observado, o mercado americano absorveu uma grande quantidade de downloads de aplicativos próprios dos estabelecimentos e ultrapassou em muito a dos *marketplaces*. No futuro, é provável que todo estabelecimento que optar por *delivery* tenha seu próprio aplicativo para se aproximar mais do seu cliente e fidelizá-lo melhor (OLD, 2017).

Ao fazer uma busca no Google pelo tema "mercado de *delivery* no Brasil", a maioria das notícias são sobre o crescimento do setor nos últimos anos. De acordo com dados do Sebrae, em 2017 o *delivery* movimentou mais de R$9 bilhões e cresce na casa dos 12% ao ano. A projeção do mercado de *delivery* no mundo também é bastante positiva. O *delivery* online, por exemplo, movimentou US$ 82,2 milhões em 2018 e a taxa média de crescimento para o setor é de 9,3% ao ano, resultando em um volume de US$ 134,5 milhões até 2023 (FRANÇA, 2019).

No cenário brasileiro, atualmente, existem grandes *marketplaces* atuando no mercado de *delivery* online, como Ubereats, Rappi e um dos principais, o iFood, que lidera o mercado brasileiro. Percebe-se que este mercado vem ganhando força ano a ano e revolucionando o mercado de alimentos em todo mundo. Porém, *marketplaces* como iFood, Ubereats e Rappi nos últimos tempos vem desagradando os donos de estabelecimentos que fazem uso dessas plataformas.

As principais reclamações das empresas que trabalham com *delivery* e utilizam dos *marketplaces* atuais é a falta de uma plataforma que permita uma melhor fidelização dos clientes, pois neste segmento geralmente não existe a identidade visual da empresa. Portanto, as dificuldades que as empresas que estão no mercado de *delivery* online encontram são: enviar notificações personalizadas na hora que elas quiserem para clientes específicos, lançar promoções de forma imediata e sem perda de tempo, baixa personalização da identidade digital e o alto custo cobrado pelos grandes *marketplaces*. Este conjunto de fatos impulsiona as empresas a criarem suas próprias plataformas, com o objetivo de atender suas necessidades e garantir um melhor atendimento para seus clientes.

Observando esse cenário, este artigo tem como objetivo apresentar o desenvolvimento de um aplicativo para as plataformas Android e IOS, junto com um painel *web*, para a empresa Me Petisque localizada em Patos de Minas. Esta empresa atualmente atende seus clientes pelo WhatsApp e iFood. No aplicativo proposto, o cliente do Me Petisque pode fazer seu pedido, ganhar cupons descontos e escolher uma forma de pagamento. Já pelo painel, o Me Petisque pode gerenciar seus produtos no aplicativo, receber e acompanhar os pedidos e visualizar gráficos que os ajudem nas tomadas de decisões.

Para que o objetivo geral fosse atingido, foram atendidos os seguintes objetivos específicos:

- *Cadastrar Clientes:* o aplicativo realiza o cadastro do cliente, cadastro esse que é utilizado para a realização dos pedidos.
- *Autenticar*: tanto o aplicativo quanto o painel realizam a autenticação.

- *Cadastrar Cupons:* Pelo painel é possível cadastrar cupons de desconto para serem utilizados pelos clientes em seus pedidos.
- *Visualizar Gráficos*: através do painel é possível visualizar alguns gráficos que ajudam o Me Petisque a tomar decisões na hora de lançar uma promoção ou desconto.

O sistema tem por objetivo também proporcionar o crescimento na disputa pelo mercado de *delivery* online para empresas como o Me Petisque, que buscam cada dia mais sua sobrevivência nesse ramo e não podem ficar à mercê de grandes *marketplaces*, os quais podem levar as mesmas à falência (MADUREIRA, 2020) e/ou diminuir seus lucros. Além dos fatos apresentados, não oferecerem uma plataforma que atenda às necessidades como lançar notificações de uma promoção ou desconto em um momento de necessidade da empresa em tempo oportuno e sem oneração, enviar um cupom de desconto para o cliente aniversariante do mês ou que seja um cliente consolidado e fazer um marketing direcionado para um grupo de cliente específico.

Garantir a fidelização dos clientes é tão importante quanto conquistar novos clientes. Tanto a fidelização quanto a busca por novos clientes são fundamentais para que a mesma se torne referência e continue atuando no mercado por muito tempo (MORAES, 2018), para isso a empresa precisa apostar na tecnologia e investir em ferramentas que proporcionem uma melhor tomada de decisão.

Todavia, a criação do aplicativo e do painel proporciona para o Me Petisque uma melhor gerência de seu negócio. O aplicativo permite aos clientes da empresa uma maneira mais fácil e ágil de realizar seus pedidos e o painel entrega a empresa uma melhor forma de gerenciar seu negócio, possibilitando assim que a mesma se mantenha no mercado de *delivery* online.

