CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO HENRIQUE DE LIMA SILVA

DESENVOLVIMENTO DE UM APLICATIVO HÍBRIDO COM QUASAR E STRAPI

Botucatu-SP
Dezembro – 2019

DESENVOLVIMENTO DE UM APLICATIVO HÍBRIDO COM QUASAR E STRAPI

DEVELOPMENT OF A HYBRID APPLICATION WITH QUASAR AND STRAPI

Pedro Henrique de Lima Silva[1], Gustavo Kimura Montanha[2]

[1] Graduando Tecnólogo em Analise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia de Botucatu, Av. José Ítalo Bacchi, s/n – Jd. Aeroporto, CEP 18606-851, Botucatu – São Paulo. E-Mail: pedro97henrique@gmail.com

[2] Professor Doutor do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Av. José Ítalo Bacchi,, s/n – Jd. Aeroporto, CEP 18606-851, Botucatu – São Paulo. E-Mail: gmontanha@fatecbt.edu.br

RESUMO

O desenvolvimento de aplicativos mobile tende a ser muito trabalhoso já que atualmente se tem duas plataformas totalmente distintas com linguagens de programação diferentes, o que causa principalmente muitos gastos com relação a custos pois é necessário ter ao menos duas equipes para cuidar de cada aplicativo, o que pode ser um impedimento para projetos pequenos. A Partir desta problemática alguns desenvolvedores criaram ferramentas para o desenvolvimento híbrido. Neste artigo será apresentado uma das ferramentas utilizadas para este fim, esta ferramenta utiliza todo o ambiente de desenvolvimento do Node.js que é uma plataforma de aplicação que utiliza de uma máquina virtual chamada V8 para interpretar os códigos gerados. Ainda existem algumas limitações com relação a esse tipo de desenvolvimento como a renderização de elementos 3D como em jogos, também há uma dificuldade em processar grandes quantidades de dados, mas como o aplicativo apresentado neste artigo não possui num destes impedimentos foi possível utilizar somente tecnologias híbridas neste projeto. Concluindo que aplicativos híbridos podem ser utilizados para fins comerciais e que nos teste o usuário final não sentirá diferenças entre um aplicativo híbrido para o nativo.

Palavras-chave: Android, Aplicações, Híbrido, Quasar, Strapi.

ABSTRACT

Mobile application development tends to be a lot of work as we currently have two totally different platforms with different programming languages, which mainly causes a lot of cost expense because you need to have at least two teams to take care of each application, which means can be a deterrent to small projects. From this issue some developers have created tools for hybrid development. In this article we will introduce one of the tools used for this purpose, this tool uses the entire development environment of Node.js which is an application platform that uses a virtual machine called V8 to interpret the generated codes. There are still some limitations regarding this type of development such as rendering 3D elements as in games, there is also a difficulty processing large amounts of data, but as the application presented in this article does not have in one of these impediments it was possible to use only hybrid technologies In this project. Concluding that hybrid applications can be used for business purposes and that testing us the end user will not feel any differences between a hybrid application for the native.

Key Words: Android, Applications, Hybrid, Quasar, Strapi.

1 INTRODUÇÃO

Com o aumento no uso de dispositivos móveis, cada vez mais aplicativos estão sendo desenvolvidos para atender às mais diversas necessidades dos usuários (IDC, 2017). A criação de um aplicativo, por muitas vezes, tende a ser muito dispendioso, devido ao custo de se manter um profissional que seja capaz de desenvolver uma aplicação para diversas plataformas do mercado atual. Com base nisso, programadores começaram a desenvolver aplicativos que tenham somente uma base de código, ou seja, aplicável para diversas plataformas (WARGO, 2015).

Surgem dessa forma, frameworks que são um conjunto de códigos que seguem um padrão definido, dedicados para a solução de um determinado problema (GOVONI, 1999), responsáveis pela criação de aplicativos híbridos, que podem ser utilizados em múltiplas plataformas, sem a necessidade do desenvolvimento de mais de um código fonte. Até então, era necessário o desenvolvimento de um aplicativo para o sistema operacional Android e outro para o sistema operacional IOS; a partir do uso desses *frameworks*, pode-se utilizar o mesmo código fonte para ambos os sistemas operacionais.

