DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

ShivaRadar: Aplicação para geolocalização de estabelecimentos gamers.

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Matheus Domingos da Silva e Silva

Florianópolis, 6 de março de 2020.

ShivaRadar: Aplicação para geolocalização de estabelecimentos gamers.

Matheus Domingos da Silva e Silva

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Ricardo José Rabelo

Banca Examinadora:

Prof. Ricardo José Rabelo
Orientador no Curso

Prof. Ricardo José Rabelo
Responsável pela disciplina

Prof. Carlos Barros Montez, Avaliador

Vanderlei Munhoz Pereira Filho, Debatedor

Iago de Oliveira Silvestre, Debatedor

I like the broken ones
because sometimes
one of my
shattered pieces
fills their cracks
and I get to make them
a little more whole
than they were before.

Agradecimentos

Aos meus pais, Maristela Maria da Silva e Silva e Sandro Domingos da Silva. Todo e qualquer reconhecimento escrito aqui é uma mera tentativa de traduzir e explicar meu amor e minha eterna dívida pela luta de uma vida inteira que vocês enfrentaram para educar a mim e a minha irmã. A minha namorada, Maria Eduarda Bernardo. Por todo apoio mental, físico e emocional que me proporcionou nesse período escuro que passei. Por ser uma mulher incrível, parceira e fiel. Por me apoiar e extrair o melhor de mim sempre (ou quase sempre). A meu sócio e amigo, Igor Tiosso Batistetti. Por acreditar nas minhas ideias e me colocar sempre em primeira opção. Por ser um irmão parceiro pra literalmente qualquer hora e principalmente tocar a empresa enquanto eu me dedicava a este projeto. A meus colaboradores Shiva, Luiz Felipe Aguero da Silva, Leonardo Ribak e Cassiano Presoto. Por também acreditar nas minhas ideias e principalmente por vestirem a camisa da empresa com muito amor. A meus amigos Brenno Araujo Queiroz e Marina Tavares por me auxiliarem em todas minhas dúvidas de programação sempre que possível. A Associação Atlética Acadêmica de Engenharia de Controle e Automação (ATACA) o qual fundei e presidi por anos. Por me integrar ao curso e fazer me sentir como uma segunda casa na UFSC. Por fim, por todos os professores que me iluminaram na caminhada do empreendedorismo, que me deram coragem a acreditar nas minhas ideias e no meu potencial.

"Você é o que você repetidamente faz.
Excelência não é um evento – é um hábito."(Aristóteles)

Resumo

A empresa Shiva e-Strategy n' Events é uma startup do ramo de esportes eletrônicos que atua nos segmentos de eventos e estratégia. Fundada pelo autor deste trabalho no ano de 2017, vem trabalhando na área de games suprindo as necessidades desse mercado inovador e em ascensão. Dentre essas necessidades, o PFC é uma continuação do Estágio Obrigatório, onde o foi realizado um Mapeamento *Gamer* em todo Brasil. Este mapeamento nada mais foi uma aquisição de dados e sua categorização em diferentes grupos. Foram capturados cerca de 370 estabelecimentos em todo Brasil, projeto financiado pela empresa multinacional Red Bull Brasil LTDA. A partir do resultado deste projeto foi possível dar início e validação a uma ideia muito maior, o desenvolvimento da plataforma Shiva. Esta plataforma tem como propósito principal conectar diferentes *players*, entusiastas e empresários do ramo de games e esportes eletrônicos, através de múltiplos serviços. Dentro do escopo do Projeto de Fim de Curso foi desenvolvido parte desta plataforma, o ShivaRadar, que é Uma aplicação focada em conectar fisicamente as pessoas, por meio da geolocalização, com estabelecimentos *gamers* por todo o Brasil. Por intermédio de tecnologias modernas como Node.js, React e React Native, a aplicação também tem como objetivo fomentar os cenários regionais, fazendo com que os entusiastas se conectem em locais que refletem suas personalidades. Por fim o trabalho mostra os resultados obtidos nessa criação, relatando detalhadamente os desafios, dificuldades e processos escolhidos, mantendo-se fiel a ideia inicial proposta.

Palavras-chave: Aplicativo, Geolocalização, Node.js, React, React Native, Mapeamento *Gamer*.

Abstract

The company Shiva e-Strategy n 'Events is a startup in the field of electronic sports that operates in the segments of events and strategy. Founded by the author of this work in 2017, it has been working in the area of games supplying the needs of this innovative and growing market. Among these needs, the PFC is a continuation of the Mandatory Internship, where a Gamer Mapping was carried out throughout Brazil. This mapping was nothing more than an acquisition of data and its categorization into different groups. Approximately 370 establishments were captured throughout Brazil, a project financed by the multinational company Red Bull Brasil LTDA. From the result of this project, it was possible to initiate and validate a much bigger idea, the development of the Shiva platform. This platform has as main purpose to connect different players, enthusiasts and entrepreneurs in the field of games and electronic sports, through multiple services. Part of this platform was developed within the scope of the End of Course Project, ShivaRadar, which is an application focused on physically connecting people, through geolocation, with gamers establishments throughout Brazil. Through modern technologies like Node.js, React and React Native, the application also aims to foster regional scenarios, allowing enthusiasts to connect in places that reflect their personalities. Finally, the work shows the results obtained in this creation, reporting in detail the challenges, difficulties and chosen processes, keeping the original idea faithful.

Keywords: Application, Geolocation, Node.js, React, React Native, Gamer Mapping.

Lista de ilustrações

Sumário

1 Introdução

1.1 Contextualização

A paixão pela qual o Brasil é conhecido no mundo esportivo é claramente visível em sua próspera cena de e-sports. Possui uma enorme audiência, com fãs conhecidos por sua dedicação a seus jogos e equipes. De fato, o Brasil tem o terceiro maior público de entusiastas de esportes do mundo, com 7,6 milhões de brasileiros assistindo a conteúdo profissional mais de uma vez por mês. [1]

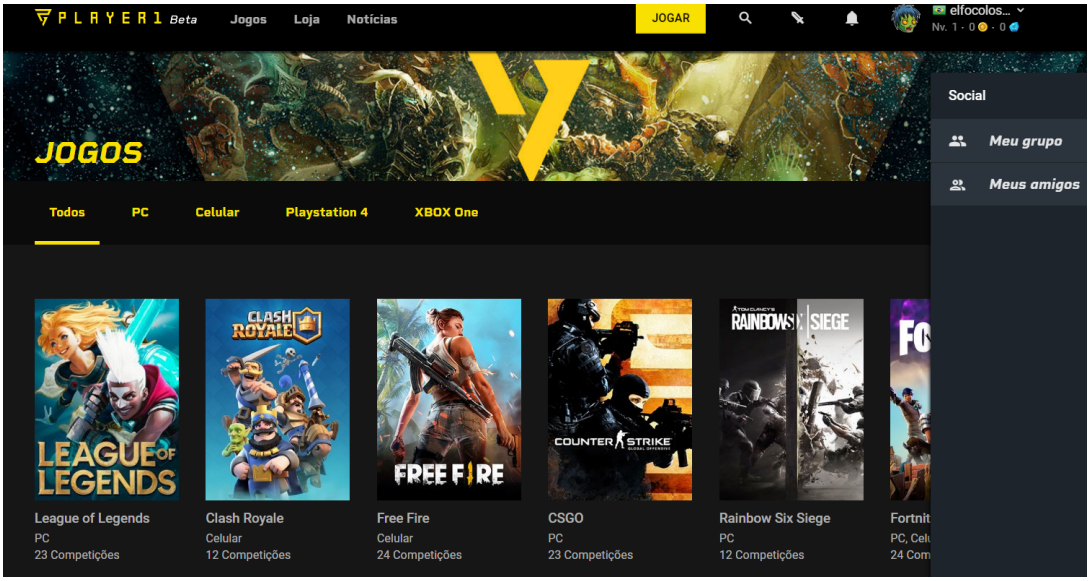
Figura 1 – Brasil: Principais estatísticas

Fonte: https://newzoo.com/

No mundo moderno, games nem sempre são apenas para diversão, onde competir profissionalmente em uma categoria de esporte eletrônico já é uma realidade no Brasil. O ciberesporte atrai inúmeras pessoas e fornece diversos benefícios e empregos, impactando assim, o lucro e a movimentação do país.