2 REVISÃO DE LITERATURA

Nessa seção são apresentados conceitos referentes à Aplicações Híbridas, Gerenciador de estado Redux e Sistemas de Informações Gerenciais - SIG.

2.1 APLICAÇÕES HÍBRIDAS

Com a demanda cada vez maior de usuários por dispositivos móveis, surgiram novas empresas. Cada uma com sua proposta de aparelho e com isso surgem também novas plataformas e sistemas operacionais, o que impacta no desenvolvimento de aplicações, para atender a todos os dispositivos, pois é necessário ter conhecimentos específicos de cada plataforma, para que seja desenvolvida uma aplicação diferente para cada uma, implicando também em custos e tempo mais elevados para o desenvolvimento (PREZOTTO, 2014).

Neste cenário surgem as aplicações híbridas, onde se desenvolve apenas uma aplicação, e esta pode ser utilizada em vários dispositivos com diferentes sistemas operacionais. Isso é possível devido aos frameworks direcionados para o desenvolvimento de aplicações híbridas, que empacotam o código-fonte para as diferentes plataformas, permitindo que elas possam ser instaladas no dispositivo e sejam capazes de acessar diferentes recursos, como contatos, câmera e GPS.

Aplicações híbridas possuem como finalidade funcionar em qualquer dispositivo, em que independentemente da plataforma, será utilizado o mesmo código-fonte. As mesmas possuem acesso a todos os recursos do dispositivo, o que permite aos desenvolvedores criarem aplicações com os mesmos recursos disponíveis para a criação de aplicações nativas. Diferente das aplicações nativas, que utilizam das linguagens específicas de cada plataforma, como no Android em que se faz uso do *Java* ou *Kotlin*, e no IOS *Objective-C* ou *Swift*, as aplicações híbridas utilizam do *Hypertext Markup Language* - HTML, *Cascading Style Sheets* - CSS e JavaScript, independentemente da plataforma.

Aplicações desenvolvidas nos frameworks que oferecem um desenvolvimento híbrido são instaladas nos dispositivos e podem ser disponibilizadas juntamente com as aplicações nativas nas lojas de aplicativos, no caso de Android (Play Store) e para o IOS (Apple Store). Apesar das aplicações híbridas conseguirem entregar bons resultados, um dos principais motivos para sua escolha é o baixo custo de tempo de desenvolvimento, já que o código-fonte é o mesmo, impactando inclusive na manutenção da aplicação.

Atualmente existem diversos frameworks híbridos, como o Ionic, React Native e Flutter por exemplo, que são alguns dos principais frameworks do mercado (SILVA, 2018). Cada um oferece recursos, tanto compartilhados quanto específicos, a escolha do framework para o desenvolvimento da aplicação varia de acordo com os requisitos de cada projeto ou preferência da equipe de desenvolvimento.

2.2 GERENCIADOR DE ESTADO REDUX

Quando se fala em desenvolvimento utilizando bibliotecas como ReactJS ou React Native, é imprescindível não utilizar componentes para facilitar a reutilização e manutenção de código. Contudo ao se utilizar componentes em uma aplicação, independente se a biblioteca escolhida for ReactJS ou React Native, é preciso se ter a preocupação de como esses componentes vão compartilhar os diferentes estados da aplicação. Para facilitar o compartilhamento desses estados entre os componentes pode-se utilizar o Redux.

Redux é um contêiner de estado para aplicações JavaScript, o estado é mantido em uma (*store*) loja e os componentes que estão interessados nos dados da loja ficam observando-a e caso haja alguma alteração nesses dados o componente reage de acordo com a mudança. O Redux implementa o padrão Flux que gerencia o fluxo de dados da aplicação (WECK, 2017).

Esse fluxo é dividido em três etapas: *store*, *reducers* e *actions*. A *store* é onde ficam armazenadas as informações e possui disponibilidade para receber e entregar para o componente o que ele requisita. Quando algum componente precisa alterar algum dado o *reducer* é encarregado de lidar com isso. Já as *actions* são responsáveis por requerer ao *reducer* uma ação de acordo com o tipo da mesma.

Com isso se houver uma alteração no valor de um produto do sistema por exemplo, é disparado uma *action* para o *reducer* que através do tipo dessa *action*, nesse caso é uma alteração no produto, altera o valor do produto na *store* e a mesma é responsável por enviar esse dado alterado para as *views* ou componentes que observam

essa *store*. Portanto, com a utilização do Redux torna-se muito mais fácil o compartilhamento dos diferentes estados da aplicação entre os diferentes componentes da mesma.