Grande parte da comunidade de programadores envolvidos nesses projetos de desenvolvimento se comunicam através de redes sociais, onde o GitHub surge como uma das ferramentas mais utilizadas, onde é possível seguir usuários e ter acessos a seus projetos de softwares públicos ou compartilhados; também abrir *issues* nos projetos, solicitar correções ou modificações de alguma funcionalidade; além de, permitir qualquer pessoa fazer uso dos projetos, modificá-los e enviá-los para aprovação no projeto principal, se desejar. (BLISCHAK, John D.; DAVENPORT, Emily R.; WILSON, Greg, 2016) Essa comunidade tem crescido cada vez mais, o que tem sido um grande motivador para desenvolvimento de aplicativos *mobiles*, *desktops* ou *web* (YU et al, 2014).

A maioria desses frameworks são desenvolvidos de maneira *open source* (código aberto), permitindo que outros profissionais tenham acesso ao código fonte, para que possam melhorá-lo ou, ainda, criar novos *frameworks* baseados no código fonte disponibilizado. Mas nem todo sistema *open source* é um sistema de código livre. Para que um *software* seja considerado livre, é necessário que disponha de quatro liberdades: A liberdade de executar o sistema; a liberdade de estudar através do sistema; a liberdade de redistribuir cópias, de modo que você possa ajudar ao seu próximo; e a liberdade de aperfeiçoar o sistema e de liberar os

seus aperfeiçoamentos, de modo que toda a comunidade se beneficie deles (STALLMAN, 2010).

O objetivo deste trabalho foi desenvolver um aplicativo híbrido para dispositivos móveis para o gerenciamento de consumo em um estabelecimento comercial.

2 MATERIAL E MÉTODOS

2.1 Material

Para o desenvolvimento do *software*, foi necessário um ambiente preparado para a criação e testes do seu código. Assim, cada linguagem de programação ou *framework* tiveram seus requerimentos básicos respeitados para que a linguagem e/ou o *framework* fossem bem aproveitados no projeto. Para o desenvolvimento do aplicativo de gerenciamento de doses, foi utilizada a linguagem de programação JavaScript e para o correto funcionamento dessa linguagem, foi necessário a utilização de computador com configurações de *hardware* compatíveis.

A linguagem interpretada JavaScript necessitou de uma máquina virtual ou *engine* para interpretar o código. Neste caso, o computador utilizado para a criação desta aplicação utilizou as *engines* V8*;* padrão no pacote Node.js.

Para instalar as dependências que vêm por padrão na maioria dos projetos desenvolvidos em JavaScript, utilizou-se os gerenciadores de pacotes NPM e Yarn, que serviram para instalar os pacotes e as bibliotecas no projeto.

O *software* necessitou de um compilador que gerou o .apk, extensão utilizada nos dispositivos Android. Para isso, foi instalado o Java JDK (*J*ava *Development Kit*), o JRE (*Java Runtime Environment*), além do Android Studio. Essas ferramentas foram utilizadas para o desenvolvimento do aplicativo para dispositivo Android.

Para que o código criado em HTML CSS e JavaScript pudesse ser executado nos dispositivos, foi utilizado um *wrapper* que serviu como uma "casca" para o aplicativo, uma vez que o sistema operacional não tem capacidade de executar os códigos em HTML, CSS e JavaScript, o mesmo foi responsável por criar um *browser* para que o aplicativo pudesse fazer as interações com os usuários, bem como, as interações internas com o sistema operacional.(LOPES, 2016) Neste caso, foi utilizado o Cordova que é um *wrapper*, que tem os métodos necessários para transportar o código em HTML5, CSS3 e JavaScript para a

webview que é um navegador nativo que tem a responsabilidade de renderizar o visual do aplicativo gerado no aplicativo (WARGO, 2017).

Para o *back-end*, foi instalado o Strapi que é uma dependência global do NPM (ou Yarn), ou seja, ele não é instalado no projeto do *back-end*, mas sim, na máquina em que o projeto está sendo criado. Esta dependência foi responsável por criar a infraestrutura e os métodos que gravam os dados no banco, bem como as consultas dos mesmos dados pela aplicação; de modo geral, a dependência foi responsável pelo CRUD - operações de *Create* (criação e armazenamento de dados), *Read* (leitura de dados), *Update* (atualização de dados) e *Delete* (deleção de dados) - do sistema. O Strapi pode ser hospedado na maioria dos servidores disponíveis no mercado - neste projeto foi utilizado o Heroku, pois oferece um servidor gratuito para a distribuição de diversas aplicações.