Dentro desse contexto, as *marketplaces* da área de jogos eletrônicos começam a aparecer no Brasil como modelo de negócio. Conhecidas também como plataformas, esses aplicativos tem como estratégia principal conectar os jogadores, jogos, empresas e competições. Como exemplo a "Player 1", uma plataforma 2 do Grupo Globo voltada para campeonatos amadores de esports de jogos como Free Fire, Clash Royale, Legue of Legends (LoL), Fortnite, Brawl Stars, FIFA 20, Counter Strike: Global Offensive (CS:GO), Call Of Duty: Mobile e Modern Warface. A ferramenta promove torneios online e também funciona como uma comunidade *gamer* para os jogadores que pretendem organizar ligas.

Figura 2 – Captura de tela - Menu da plataforma Player1



Fonte: Arquivo pessoal.

A grande limitação dessas plataformas é a quase inexistência de preocupação com os encontros presenciais dos usuários, focando mais na parte online dos games e esportes eletrônicos. Um problema para a indústria de entretenimento, uma vez que outros participantes desse ramo como luderias, lan house gamers, estabelecimentos geek e entre outros ficam de fora dessa conectividade. Sendo assim, a startup Shiva e-Strategy n' Events trás uma alternativa para o modelo atual de plataformas *gamers*. Após anos trabalhando com a realização de eventos amadores em Santa Catarina, em contato com pequenos empresários e estabelecimentos do ramo, percebeu-se um mercado com grande potencial, porém pouco explorado no Brasil.

A ShivaRadar é um aplicativo que busca conectar jogadores, entusiastas e pequenos empresários. Através de tecnologias modernas e eficientes a aplicação não apenas irá conectar estabelecimentos *gamers* e possíveis entusiastas, mas pode servir também de catalisador para todo mercado, reunindo em um mesmo espaço concorrentes, parceiros e investidores, cada um buscando atingir seus próprios objetivos.

1.2 Objetivo Geral

Desenvolver uma solução para conectar estabelecimentos *gamers* com jogadores, entusiastas e empresário através de uma aplicação multiplataforma.

1.3 Objetivos Específicos

- Desenvolver um *back-end* em JavaScript utilizando o ambiente de execução Node.js para implementação das regras de negócio;
- Desenvolver uma SPA utilizando JavaScript através da biblioteca React;
- Desenvolver de maneira hibrida, através do *framework* Expo.io e da biblioteca React Native, uma aplicação móvel;
- Permitir ao usuário buscar estabelecimentos *gamers* através da geolocalização;
- Possibilitar ao usuário filtrar a pesquisa de estabelecimentos por categoria;
- Permitir que os administradores gerenciem e manipulem os dados de maneira fácil e rápida;
- Apresentar os detalhes de funcionamento do aplicativo.

1.4 Justificativa

Com o amadurecimento da startup e percepção do mercado em que ela atua, o desenvolvimento de um aplicativo que venha a automatizar e facilitar o trabalho é algo natural e certo. Após anos testando diferentes nichos da área de games, percebeu-se que o melhor caminho hoje é conectar pessoas. Além disso, uma plataforma que valorize os pequenos estabelecimentos e valorize seus jogadores amadores também é um catalisador para a industria de jogos eletrônicos, algo inédito no território brasileiro, o que vem a somar ainda mais para a empresa.

Já as tecnologias e estratégias de desenvolvimento foram tomadas de acordo com os conceitos da escalabilidade e atualização tecnológica. A escalabilidade do software é sobre ter um código e uma arquitetura que é fácil de dar manutenção, de aumentar suas funcionalidades, de várias pessoas trabalharem nele. Já a modernidade é sobre usar linguagens, bibliotecas, *frameworks* e tecnologias que são usadas por grandes empresas do momento, com alto grau de suporte e expectativa de vida.

1.5 Estrutura do Trabalho

Este trabalho está organizado em 7 capítulos, sendo estes:

- Capítulo 1 – introdução: Informa sobre a temática do trabalho, descrevendo os objetivos traçados, para alcançar a solução, que será proposta por este trabalho.
- Capítulo 2 – Empresa: descreve a empresa em questão, bem como seu mercado de atuação e modelo de negócio.
- Capítulo 3 – Fundamentação Teórica e Ferramentas: Nesta parte são apresentados conceitos e definições importantes, além do referencial teórico, demonstrando a metodologia de desenvolvimento e as principais ferramentas utilizadas na construção da plataforma.
- Capítulo 4 - Modelagem da Plataforma: Apresentação da modelagem conceitual da plataforma, incluindo requisitos funcionais e não funcionais e modelagem de dados.
- Capítulo 5 – Desenvolvimento da aplicação: Neste capítulo, foram dispostas todas as etapas do desenvolvimento, bem como a plataforma desenvolvida, incluindo o banco de dados, sistemas auxiliares e o design das principais interfaces
- Capítulo 6 - Análise dos Resultados: Abordagem dos resultados do projeto, seu uso na empresa e cumprimento dos requisitos.
- Capítulo 7 - Conclusões e Perspectivas:neste capítulo, são apresentadas as conclusões do trabalho e perspectivas de trabalhos futuros.

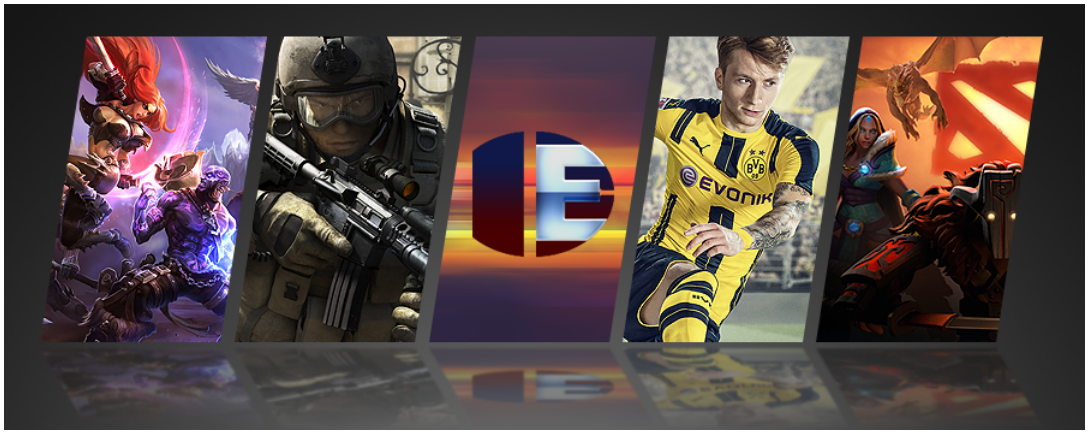
2 Empresa

2.1 A Shiva e-Strategy n' Events

A eSports Floripa surgiu de um *insight* do sócio-fundador, que em seu intercâmbio na Austrália, se confrontou com a cultura *gamer* do lado oriental do planeta, liderados por China, Japão e Coreia do Sul.