2.3 SISTEMAS DE INFORMAÇÕES GERENCIAIS - SIG

Existem diferentes definições sobre a expressão sistemas de informações gerenciais dentre as várias, pode-se definir a mesma como um método organizado que fornece informações passadas, presentes e futuras, relacionadas com as operações internas e serviços de inteligência externa. Dão suporte ao planejamento, controle e operação de uma empresa, a partir do fornecimento de informações, no tempo apropriado, para auxiliar o tomador de decisão (OLIVEIRA, 2012).

Sistemas de Informações Gerenciais para auxiliar o gerenciamento das atividades de um negócio é algo de extrema importância nos dias de hoje, bem como para auxiliar no planejamento estratégico do mesmo. O sistema de informação, combinado com a pesquisa de mercado, fornece informações valiosas para que a empresa tome decisões importantes e precisas diante do mercado em que se encontra, atuando em todos os níveis de uma organização: estratégico, tático e operacional. É de uma grande importância que os sistemas de informação forneçam informações precisas para uma adequada tomada de decisão.

Antigamente sistemas de informação apoiados por computadores eram opções caras e acessíveis para poucos, além de terem limitações graves e serem muito simples. Atualmente, a realidade é outra, pois com o avanço tecnológico e com o surgimento de equipamentos mais poderosos os sistemas agora são complexos e conseguem fornecer informações precisas para o apoio às decisões gerenciais.

A Informação nos dias atuais está cada vez mais disponível para todos e em grande abundância, podendo ser acessada de forma instantânea e ser consumida a qualquer hora. Portanto, é cada vez maior a importância de que, gestores ou responsáveis por uma empresa tenha em mãos ferramentas que os auxiliem em uma tomada de decisão gerencial mais precisa e correta, proporcionando uma vantagem estratégica em relação aos seus concorrentes (DOS SANTOS CLARO, 2013).

Para se desenvolver um sistema que seja eficaz é preciso diferenciar conceitos como: dados, informação e conhecimento. Esses conceitos muitas vezes são confundidos ou tratados como a mesma coisa, entretanto, cada um tem o seu significado e importância. Dados são registros soltos, ainda não tratados e sem sentido, representa a matéria-prima da informação, ou seja, a informação sem tratamento que ainda não tem relevância, com isso o dado por si só não é capaz de transmitir nenhum conhecimento. Informação representa a estrutura ou organização dos dados, ela é um registro derivado dos mesmos e também é insumo para o conhecimento. Já o conhecimento é a informação processada, que pode ser utilizada para criar, inovar, progredir, ou seja, o conhecimento é uma ferramenta de grande valor que permite pessoas evoluírem, assim como as empresas (REZENDE, 2015).

Portanto, um software, aplicativo ou afins, apesar de serem importantes para automatizar os serviços e processos de uma empresa, se os mesmos não forem capazes de gerar dados com qualidade, para serem processados e gerar informações que

possam ser transformadas em conhecimento para que a empresa possa evoluir com seus negócios, então talvez esse software ou aplicativo não tenha tanta relevância para a mesma.

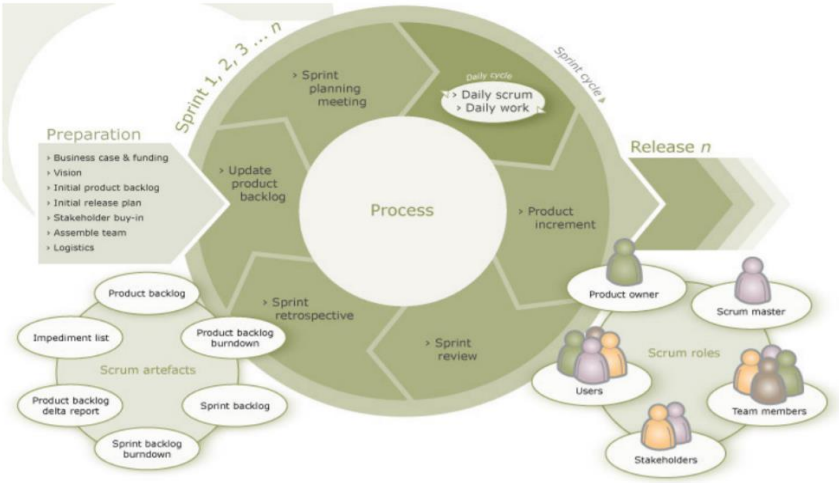
3 METODOLOGIA

A metodologia utilizada no desenvolvimento do artigo segue as premissas do Scrum. O Scrum é um framework para desenvolver e manter produtos. Desenvolvido por Ken Schwaber e Jeff Sutherland, o Scrum não é um processo ou uma técnica para construir produtos, mas sim um framework dentro do qual se emprega vários processos ou técnicas (SCHWABER E SUTHERLAND, 2013).