Para o armazenamento dos dados gerados na aplicação, foi utilizado o MariaDB, que é um banco de dados *open source*, mas também pode ser utilizado outros bancos relacionais como MySQL e Oracle, ou bancos não relacionais como o MongoDB (BANKER, 2011).

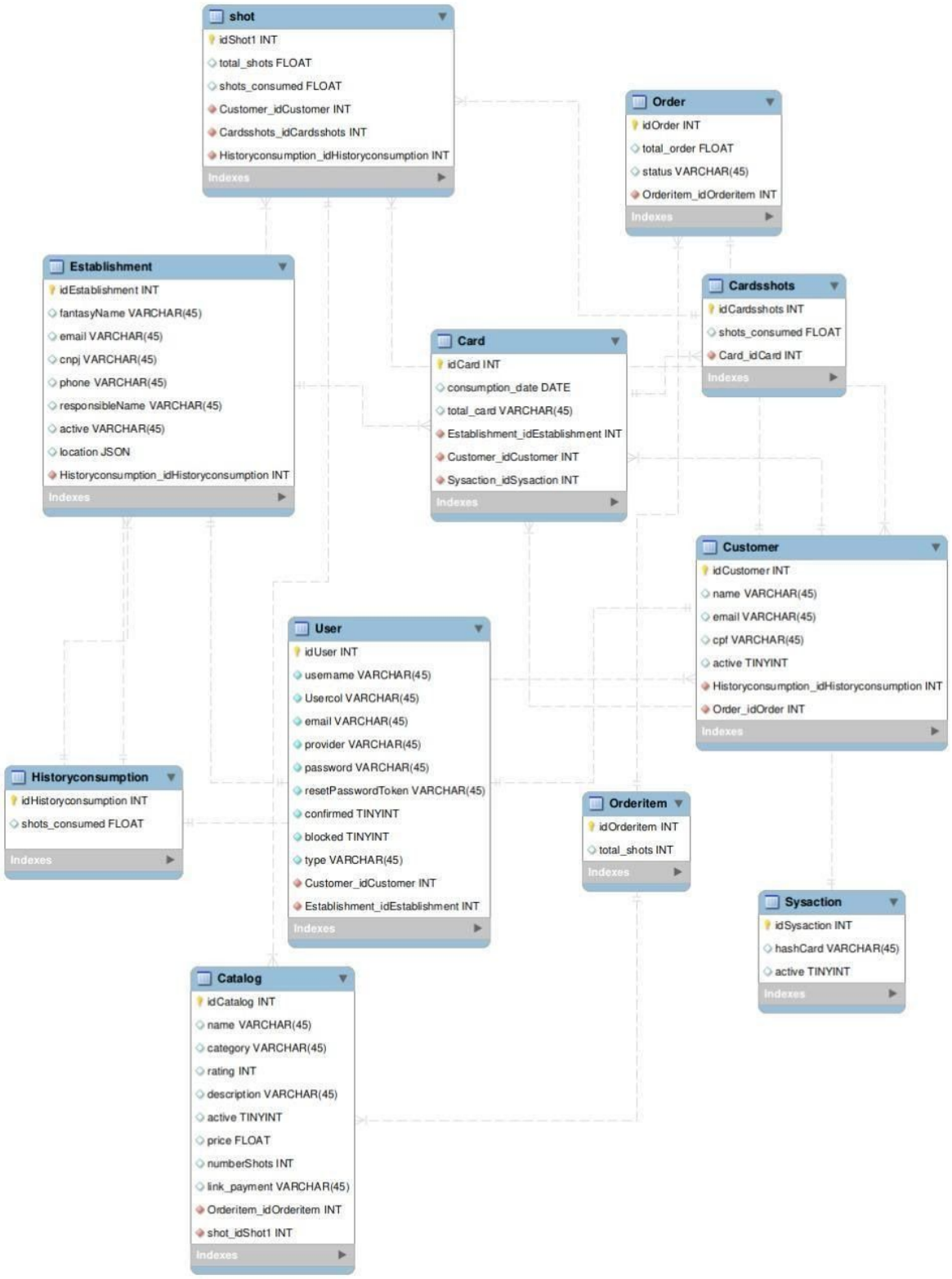
Por fim, para facilitar a criação da UI (*user interface*) do aplicativo, foi necessário utilizar um navegador *web.* Os mais recomendados para a criação desta aplicação foram o Google Chrome e o Firefox, que possuem recursos visuais para o desenvolvimento *web*.