Ao voltar para o Brasil, percebeu que este mercado ainda era pouco explorado, porém se encontrava em crescimento acelerado. Por Florianópolis ser reconhecida nacionalmente como polo de tecnologia, desenvolvimento de software e fonte de muitas startups, acreditou-se ainda mais no potencial deste mercado. Assim, surgiu a ideia de criar a marca eSports Floripa, tendo como o objetivo principal de promover, divulgar e integrar o cenário e-sports na região da grande Florianópolis.

Figura 3 – Logo eSports Floripa aplicado



Fonte: Arquivo pessoal.

Resultado de uma reestruturação da eSports Floripa, a Shiva e-Strategy n' Events vem com objetivo de expandir suas áreas de atuação e levar o profissionalismo ao mercado de e-Sports. A Shiva apresenta-se como uma startup com dois grandes enfoques: Eventos e Estratégia, ambos inseridos no mundo *gamer*, cenário em crescente ascensão no país.

Composta por universitários entusiastas, hoje a Shiva conta com uma equipe eclética de empreendedores. Sendo assim nosso modus operandi é adaptável e personalizado, se moldando a cada projeto abraçado. Sua missão é criar oportunidades únicas no esporte eletrônico através de eventos e estratégia.Sendo assim, sua visão é enriquecer o mercado de esporte eletrônico no sul do Brasil com responsabilidade e profissionalismo.

Na área de eventos, com o *know-how* adquirido nos últimos anos, a Shiva consolidou-

Figura 4 – Copa Catarinense de League of Legends

Fonte: Arquivo pessoal.

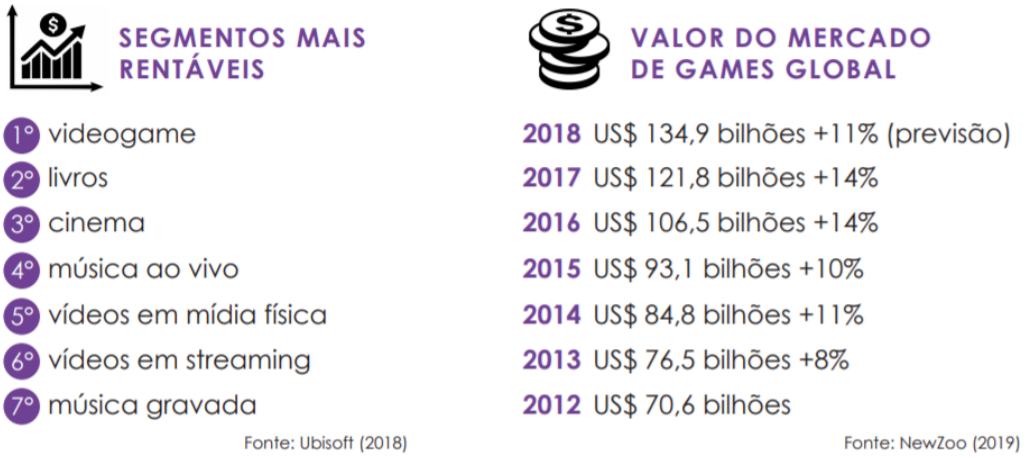
se como a grande organizadora de eventos voltados ao esporte eletrônico no estado de Santa Catarina, trazendo torneios de diversos segmentos do mercado. Já a área de estratégia busca trazer, além de profissionalismo, mais assertividade para os investidores no segmento de e-Sports. Por meio de levantamento e análise de dados a Shiva encontra oportunidades para as empresas lucrarem mais e gastarem menos.

2.2 Mercado de e-Sports

Mas afinal, o que são e-Sports? Esportes Eletrônicos ou eSports é uma nova modalidade surgida há poucos anos que vêm dominando o mercado de games e atraindo legiões de jovens no mundo todo. O mercado de games eletrônicos já não é mais apenas uma brincadeira de crianças. Nos últimos anos, a indústria de e-sports cresceu exponencialmente e vem gerando muito lucro.

O mercado de eSports está em crescimento e deverá alcançar mais de 450 milhões de pessoas em todo o mundo, o Brasil ocupa o terceiro lugar com maior número de jogadores com cerca de 66 milhões. O segmento de games está entre os que receberão investimentos do governo nos próximos anos, e já foram anunciados R$ 100 milhões para esta área. A indústria de videogames já é a maior do setor de entretenimento mundial, superando a produção de filmes, livros e músicas, como mostra a figura 5.

Figura 5 – Análise do mercado global



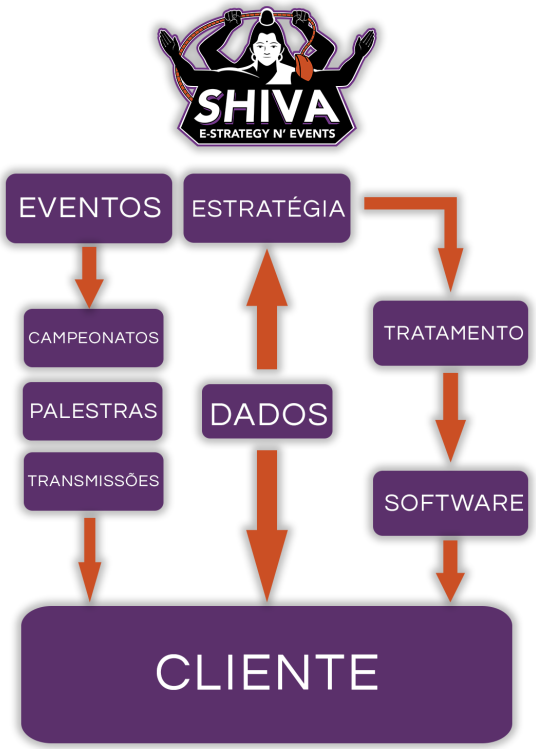
Fonte: Ubisoft(2018) e NewZoo(2019).

2.3 Modelo de negócio

No primeiro momento a empresa, composta de duas grandes áreas, funcionava de modo simples. A primeira área, de Eventos, ficava encarregada de organizar ações como campeonatos, transmissões e palestras, que geravam dados para a segunda área, de Estratégia. Esta, por sua vez, poderia agir de diferentes maneiras, apenas vendendo os dados crus ou tratando-os com o intuito de potencializar sua utilidade. O diagrama da figura 6 mostra como era o modelo de negócio da empresa.

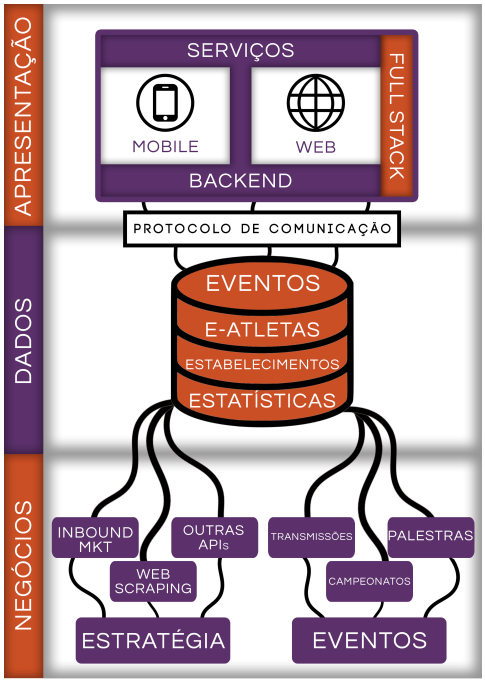
No novo modelo de negócio da empresa, a área de Eventos manteve suas operações da mesma maneira, porém passou a dar um maior valor nos dados e informações adquiridas em suas produções. Enquanto a área de estratégia tornou-se o foco da empresa, sendo responsável por monetizar esses dados de uma maneira efetiva e escalável. Com o desenvolvimento da Plataforma Shiva, em parte descrita aqui como ShivaRadar, a área de Estratégia passou por uma mudança estratégica, também assumindo papel na aquisição dos dados e tratando-os de maneira mais complexa, possibilitando que eles sejam usados tanto na Plataforma Shiva quanto em serviços personalizados.