O framework Scrum consiste nos times do Scrum, cada um associado aos seus papéis, artefatos, eventos e regras. Cada um desses componentes dentro do framework serve a um propósito, sendo fundamentais para o sucesso do Scrum. Na Figura 1 pode-se observar que o Scrum é constituído de vários artefatos e eventos que facilitam o andamento de um dado processo ou projeto. Para o desenvolvimento desse projeto foram consideradas *sprints* de 4 semanas. Dos artefatos e eventos que o Scrum oferece, foram utilizados:

- *Product Backlog:* consiste em uma lista ordenada das funcionalidades necessárias no projeto, normalmente organizada pelo valor que cada funcionalidade entrega.
- *Sprint Backlog:* um conjunto de algumas das funcionalidades do *Product Backlog* que são selecionadas para serem desenvolvidas durante uma *Sprint*.
- *Sprint:* é um time-boxe de duração de um mês ou menos, no qual os itens da *Sprint Backlog* são desenvolvidos, gerando uma versão incremental do produto.
- *Increment:* é onde todos os itens do *Product Backlog* desenvolvidos durante a *Sprint* serão somados aos itens das outras *Sprints*, incrementando mais valor ao já desenvolvido.

Figura 1: Fluxo Scrum



Fonte: BISSI, 2007.

O Quadro 1 apresenta as sprints definidas e executadas durante o desenvolvimento do projeto.

Quadro 1: Estruturação do desenvolvimento do projeto seguindo o framework Scrum

Sprints	Atividades
Sprint 1	· Desenvolver a estrutura das aplicações *backend, frontend* e *mobile*. · Desenvolver o CRUD de usuários e administradores. · Autenticar no painel. · Cadastrar usuários pelo aplicativo.
Sprint 2	· CRUD de produtos. · Apresentar produtos no painel e aplicativo.
Sprint 3	· Autenticar no aplicativo. · Realizar os pedidos pelo aplicativo. · Gerenciar pedidos pelo painel.
Sprint 4	· Desenvolver o serviço de cupons do sistema. · Desenvolver a tela de informações gerenciais no painel.
Sprint 5	· Escrever o artigo final. · Estruturar apresentação do artigo.

Fonte: Dados do trabalho, 2020.

Para a implementação do sistema foram utilizadas as subsequentes ferramentas:

- *Visual Studio Code:* trata-se de um editor de código leve, multiplataforma, gratuito e *open source*. Mantido pela Microsoft, essa ferramenta conta com suporte a várias linguagens, extensões, integração com *Git*, *debug*, terminal integrado, entre outros recursos.
- *NestJS:* é um framework *Node.js* progressivo para criar aplicativos eficientes, confiáveis e escalonáveis do lado do servidor.
- *ReactJS:* Biblioteca JavaScript para criar interfaces de usuário.
- *React Native:* é uma biblioteca JavaScript criada pelo Facebook. É utilizada para desenvolver aplicativos para os sistemas Android e IOS de forma nativa.
- *PostgreSQL*: sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language) para a persistência de dados.
- *Git*: sistema de controle de versão que gerencia arquivos e diretórios e as modificações feitas neles ao longo do tempo.

O desenvolvimento do sistema foi dividido em três aplicações, sendo elas: *backend*, *frontend* e *mobile*. Durante o desenvolvimento do mesmo foi utilizado o fluxo de trabalho do Gitflow. O Gitflow é um design de fluxo de trabalho Git, que define um modelo de ramificação projetado com base no lançamento do projeto, oferecendo uma estrutura robusta para gerenciar projetos.

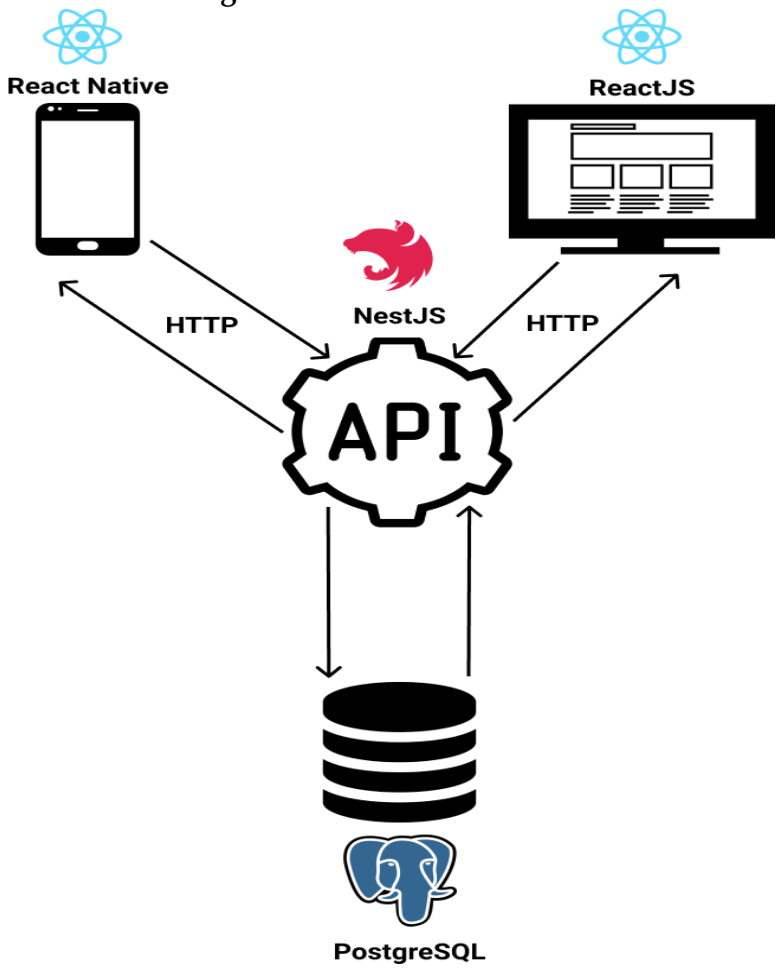
4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Na fase inicial do projeto foi definido a linguagem de programação, banco de dados, framework e bibliotecas. A linguagem utilizada no desenvolvimento foi o