2.2 Métodos

Na criação deste aplicativo foi utilizado conceitos de engenharia de software para a criação da modelagem do banco de dados bem como para a criação dos códigos responsáveis pela geração de informações.

Para a modelagem do banco de dados foi utilizado o modelo o DER (Diagrama Entidade Relacionamento) que contém os dados que cada tabela irá armazenar. A Figura 1 ilustra as tabelas criadas para o armazenamento dos dados gerados na aplicação.

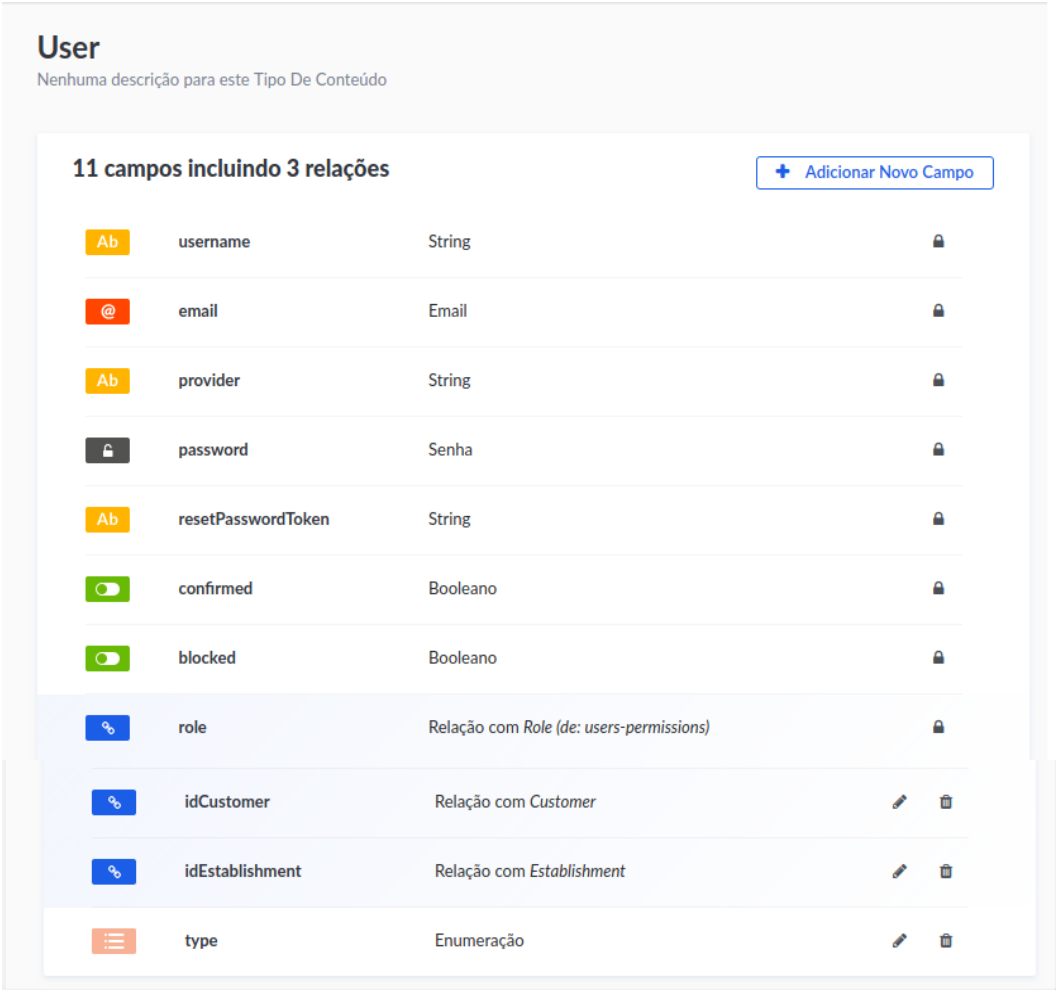
Figura 1 - Diagrama entidade relacionamento



A partir dos modelos criados foi possível o desenvolvimento do *back-end* que utilizou o *framework* Strapi para sua construção permitindo o uso de uma série de ferramentas já

prontas para agilizar o processo de desenvolvimento. Uma das ferramentas utilizadas foi o *dashboard* que facilitou a criação das tabelas e seus relacionamentos, o Strapi que oferece recursos como autenticação de usuários, servidor para *upload* de arquivos, serviço de e-mail, dentre outros (Figura 2).