Figura 6 – Modelo de negócio inicial



Fonte: Arquivo pessoal.

Figura 7 – Shiva Moderna



Fonte: Arquivo pessoal.

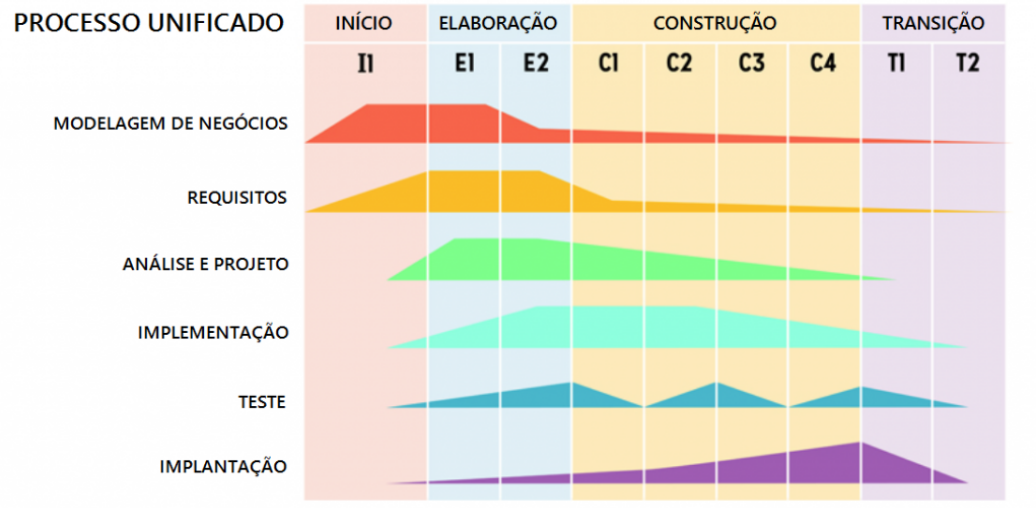
3 Fundamentação Teórica

3.1 Processo Unificado

O Processo Unificado (PU) nasceu como um processo popular para o desenvolvimento de software visando à construção de sistemas orientados a objetos. É um processo iterativo e adaptativo de desenvolvimento e ganhou visibilidade devido a maneira organizada e consistente que permite conduzir um projeto. [2]

Este modelo enfatiza que a equipe deve construir os projetos arquiteturais, junto com o produto de serviço. Como as fases do modelo foram elaboradas para ocorrerem iterativamente, elas podem acontecerem paralelamente. Desse modo, ao invés do processo acontecer em uma sequência de fases, os desenvolvedores podem criar a arquitetura enquanto desenvolvem testes, visto em detalhe no gráfico 8.

Figura 8 – Gráfico das fases do Processo Unificado



Fonte: http://tassinfo.com.br/

3.1.1 Início

A primeira fase do processo unificado é chamada de início (*inception*). Esta fase foi criada para ser pequena, com tempo apenas para garantir que tem uma base suficientemente forte antes de dar continuidade ao projeto.

De fato esta é a única fase do processo unificado onde as ações não se dão por iterações. Se a fase de início for longa, quer dizer que foi gasto demasiado tempo levantando

requisitos, onde o objetivo é verificar se existe um caso de negócios bom o bastante para dar início ao serviço. O que muitas vezes significar verificar se temos uma razão financeira boa o suficiente para o projeto.

Para verificar isto, a fase de início se utiliza de casos de uso, que delinearão as principais interações do usuário com o produto. Nesta fase também define-se o escopo do projeto, e seus potenciais riscos.

No fim da fase de início você atinge o primeiro marco (*milestone*) do ciclo de vida do processo. Neste ponto você terá uma descrição razoável do que se trata o produto e estará pronto para seguir à próxima fase de elaboração.

3.1.2 Elaboração

A fase de elaboração é a primeira fase do processo unificado a implementar as pequenas iterações citadas acima. O objetivo desta fase é basicamente criar um modelo ou protótipo do produto, o qual será refinado no futuro. Definindo assim a arquitetura inicial do produto.

Os desenvolvedores refinam os requisitos levantados na fase de início e também definem os principais requisitos na documentação da arquitetura, como diagramas de caso de uso e diagramas de classe. Isto dará a fundação para que o desenvolvimento aconteça.

Lembre-se, esta fase permite iterações, então o protótipo criado em uma iteração poderá sofrer alterações e ser reconstruído, até que os modelos de arquitetura e requisitos estejam maduros o suficiente para seguir em frente. No fim da fase de elaboração os desenvolvedores entregam um plano de desenvolvimento, que ocorrerá na próxima fase de construção. Este plano basicamente aprimora o que foi desenvolvido na fase de início e integra com tudo que foi aprendido na fase de elaboração, para que a construção do produto seja efetiva.

3.1.3 Construção

Lembre-se de que neste modelo o desenvolvimento pode acontecer em paralelo, isto significa que quando você inicia a fase de construção você continua fazendo serviços iniciados na fase de elaboração. A única diferença é de que a enfase no serviço pode mudar. Por exemplo, testes e verificações são importantes na fase de elaboração e se tornam ainda mais importantes na fase de construção.

A fase de construção é bem simples, é outra fase onde o desenvolvimento pode ocorrer iterativamente. Se enfoca em construir sobre o serviço feito em elaboração. Nesta fase é onde o produto ganha vida. Casos de uso mais específicos são criados para detalhar as funções do produto. E o desenvolvimento ocorre iterativamente até que o produto esteja pronto para ser lançado.

3.1.4 Transição

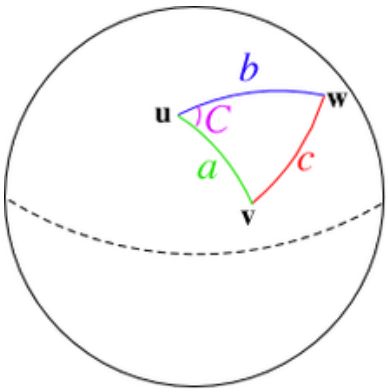
Neste marco do projeto sua equipe inicia a transição do produto aos clientes e usuários finais. A equipe de desenvolvimento recebe informações dos usuários finais e verificam se o produto realmente atende as necessidades do cliente. E ao receber estas informações sua equipe pode aprimorar o produto, consertar *bugs* e planejar próximas versões. Após o processo ter finalizado esta iteração, ele pode retornar e reiniciar o processo novamente, o que normalmente ocorre quando se planeja lançar uma nova versão do produto levando também em conta as informações coletadas dos usuários.