JavaScript, o framework escolhido e as bibliotecas utilizadas para o desenvolvimento do *backend* da aplicação, painel e aplicativo foram respectivamente: NestJS, ReactJS e React Native. Já para o armazenamento dos dados foi utilizado o banco de dados *PostegreSQL*.

Na Figura 2 é apresentada a estrutura do sistema que consiste em um padrão *Model-View-Controller* - MVC, mostrando as tecnologias que foram citadas anteriormente e ilustrando como funciona o fluxo do sistema.

Figura 2: Estrutura do sistema



Fonte: Dados do trabalho, 2020.

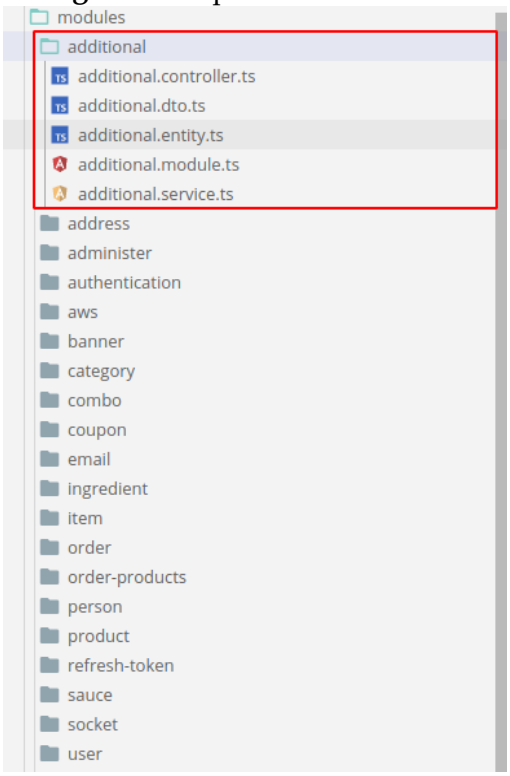
Falando um pouco mais sobre o backend, o mesmo foi desenvolvido em NestJS como já foi dito. O NestJS é um framework Node.js progressivo para criar aplicativos eficientes, confiáveis e escalonáveis do lado do servidor. Permite trabalhar com *Object-relational mapping* - ORM, utiliza do *TypeScript* o que torna o desenvolvimento mais seguro e menos propenso a erros. O NestJS tem sua arquitetura inspirada no *Angular*, possuindo um baixo acoplamento e uma fácil manutenção.

A arquitetura da aplicação *backend* está dividida de forma que cada funcionalidade está separada, deixando a mesma coesa, menos acoplada e de fácil manutenção. Cada funcionalidade da aplicação possui seu módulo específico, nesse módulo encontra-se a *entity*, *controller*, *dto*, *service* e *module* da funcionalidade

E*ntity* é o modelo da entidade que é utilizado para gerar as tabelas no banco de dados, a *controller* é responsável por fazer a comunicação entre as requisições e as regras de negócio da aplicação, ou seja, ela é uma fronteira. O *dto* é uma classe responsável por validar os atributos vindos na requisição antes de chegar na *controller*. A *service* possui toda a regra de negócio, validações e é responsável por persistir os dados no banco de dados caso necessário. E a *module* é onde é configurado e exportado o módulo para todo o projeto caso uma outra funcionalidade precise do mesmo.

Na Figura 3 pode-se observar o que foi explicado anteriormente, utilizando como exemplo o módulo de adicionais.