Figura 2 Tabela usuários criadas pelo Strapi



Após a criação das demais tabelas no Strapi, foi possível disponibilizar este serviço para que o aplicativo pudesse interagir com os dados. Para disponibilizar o *back-end* foi utilizado o Heroku que é um serviço de hospedagem que oferece uma infraestrutura de servidores para hospedar o serviço do *back-end* no Heroku. Nele, existem 4 tipos de contas que variam de contas *free* para contas *performance* diferindo-se fundamentalmente nas funcionalidades disponíveis. Na conta *free* não é necessário nenhum pagamento, mas o

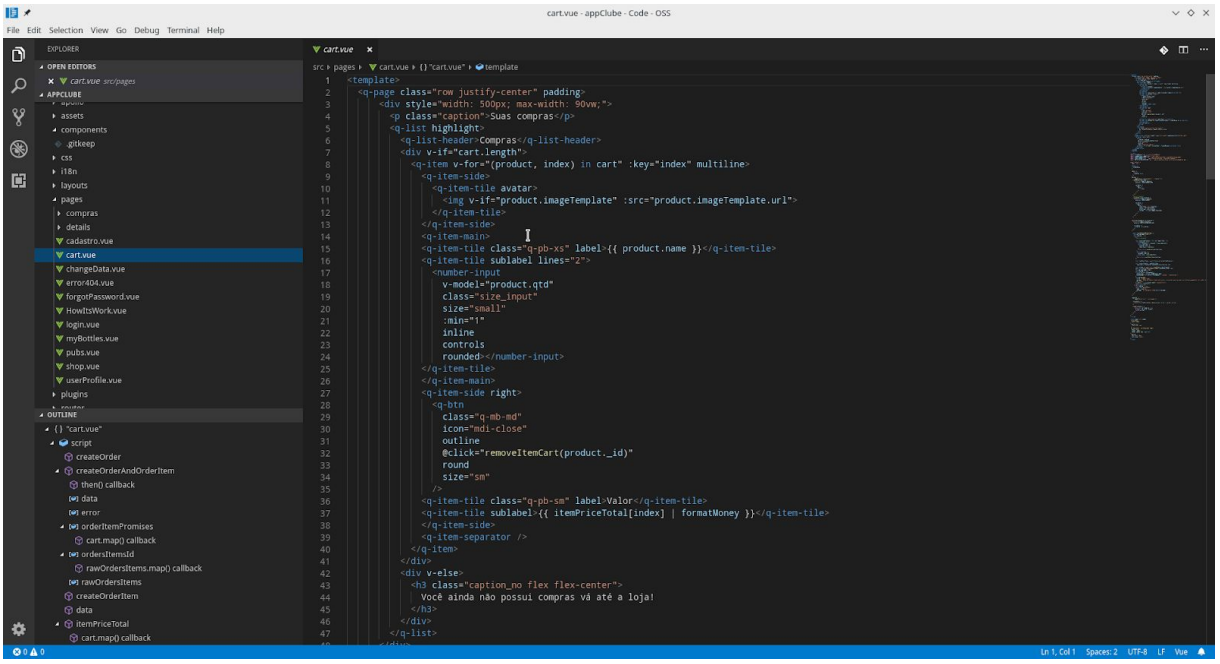
servidor utilizado para hospedar a aplicação pode entrar em modo de hibernação se não houver nenhum tipo de interação com o mesmo dentro de 30 minutos e também há uma restrição na quantidade de servidores que se pode ter na conta, já a conta performance apresenta um custo que varia entre 25 a 500 dólares mensais, e uma das principais vantagens é que esses servidores não hibernam e não limitam a quantidade de servidores.

Para hospedar o serviço de *back-end* da aplicação no heroku foi necessário criar uma conta, nesse projeto, uma conta *free.* Após a criação do usuário na plataforma, foi criado um novo servidor para hospedar o *back-end*, que dentre as diferentes formas de disponibilizar o código do *back-end* para o servidor recém criado, optou-se pela plataforma do Github que permitiu que toda vez que um novo código fosse inserido no seu repositório, também fosse disponibilizado para o servidor do heroku.