3.2 Fórmula de *Haversine*

A distância entre locais, como por exemplo, cidades, estados e países é delimitado pela latitude e longitude, ou seja uma referência de localização geográfica (coordenadas geográficas). A latitude é a distância de um ponto a linha do Equador, no qual esta divide os polos norte e sul, já a longitude é a distância até o meridiano de Greenwich, que por convenção, separa o hemisfério oriental (leste) e ocidental (oeste). Tanto a latitude, como a longitude são representados em graus, partindo-se do 0º, a latitude aumenta até 90º (para norte) e diminui até -90º (para sul), já para a longitude aumenta até 180º (para leste) e diminui até -180º (para oeste). [3]

A fórmula que será construída é conhecida como fórmula de *Haversine* 9, e é uma das equações básicas na navegação, tendo como embasamento matemático, a Lei dos Cossenos, considerando no modelo a curvatura da Terra, isto é o raio da mesma, que possui o valor de aproximadamente 6371 km. A fórmula de *Haversine* é uma importante equação usada em navegação, fornecendo distâncias entre dois pontos de uma esfera a partir de suas latitudes e longitudes. É um caso especial de uma fórmula mais geral de trigonometria esférica, a lei dos *Haversines*, relacionando os lados a ângulos de uma esfera "triangular".

Figura 9 – Fórmula de Haversine



Fonte: https://pt.wikipedia.org/

Como se tratam de coordenadas geográficas que ligam dois pontos sobre esfera terrestre, poderemos trabalhar com a Fórmula de *Haversine* que é a equação utilizada em navegação, a partir de suas latitudes e longitudes. Quando aplicada à Terra, ela representa apenas uma aproximação, pois o nosso planeta não é uma esfera perfeita. O raio da Terra é variável, sabemos que nos polos é da ordem de 6357 km; enquanto no equador é de 6378 km. Como o raio é variável teremos que calcular utilizando o valor médio do raio que é de 6371km.

Essa fórmula é definida como sendo o seno verso de um ângulo α, onde temos a seguinte relação:

$$versin(\alpha) = 1 - cos(\alpha)$$

Abaixo temos o exemplo de como se aplica a fórmula de Haversine: O raio da terra é de 6371 quilômetros, então:

$$R = 6371$$

A diferença das latitudes dos pontos em radianos, então:

$$dlat = lat2 - lat1$$

A diferença das longitudes dos pontos em radianos, então:

$$dlong = long2 - long1$$

O valor de a é o quadrado da metade do arco dos pontos.

$$a = sen(dlat/2) * sen(dlat/2) + cos(lat1) * cos(lat2) * sen(dlong/2) * sen(dlong/2)$$

O valor de c é o resultado da distância em radianos.

$$c = 2 * atan2(sqrt(a), sqrt(1 - a))$$

$$distancia = R * c$$

3.3 Aplicativo Móvel

O desenvolvimento de aplicações móveis se tornou uma nova oportunidade de negócio para os desenvolvedores e para as empresas, tendo em vista o crescimento exponencial do número de *smartphones* no mercado. Cresce na mesma proporção a busca por aplicativos que satisfazem as necessidades dos usuários. Essas necessidades vão de entretenimento a aplicações para uso no trabalho que são vinculadas aos sistemas corporativos, muitas vezes de grande porte. Os aplicativos mobiles são instalados em *smartphones* que possuem diferentes sistemas operacionais como iOS e Android, estes diferentes ambientes de programação geram métodos de desenvolvimentos de aplicativos distintos tanto pelo custo como, pelo tempo de aprendizagem. [4]

3.3.1 Ambientes de Desenvolvimento

3.3.1.1 Desenvolvimento Nativo

Seu desenvolvimento é específico para determinado sistema operacional, podendo utilizar todos os recursos de hardware e de software disponíveis no dispositivo. Além do fácil acesso aos recursos do dispositivo, ele respeitará os padrões de design para cada plataforma, respeitando os padrões de usabilidade. Sua maior desvantagem é o fato de ser necessário desenvolver aplicativos em linguagens específicas para cada sistema operacional, como por exemplo swift para iOS e Kotlin para Android. Desenvolvimento Nativo é recomendado em casos onde a performance é imprescindível, principalmente em *apps* que vão usar muitos gráficos, como jogos. Além disso desenvolver um *app* nativo custa mais caro, pois demanda mais tempo e precisa de um programador específico naquela plataforma.

3.3.1.2 Desenvolvimento Híbrido

Desenvolvimento híbrido é a utilização de várias de tecnologias Web, como HTML5, JavaScript e CSS, em conjunto com alguns *frameworks* que tenham acesso às funções nativas do aparelho. Este tipo de ambiente permite que desenvolvedores web possam facilmente migrar para o desenvolvimento de apps híbridos, pois a curva de aprendizado é menor do que aprender Java para android e swift para iOS. O desenvolvimento híbrido é recomendado em aplicações onde não são necessárias performances robustas e funções avançadas. Umas das grandes vantagens além do custo, é a portabilidade do código, ou seja, com pequenos ajustes o aplicativo pode ser lançado paras as diversas plataformas, como Android, iOS, Firefox OS. [5]

3.3.2 Sistema Operacional

Existem diversas plataformas de sistemas operacionais em uso nos smartphones atuais, sendo essas o meio que permite que o usuário interaja com os aplicativos e itens

do hardware, como, câmeras, GPS, microfone, alto-falante. As principais plataformas são Android e iOS, diante disso, o projeto em questão desenvolveu o aplicativo móvel para operar em ambas as plataformas. [6]

Figura 10 – Market Share

Fonte: https://www.statista.com

3.3.2.1 Android

Sistema operacional desenvolvido pela Google, instalado em mais de 80% dos *smartphones* em 2015. O que justifica sua ampla utilização é que as marcas e os modelos que possuem esse sistema são os mais variados, de aparelhos com muitos recursos aos modelos mais básicos ou simples.

3.3.2.2 iOS

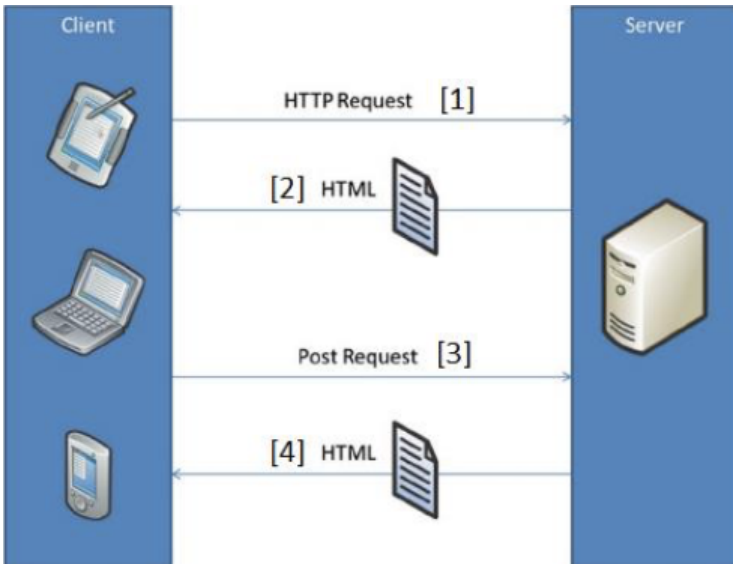
Plataforma introduzida no mercado em 2007 pela Apple e de utilização exclusiva em produtos da própria marca. Por ser de uso exclusivo da marca Apple, o iOS é desenvolvido para um modelo específico de hardware, permitindo assim, que suas potencialidades sejam exploradas ao máximo. Desde sua primeira versão, foi criado para utilizar em aparelhos com telas sensíveis ao toque. No final de 2015, 15.9 % dos *smartphones* vendidos utilizavam o sistema operacional da Apple.