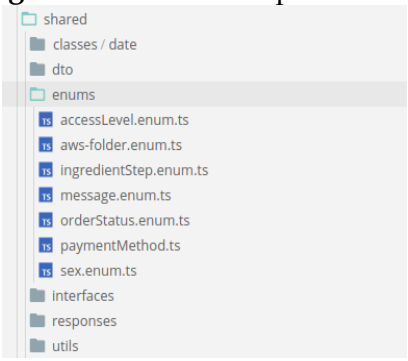
Figura 3: Arquitetura do *backend*



Fonte: Dados do trabalho, 2020.

A aplicação *backend* também conta com uma pasta *shared*, que contém todas as classes, interfaces e funções que são comuns para mais de um módulo da aplicação, centralizando tudo o que é compartilhado em apenas um lugar, permitindo uma melhor reusabilidade e manutenção. Na figura 4 é possível observar melhor a utilização da pasta *shared*.

Figura 4: Estrutura da pasta *shared*



Fonte: Dados do trabalho, 2020.

O painel do sistema foi desenvolvido utilizando o ReactJS, uma biblioteca JavaScript de código aberto usada para construir interfaces de usuário eficientes e que permite a reusabilidade de componentes que tenham sido desenvolvidos em outras aplicações. Esses componentes são fáceis de escrever por usarem JSX, uma extensão de sintaxe opcional para JavaScript, a qual permite que se combine Hypertext Markup Language - HTML, com o JavaScript. Na Figura 4 pode-se observar um exemplo de uma sintaxe JSX.

Figura 5: Sintaxe JSX

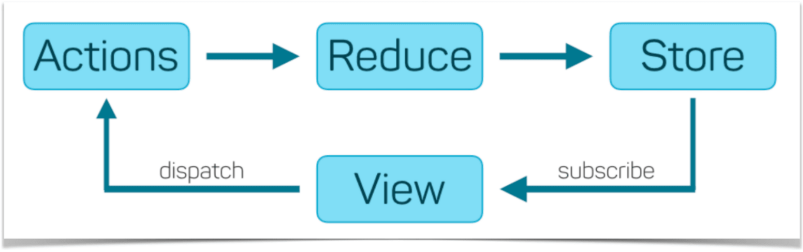
```
1
2   const element = <h1>Hello, world!</h1>;
3
```

Fonte: Dados do trabalho, 2020.

O conceito da arquitetura utilizada no painel é a mesma do *backend*, com apenas algumas diferenças para adaptar-se à aplicação que agora é *web*, mas ainda procurando manter a coesão, a fácil manutenção e uma boa reusabilidade. Para facilitar o compartilhamento de estados entre vários componentes, foi utilizado o Redux tanto no painel quanto no aplicativo.

Teoricamente o Redux é um controlador de estados geral para uma aplicação. Compartilhar estados entre os diferentes componentes torna- se muito mais fácil quando se utiliza o Redux, o qual é basicamente divido em 3 partes: *store*, *reducers* e *actions*, como pode ser observado na Figura 6.

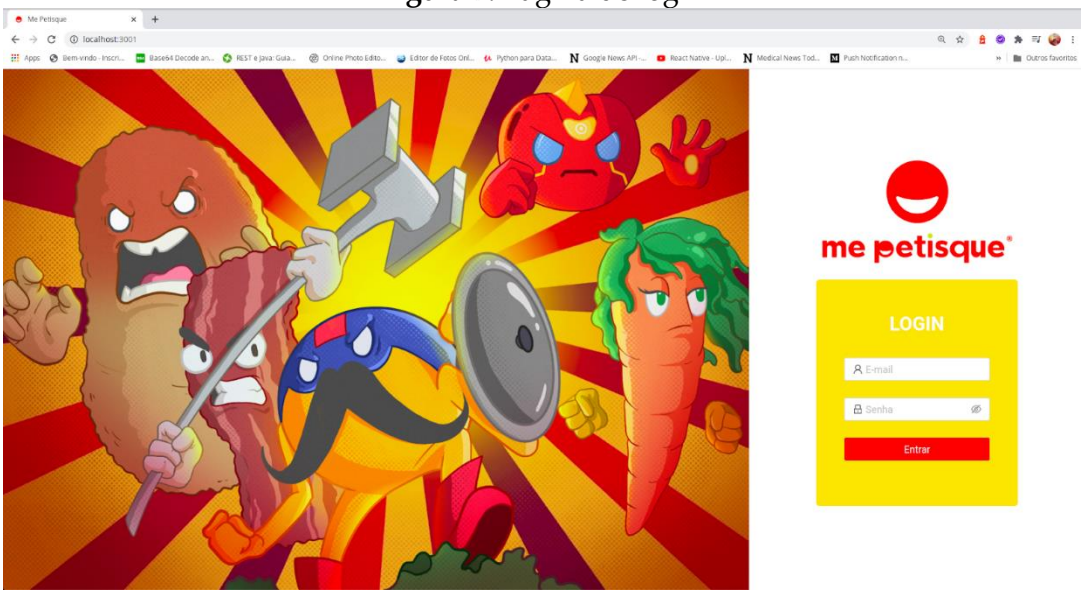
Figura 6: Fluxo do Redux



Fonte: WECK, 2017.