Para que o usuário pudesse interagir com os dados, foi necessário a construção de um *front-end* ou seja uma interface para comunicação também chamada de *UI* (interface de usuário). Essa interface foi responsável por garantir que o usuário tivesse maior controle sobre o aplicativo e pudesse manuseá-lo de forma fluida e intuitiva. Para esta fase do projeto foi utilizado conceitos de *UX* (experiência do usuário) que estuda a interação do usuário com o produto, sendo esta interação a mais intuitiva possível para permitir analisar as informações que devem ser mostradas e qual a melhor maneira de exibi-las para se criar de uma estrutura onde o usuário consiga, de forma fácil, interagir com o aplicativo. Salienta-se que isso não impede que erros de usabilidade aconteçam, devendo a construção do produto ser contínua e com *feedbacks* dos clientes (GARRETT, 2010). Como ferramenta para a criação desta interface foi utilizado o *framework* do Quasar que conta com diversos *layouts* padronizados, como botões, entradas de textos, formulários para inserção de dados, listas dentre outros. O Quasar utiliza da linguagem HTML, CSS e JavaScript para a criação destes componentes visuais e tem toda a sua documentação disponibilizada no site.

Para a criação e edição dos códigos do aplicativo foi utilizado o *VsCode* que é um editor de texto que conta com muitas ferramentas que auxiliam no desenvolvimento como: análise sintática, realce de sintaxe, *debugger*… etc Além destas ferramentas conta com uma infinidade de *plugins* para as mais variadas necessidades. Como mostra a figura 3 este é o código responsável por criar os elementos visuais do carrinho de compras do aplicativo que foi codificado com o VsCode.

Figura 3 - *VsCode*



Após a criação das demais funcionalidades do aplicativo foi possível gerar o *apk* para teste nos dispositivos móveis. Para isso foi necessário executar o seguinte comando no terminal *quasar build -m cordova -T android* para que o cordova possa gerar o *apk*.

Importante salientar que para o desenvolvimento do aplicativo em sistemas IOS deve se ter o ambiente de desenvolvimento baseado no MacOs com *x-code*, contudo, com os mesmos comandos para gerar o instalador do aplicativo.

3 RESULTADOS E DISCUSSÃO

Uma das grandes dúvidas antes de se iniciar o desenvolvimento de uma aplicação híbrida é saber o quanto ela será performática para os seus usuários já que está performance garante para o cliente uma interface fluida e livre de travamentos que podem causar desconforto para seus utilizadores.

Aplicações híbridas possuem certas limitações com relação a performance principalmente quando se trata de renderização de modelos em 3D como em jogos, outra limitação é com relação a serviços em segundo plano que podem ser integrados com o aplicativo, mas podem causar consumo excessivo da bateria do equipamento em que o

aplicativo for instalado, o grande causador desses empecilhos é a *webview* como o aplicativo é executado dentro de um navegador nativo do sistema acaba se tornando difícil o acesso a ferramentas nativas do sistema, na grande maioria dos casos a cliente final nem sentirá a diferença entre um aplicativo nativo e um aplicativo híbrido, pois grande partes destes não possuem serviços que rodam em segundo plano ou renderização de modelos 3D como em jogos. Como este projeto não conta com nenhum deste impedimento ele foi inteiramente construído com tecnologias híbridas mais caso ele possuísse uma grande quantidade processamento de dados ou muitos serviços em segundo plano a melhor escolha seria o desenvolvimento nativo.

Para que o usuário possa ter acesso às demais funcionalidades do aplicativo ele deverá se cadastrar fornecendo seu CPF, nome, e-mail, telefone e uma senha como mostra a figura 4. Após o cadastro, o usuário terá acesso às demais opções do aplicativo como compra de bebidas, localização de estabelecimentos e formas de consumo.

Figura 4 - Layout para cadastro do cliente

Após o cadastro no aplicativo, o usuário pode fazer *login* e ter acesso a loja como ilustra a Figura 5.