3.4 Aplicações Web

3.4.1 Aplicações Web Tradicionais

Com o surgimento dos *webservers* (servidores web), que recebem as requisições e devolvem o conteúdo requisitado para o cliente, as páginas da web passaram a se tornar dinâmicas. O ciclo de vida destas aplicações pode ser visualizado abaixo:

Figura 11 – ciclo de vida das aplicações web



Fonte: FINK; FLATOW, 2014

Este fluxo pode ser traduzido da seguinte maneira: Inicialmente o cliente realiza uma requisição HTTP ao servidor, o servidor responde com uma página HTML através da submissão de um formulário, por exemplo, o usuário realiza uma requisição HTTP POST para o servidor. O mesmo retorna novamente uma página HTML, fazendo com que a página inteira seja recarregada e o novo conteúdo seja exibido para o usuário. Em toda interação entre o usuário e a página são enviadas requisições HTTP ao servidor,que é o responsável por controlar a aplicação. Neste tipo de aplicação a cada recarga na página, todos os arquivos HTML, CSS e JavaScript são baixados novamente, gerando um *overhead* (sobrecarga) desnecessário de cabeçalhos e requisições ao servidor. Esta abordagem gera muita lentidão, o que levou a criação do modelo de desenvolvimento moderno de aplicações web.

3.4.2 Aplicações Web Modernas

Ao contrário do desenvolvimento tradicional, o estilo de desenvolvimento moderno trabalha de forma assíncrona, buscando não bloquear a UI (interface do usuário) durante as requisições. Para isto, são utilizadas linguagens *client-side* nativas dos navegadores e

extensamente suportadas, como o JavaScript, para a manipulação do documento. Estas linguagens não necessitam da instalação de nenhum *plug-in* e são multiplataforma, uma vez que basta o dispositivo possuir um navegador que as interprete. Este estilo de desenvolvimento trabalha de forma independente da linguagem *server- side*, utilizando HTML5 e chamadas AJAX. Desta forma o servidor passa apenas a receber requisições feitas pelas chamadas AJAX e a responder com objetos JSON. O AJAX teve seu potencial visto por grandes empresas como a Google, tornando-se assim um padrão para criação de aplicações web modernas e contribuindo para a grande evolução da linguagem JavaScript. A sua capacidade de atualizar apenas uma parte da página sem precisar recarregá-la, utilizando chamadas assíncronas, revolucionou a experiência dos usuários.

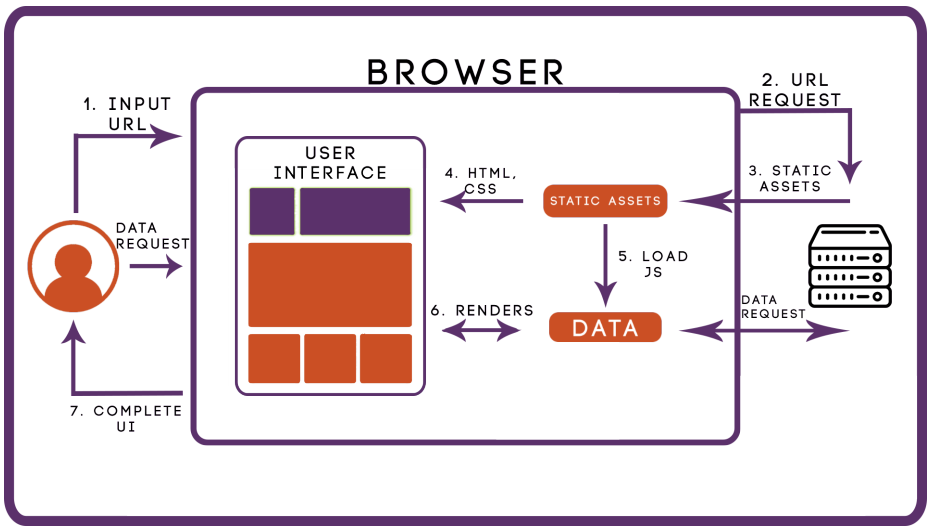
3.4.3 Single-page Applications

Junto a esse novo mundo do desenvolvimento web *front-end*, vieram as *Single-page Applications* (SPA), também conhecidas como aplicações de página única. A SPA consiste em uma técnica de desenvolvimento web que utiliza uma única página HTML como base para todas as outras páginas da aplicação. Podemos ilustrar o ciclo de vida de uma SPA na figura 13.

Este fluxo pode ser traduzido da seguinte maneira: O cliente faz um requisição HTTP ao servidor. O servidor retorna um documento HTML e todos os arquivos JavaScript e CSS que compõem a aplicação. Desta forma, este documento HTML é exibido pelo navegador e nunca é recarregado pelo servidor. Todas as próximas interações do usuário com a aplicação realizarão requisições AJAX para o servidor. O servidor, por sua vez, retornará todos os dados necessários via JSON ou partes de HTML pré-renderizados, que serão atualizadas na DOM (Document Object Model) e exibidas para o usuário. [7]

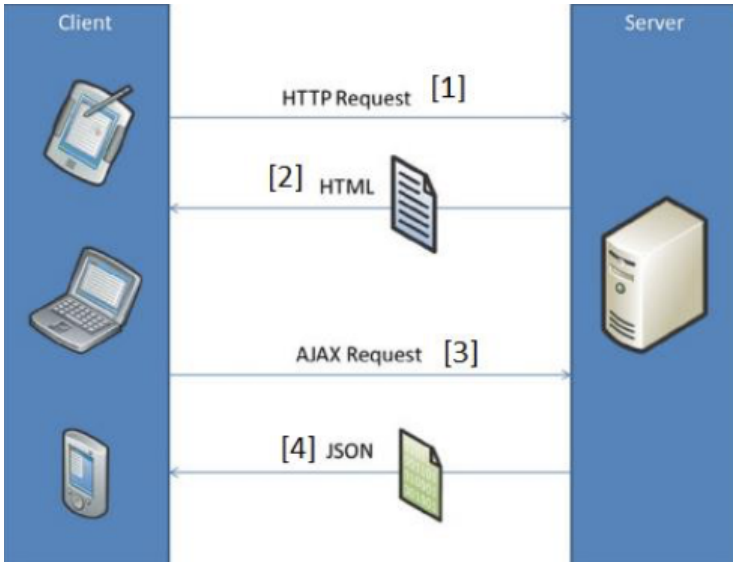
A SPA, que faz parte do modelo moderno de desenvolvimento web, diminui de forma considerável o *overhead* de requisições ao servidor, fazendo com que a aplicação fique muito mais rápida. Ao mesmo tempo garante melhor usabilidade para o cliente, uma vez que as requisições AJAX, por serem assíncronas, não bloqueiam a interface do usuário. A maioria das SPA são desenvolvidas do lado do cliente, não necessitando a recarga constante das páginas inteiras pelo servidor.

Figura 12 – Diagrama de uma SPA



Fonte: Arquivo pessoal.

Figura 13 – Ciclo de vida de uma SPA



Fonte: FINK; FLATOW, 2016

3.5 Tecnologias

No quadro 14, estão elencadas as tecnologias utilizadas no desenvolvimento da aplicação, bem como sua versão e uma breve definição. Nas subseções a seguir explica-se de maneira sucinta sobre cada uma delas.