A seguir é apresentado, na Figura 7, a página de login do painel um dos resultados obtidos durante o desenvolvimento da aplicação *web*.

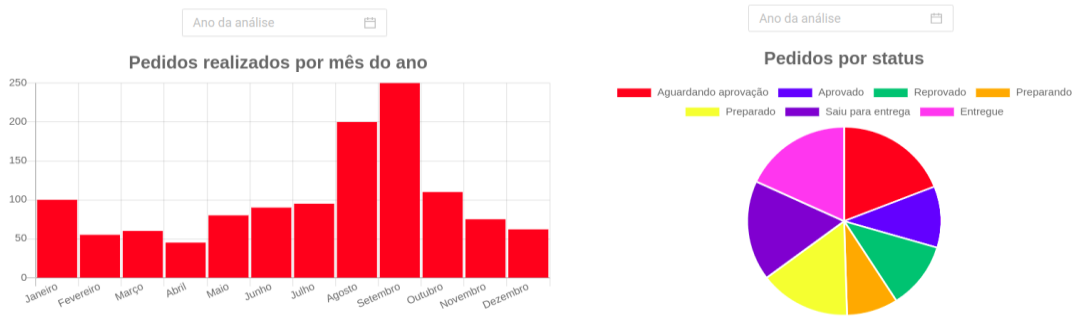
Figura 7: Página de login



Fonte: Dados do trabalho, 2020.

Outro resultado obtido durante o desenvolvimento da aplicação *web*, foi a parte de análise gráfica do sistema, como pode-se observar na Figura 8, tem-se dois gráficos, no primeiro é mostrado uma relação da quantidade de pedidos realizados por cada mês do ano e o segundo traz informações da quantidade de pedidos que se encontra em cada status proposto, tanto o primeiro quanto o segundo gráfico pode ter suas informações filtradas pelo ano, possibilitando a análise de dados também de anos anteriores ou posteriores.

Figura 8: Página de análise



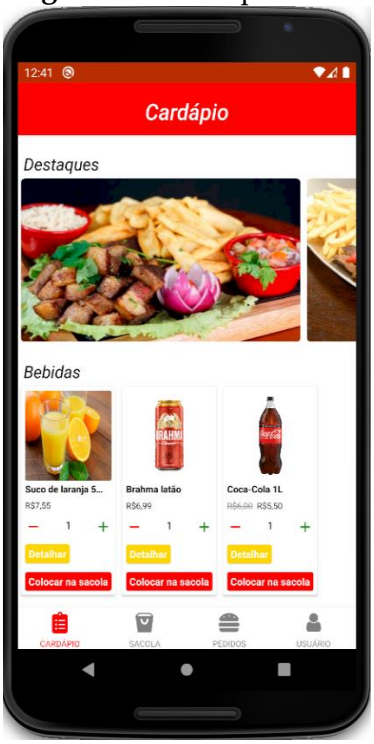
Fonte: Dados do trabalho, 2020.

Para finalizar a parte do aplicativo, este foi desenvolvido utilizando a biblioteca React Native, sendo essa uma biblioteca JavaScript criada pelo Facebook e utilizada para desenvolver aplicativos para os sistemas Android e IOS de forma nativa. Com o React Native é possível gerar aplicativos tanto para a plataforma Android

quanto IOS utilizando apenas uma linguagem de programação, agilizando muito o desenvolvimento *mobile*.

Praticamente não existe diferença entre as bibliotecas ReactJS e React Native, o que agiliza mais ainda o desenvolvimento de um projeto que possui uma versão *web* e *mobile*. Com isso, a arquitetura, estrutura e forma de programação tanto do painel quanto do aplicativo não possuem diferenças. Na Figura 9 é apresentada a tela de produtos do aplicativo, nela é possível observar como os produtos estão dispostos na tela. Já na figura 10 é apresentado a tela de detalhes do produto do aplicativo, nela é possível visualizar de forma mais detalhada as informações do produto.

Figura 9: Tela de produtos



Fonte: Dados do trabalho, 2020.

Figura 10: Detalhes do produto

Fonte: Dados do trabalho, 2020.

De acordo com a proposta do projeto e os objetivos propostos, foi apresentado nesta seção alguns dos resultados obtidos durante o desenvolvimento do sistema, mostrando algumas páginas, telas, padrões de projeto e arquiteturas.