Figura 5 - Layout da tela de login

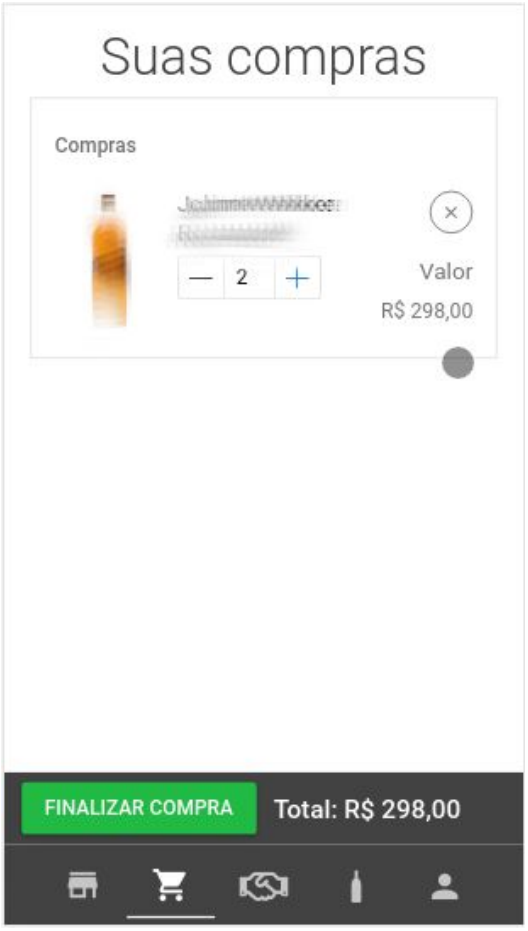
Após o seu desenvolvimento, na fase de emprego do produto, foram realizadas uma série de testes de fluxo para validar se os processos de acessar o aplicativo, buscar pelo produto e efetuar compras, estavam devidamente corretos para que o usuário final conseguisse usar o aplicativo em sua totalidade.

A partir destes testes foram levantadas algumas alterações de melhorias da usabilidade, dentre elas, o design e o layout da loja, pois ambos estavam com uma densidade muito grande de informações, o que confundia em momentos decisivos, como ilustra a Figura 5 com as melhorias já implementadas.

Figura 5 - Loja do aplicativo

Foram identificadas também algumas melhorias em relação ao carrinho de compras que estava com imagens cortadas ou distorcidas. Essa melhoria proposta foi inserida no software conforme ilustrado na Figura 6.

Figura 6 - Loja do aplicativo



Após atendimento à todas as melhorias propostas, o aplicativo passou a ser disponibilizado para um grupo maior de usuários que ainda estão testando e validando as informações.

Como o processo ainda está em andamento, algumas melhorias ainda podem se propostas a fim de lançar versões atualizadas com as correções de layout ou bugs encontrados por estes testadores.

4 CONCLUSÕES

A aplicação desenvolvida atendeu às expectativas de performance e portabilidade em diferentes plataformas, o que demonstra que mesmo com algumas limitações, as tecnologias híbridas mostram-se como boa opção no desenvolvimento de aplicativos comerciais.

Atualmente o projeto está sendo testado por um grupo de usuários da cidade de Recife e após a conclusão dos testes e levantamento de melhorias propostas, o mesmo deve ser disponibilizado para uso comercial em âmbito nacional.

REFERÊNCIAS

BANKER, Kyle. **MongoDB in action**. Manning Publications Co., 2011.

BLISCHAK, John D.; DAVENPORT, Emily R.; WILSON, Greg. **A quick introduction to version control with Git and GitHub. PLoS computational biology**, v. 12, n. 1, p. e1004668, 2016.

GARRETT, Jesse James. **The elements of user experience: user-centered design for the web and beyond. Pearson Education**, 2010.

GOVONI, Darren. **Java application frameworks**. John Wiley & Sons, Inc., 1999.

INTERNATIONAL DATA CORPORATION (IDC) (Framingham) (Comp.). International Data Corporation (IDC). **Worldwide Business Use Smartphone 2013-2017 Forecast and Analysis**. 2017. Disponível em: <http://www.idc.com/>. Acesso em: 06 maio 2019.
PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software-8ª Edição. McGraw Hill Brasil, 2016.

LOPES, Sérgio. **Aplicações mobile híbridas com Cordova e PhoneGap**. 2016.

STALLMAN, Richard M. **Software livre**. 2010. Disponível em: <https://www.fsf.org/pt-br>. Acesso em: 12 mar. 2010.

WARGO, John M. **Apache Cordova API Cookbook. Pearson Education**, 2015.

WARGO, John M. **Apacheplatform Desktop Applications: Using Node, Electron, and NW. js. Manning Publications** Co., 2017.

YU, Yue et al. **Exploring the patterns of social behavior in GitHub**. In: Proceedings of the 1st international workshop on crowd-based software development methods and technologies. ACM, 2014. p. 31-36.