Figura 14 – Quadro de tecnologias

Ferramentas	Versão	Referência	Finalidade
Visual Studio Code	1,42	https://code.visualstudio.com/	Editor de código-fonte para Windows, Linux e macOS.
Javascript	ES2018	https://www.javascript.com/	Linguagem orientada a objetos para aplicações web.
CSS 3	N/A	https://www.w3.org/Style/CSS/Overview.en.html	Cascading Style Sheets, onde se define estilos para seu projeto web.
HTML5	N/A	https://html.spec.whatwg.org/multipage/	Linguagem de marcação para a web.
Node.JS	13.7.0	https://nodejs.org/en/	Node.js é um interpretador de JavaScript
React	16.9.0	https://pt-br.reactjs.org/	Biblioteca JavaScript focada em criar interfaces.
React Native	0,61	https://facebook.github.io/react-native/	Desenvolver aplicativos para os sistemas Android e IOS de forma nativa.
Expo.io	2.14.0	https://expo.io/	Plataforma para criar aplicativos nativos para Android, iOS e Web.
MongoDB	4,2	https://www.mongodb.com/	Software de banco de dados orientado a documentos livre, escrito em C++.
Firebase	7.8.1	https://firebase.google.com/?hl=pt-br	Plataforma de desenvolvimento de aplicativos móveis e da Web
Yarn	1.21.1	https://yarnpkg.com/	Gerenciador de pacotes que também funciona como gerente de projetos.
Npm	6.13.6	https://www.npmjs.com/	Gerenciador de pacotes para a linguagem de programação JavaScript.
Express	4.17.1	https://expressjs.com/pt-br/	Estrutura de aplicativo da web e APIs para Node.js.
Nodemon	2.0.2	https://nodemon.io/	Utilitário que monitora qualquer alteração na sua fonte e reinicia automaticamente o servidor.
Insomnia	7.1.0	https://insomnia.rest/	Testar API para encontrar bugs.
Mongoose	5.8.11	https://mongoosejs.com/	Ferramenta de modelagem do MongoDB projetada para trabalhar em ambiente assíncrono.
Axios	0.19.2	https://www.npmjs.com/package/axios	Axios é uma biblioteca Javascript usada para fazer solicitações HTTP.
Cors	2.8.5	https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle_Acesso_CORS	Permite que um site acesse recursos de outro site mesmo estando em domínios diferentes.
Socket.io	2.0.0	https://socket.io/	Socket.io é uma implementação em node.js para web socket.

Fonte: Arquivo Pessoal

3.5.1 Visual Studio Code

O Microsoft Visual Studio Code é uma ferramenta desenvolvida pela Microsoft e tem ganhado um espaço e presença marcantes no cenário de desenvolvimento e programação de aplicações web pois possui suporte não somente a ASP.NET, mas também ao Node.js, PHP, Ruby, Python, entre outras mais. Sua origem foi no ano de 2015 e sua licença é gratuita tendo seu código fonte disponibilizado no GitHub. Vale explicitar que o editor não é uma cópia do Visual Studio Enterprise, por sua vez, é semelhante a outros editores como o Sublime Text, Atom ou Brackets. Neste trabalho usou-se o VSCODE como ferramenta principal para elaboração de todos os códigos do projeto independente da linguagem. [8] Este software foi escolhido como IDE pois foi fortemente recomendado pela comunidade de desenvolvedores e também já abordado durante a graduação.

3.5.2 Git

O Git é um sistema *open-source* de controle de versão utilizado pela vasta maioria dos desenvolvedores. Com ele pode-se criar todo histórico de alterações no código do projeto e facilmente voltar para qualquer ponto para saber como o código estava naquela data. Além disso, o Git nos ajuda muito a controlar o fluxo de novas funcionalidades entre vários desenvolvedores no mesmo projeto com ferramentas para análise e resolução de conflitos quando o mesmo arquivo é editado por mais de uma pessoa em funcionalidades distintas. [9] O uso do git também foi uma escolha por indicativa de pessoas e empresas referencia no meio de *dev*.

3.5.3 JavaScript

JavaScript encontra-se como a linguagem principal de programação utilizada para o desenvolvimento deste projeto, a qual funcionará em conjunto com HTML e CSS. Essa, foi inventada por Brendan Eich em 1995, sendo atualmente a linguagem mais usada em páginas Web e aplicações de servidores. Dinâmica e multi-paradigmática, ela suporta os estilos orientado a objetos, imperativa e funcional baseados em protótipos. Chamada também de linguagem *client-side*, uma vez que os *scripts* rodam no lado do cliente da web e não em um servidor. [10] A escolha dessa linguagem foi feita pois era a que melhor combinava performance e curva de aprendizado.

3.5.4 HTML 5

HTML significa 'Linguagem de Marcação de HiperTexto', sendo o componente básico para desenvolvimento de páginas Web. Ele tem, como finalidade, estruturar e apresentar as informações e o conteúdo dentro de um *website*. Analisando a terminologia, temos que hipertexto se refere aos links que conectam as páginas umas nas outras, tanto dentro de um site quanto outros diferentes sites. O *markup* é usado para mostrar os elementos, texto, imagens, e outros necessários, sendo comum a utilização de tags como <head>, <title>, <body>, , e muitas outras. A maior utilização no trabalho foi usufruir das tags para estruturação de todas as páginas do software agilizando o desenvolvimento do mesmo. [11] A compreensão do HTML é de suma importância para ter uma boa base para programação *front-end*, sendo assim optou-se por não utilizar-se de ferramentas de design e ilustrações prontas de *website*.

3.5.5 CSS 3

'Cascading Style Sheet' é uma maneira de facilitar o desenvolvimento de aplicação Web, pois centraliza as configurações referentes a exibição dos elementos como cores das letras, bordas, espaçamentos, tamanho dos títulos, posição de um elemento na tela ou

tamanho de uma imagem em um arquivo .css, futuramente importado para as páginas, garantindo assim uma padronização e melhorando o carregamento da mesma. Neste trabalho a utilização do CSS será essencial para compor a parte visual do software e melhorar a visibilidade dos elementos. [11] A justificativa do CSS 3 é a mesma já citada no secção HTML.

3.5.6 Node.JS

Recentemente, o JavaScript do lado do servidor tem recebido cada vez mais atenção. Node.js é orientado a eventos e o processamento de requisições I/O (*input* e *output*) não-bloqueante. O que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuído. [12] Uma plataforma construída em cima do motor de JavaScript do Chrome, o V8, está se tornando cada vez mais popular e algumas das razões para isso podem ser explicadas pela:

- (i) possibilidade de compartilhar o código entre o servidor e o cliente;
- (ii) seu gerenciamento eficiente de I/O;
- (iii) o maior ecossistema de bibliotecas de código aberto no mundo o Npm.

3.5.7 React

Segundo Hudson, atualmente é difícil trabalhar na web sem ter ouvido falar alguma informação sobre React. Criado pela equipe de desenvolvedores do Facebook, hoje é usado por grandes empresas do mercado como Uber, Airbnb, Netflix e outras empresas. De acordo com Robbestad, React não é um *framework*. React representa o V no padrão de construção MVC. Ele na verdade é uma biblioteca de Javascript para construção de interfaces de usuários que pode ser combinada com outras tecnologias. Outra característica importe é que o React é uma linguagem de programação declarativa, em poucas palavras, enquanto a declarativa se preocupa com o que o programador quer fazer, a imperativa quer saber como atingir o objetivo desejado. [13] A escolha da ferramenta foi natural, pois é essencial para qualquer profissional em front-end, uma vez que existe uma alta demanda do mercado por esses programadores.