5 CONSIDERAÇÕES FINAIS

O painel desenvolvido tem como proposta entregar para o Me Petisque uma aplicação *web* por meio da qual pode-se gerenciar os produtos que serão vendidos no aplicativo, visualizar os clientes cadastrados pelo aplicativo, cadastrar cupons de descontos para serem disponibilizados para os clientes, gerenciar os pedidos do dia e também visualizar algumas análises. Com isso a empresa tem em mãos um painel para poder administrar seu negócio.

Pelo aplicativo desenvolvido o cliente do Me Petisque pode se cadastrar, efetuar login, visualizar os produtos disponíveis para venda, realizar seus pedidos e se necessário fazer a atualização dos seus dados cadastrados no aplicativo. Entregando para o cliente uma aplicação *mobile* que permite a utilização dos serviços prestados pela empresa.

Mesmo com todos os objetivos propostos para este trabalho terem sido alcançados, salienta-se que o sistema desenvolvido não passou por uma etapa de testes, para garantir que todas as suas funcionalidades estão funcionando como o planejado e o mesmo também não foi colocado em produção. Além disso, para projetos futuros podem ser desenvolvidas novas funcionalidades, como *push notifications* para a empresa poder enviar notificações de promoções ou algum cupom de desconto para seus clientes, ou ainda alguma forma de autenticação diferente como, pela conta Google ou Facebook.

Com a mudança no estilo de vida da população, a mesma está cada vez mais optando por fazer suas refeições fora do domicílio ou pedir sua comida no conforto de sua casa, com isso o mercado de *delivery* vem ganhando força a cada ano. Portanto, é imprescindível que as empresas desse ramo apostem em tecnologia para conseguirem se manter na disputa do mercado de *delivery*. Neste sentido, o aplicativo desenvolvido para a empresa Me Petisque pode ser adaptado e atender a diversas outras empresas do ramo de alimentação que trabalha com *delivery*.

REFERÊNCIAS

BISSI, Wilson. Metodologia de desenvolvimento ágil. **Campo Digital**, v. 2, n. 1, 2007. Disponível em: https://bit.ly/2SMTjnm. Acesso em: 5 maio. 2020.

DOS SANTOS CLARO, José Alberto Carvalho. **Sistemas de Informações Gerenciais,** São Paulo. 2013. Disponível em: https://bit.ly/2yd0NIE. Acesso em: abr. 2020.

FRANÇA, Ivanir. **Investimentos**: tendências para o mercado de *delivery* no Brasil e no mundo em 2019. 2019. Disponível em: https://bit.ly/33TSopd. Acesso em: 8 fev. 2020.

LEAL, Daniele. Crescimento da alimentação fora do domicílio. **Segurança Alimentar e Nutricional**, v. 17, n. 1, p. 123-132, 2010. Disponível em: https://bit.ly/2Ueqm4U. Acesso em: 18 março 2020.

MADUREIRA, Daniele. **Como apps de entrega estão levando pequenos restaurantes à falência**, 2020. Disponível em: https://bit.ly/2UotevY. Acesso em: 18 março 2020.

MORAES, Daniel. **A arte de fidelização do cliente**: entenda o que você precisa para ter um consumidor que propague sua marca, 2018. Disponível em: https://bit.ly/3ao0JUG. Acesso em: 16 março 2020.

OLIVEIRA, Djalma. **Sistemas de Informações Gerenciais**. 15. ed. São Paulo: Atlas, 2012.

OLD, Dino. **Startup brasileira que faz apps de delivery online vai para o Vale do Silício pelo Google**, 2017. Disponível em: https://bit.ly/2UotevY. Acesso em: 23 fev. 2020.

PREZOTTO, Ezequiel Douglas; BONIATI, Bruno Batista. **Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas**. Trabalho de Conclusão de Curso. Universidade Federal de Santa Maria. 2014. Disponível em: https://bit.ly/2Jq0g8y. Acesso em: mar. 2020.

REZENDE, Eliana. **Dados, informação e conhecimento**: o que são?. 2015. Disponível em: https://bit.ly/2WZ5U9A. Acesso em: 27 março 2020.

SILVA, Mateus. **Quais são os principais frameworks para desenvolver aplicações móveis híbridas?**. 2018. Disponível em: https://bit.ly/2yiyAjP. Acesso em: 27 março 2020.

SCHWABER, Ken; SUTHERLAND, Jeff. **Um guia definitivo para o Scru**m: as regras do jogo. Processo de Desenvolvimento de Software, 2013. Disponível em: https://bit.ly/2yurxoo. Acesso em: 21 fev. 2020.

WECK, Sandy. **Developing modern offline apps with ReactJS, Redux and Electron** - Part 3 ReactJS + Redux. 2017. Disponível em: https://bit.ly/346KlY1. Acesso em: 18 out. 2020.