3.5.8 React-Native

O React Native é uma estrutura JavaScript para escrever aplicativos móveis reais e nativos para iOS e Android. Ele é baseado no React, a biblioteca JavaScript do Facebook para criar interfaces de usuário, mas, em vez de segmentar o navegador, ele segmenta plataformas móveis. Em outras palavras: os desenvolvedores da Web agora podem criar aplicativos para dispositivos móveis que pareçam realmente "nativos", tudo a partir do

conforto de uma biblioteca JavaScript. Além disso, como a maior parte do código pode ser compartilhado entre as plataformas, o React Native facilita o desenvolvimento simultâneo para Android e iOS. [14] A escolha veio pelo fato de integrar muito bem com as outras ferramentas e também a possibilidade do desenvolvimento híbrido mantendo performance.

3.5.9 Expo.io

O Expo é uma ferramenta utilizada no desenvolvimento *mobile* com React Native que permite o fácil acesso às API's nativas do dispositivo sem precisar instalar qualquer dependência ou alterar código nativo. Ele oferece grande parte de recursos de forma nativa e integrada e, por exemplo, possibilita o acesso à recursos como câmera, microfone, *player* de música, entre outros, de forma muito simples utilizando essa ferramenta. [15] Dentre as tecnologias disponíveis no mercado, a que mais satisfez foi a Expo.io pela excelente documentação de apoio.

3.5.10 MongoDB

Desenvolvido pela MongoDB Inc e publicado pela GNU Affero General Public License e Licença Apache, é um software de banco de dados orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++ o MongoDB armazena dados em documentos semelhantes a estrutura JSON, o que significa que os campos podem variar de documento para documento e a estrutura de dados pode ser alterada ao longo do tempo. [16] O motivo para sua escolha foi, além do custo, a baixa verbosidade da sua sintaxe, tornando a comunicação de fácil codificação.

3.5.11 Firebase

Firebase é atualmente comandado pela Google, um conglomerado de tecnologias disponíveis em diversas linguagens de programação como: Java, Swift, Objective-C, Python, JavaScript (incluindo Node.js), Go, Unity e C++ [17]. Sua escolha veio no meio do desenvolvimento por oferecer serviços necessários a aplicação, que o MongoDB acabou não satisfazendo totalmente.

Entre todas as tecnologias oferecidas pelo Firebase, as usadas durante o desenvolvimento do software foram:

3.5.11.1 Firebase Authentication

Fornece serviços de *back-end*, SDKs e bibliotecas de IU prontas para autenticar usuários nos aplicativos, além de oferece suporte à autenticação por meio de senhas, números de telefone e provedores de identidade federados como Google, Facebook, Twitter.

3.5.11.2 Cloud Storage

O 'Cloud Storage' para Firebase é um serviço de armazenamento de objetos criado para a escala do Google. Com os SDKs do Firebase para 'Cloud Storage', é possível usar a segurança do Google para fazer o *upload* e o *download* de arquivos nos aplicativos Firebase. Também é possível utilizar os SDKs da google para armazenar imagens, áudio, vídeo ou outros conteúdos gerados pelo usuário.

3.5.11.3 Firebase Realtime Database

O 'Firebase Realtime Database' é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando um aplicativo é criado em plataformas cruzadas com os SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do 'Realtime Database' e recebem automaticamente atualizações com os dados mais recentes.

3.5.12 Yarn

Yarn é um gerenciador de pacotes para Javascript que também se torna gerente de projetos. Ele permite que seja possível a utilização de soluções de outros desenvolvedores para diferentes problemas, facilitando o desenvolvimento do software. Caso houver problemas, é possível relatar problemas ou contribuir de volta e, quando o problema for resolvido, poderá usar o Yarn para manter tudo atualizado. O código é compartilhado através de algo chamado pacote, um pacote contém todo o código que está sendo compartilhado, além de um arquivo package.json (chamado manifesto) que descreve o pacote. [18] A escolha do Yarn foi por mera facilidade e integração com as outras tecnologias modernas usadas no decorrer deste projeto.

3.5.13 Node Package Manager

Node Package Manager(NPM) é o gerenciador de pacotes padrão para o ambiente de tempo de execução JavaScript Node.js; também é usado como utilitário de linha de comando que interage com este repositório online, que ajuda na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências. A Npm já conta com mais de 35 mil pacotes (Jul-2013), são bibliotecas e aplicações de código aberto, e muitas são adicionadas todos os dias. Estas aplicações podem ser encontradas através do portal de busca da Npm. Uma vez encontrado o pacote que você deseja instalar, ele pode ser instalado com uma única linha de comando. [19] O uso do Npm foi necessário no gerenciamento de algumas tecnologias cuja instalação desempenhava melhor com seu uso.

3.5.14 Express

O Express é um *framework* para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel. Com uma miríade de métodos utilitários HTTP e *middleware*, é possível criar uma API de forma rápida e fácil. [20] Foi escolhido principalmente por sua robustez e compactuação com o resto as tecnologias.

3.5.15 Nodemon

Nodemon é uma ferramenta que ajuda a desenvolver aplicativos baseados no node.js, reiniciando automaticamente o node *exapplication* quando alterações de arquivo no diretório são detectadas. Também é possível citar quais arquivos e ou diretórios serão ignorados pelo file *watcher* do Nodemon, assim evitando *restarts* desnecessários, que é muito útil quando se tem uma aplicação que demore a reiniciar. [21] Ferramenta usada exclusivamente no desenvolvimento que melhora a velocidade de desenvolvimento de novas ideias, *softwares* e investimentos.

3.5.16 Insomnia

O Insomnia é um poderoso REST API Client com gerenciamento de *cookies*, variáveis de ambiente, geração de código e autenticação para Mac, Windows e Linux. Usado no desenvolvimento e teste do *back-end*, testando as rotas de maneira prática e rápida. [22] Seu software foi escolhido pela simplicidade que a ferramenta trás suas funcionalidades. De maneira sucinta você pode testar todas as rodas e regras de negócio em prática.

3.5.17 Mongoose

O Mongoose é uma ferramenta de modelagem de objeto do MongoDB, ou ODM (Object Document Mapper), escrita em JavaScript e projetada para funcionar em um ambiente assíncrono, fornece uma solução direta e baseada em esquema para modelar os dados dos aplicativos. Ele inclui conversão de tipo, validação, criação de consultas, e lógica de negócios. [23] Também indo de encontro com a familiaridade das tecnologias, o mongoose foi escolhido pela fácil montagem dos *schemas*, o que facilitou a conversa com o *back-end*.

3.5.18 Axios

Axios é um cliente HTTP, que funciona tanto no *browser* quanto em Node.js. A biblioteca é uma API que sabe interagir tanto com XMLHttpRequest quanto com a

interface http do Node.js, isso significa que o mesmo código utilizado para fazer requisições ajax no *browser* também funciona no servidor. As requisições feitas através da biblioteca retornam uma *promise*, compatível com a nova versão do JavaScript - ES6. [24] Por ser uma biblioteca pequena e poderosa, o Axios tem como objetivo conectar as API's requisitadas até ao final.

3.5.19 Cors

CORS (Compartilhamento de recursos com origens diferentes) é um mecanismo que usa cabeçalhos adicionais HTTP para informar a um navegador que permita que um aplicativo Web seja executado em uma origem (domínio) com permissão para acessar recursos selecionados de um servidor em uma origem distinta. Um aplicativo Web executa uma requisição *cross-origin* HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem. [25]O uso da tecnologia Cors permitiu que o domínio tenha comunicação livre, ideal para ambiente de desenvolvimento e aprendizado, por isso se escolheceu.

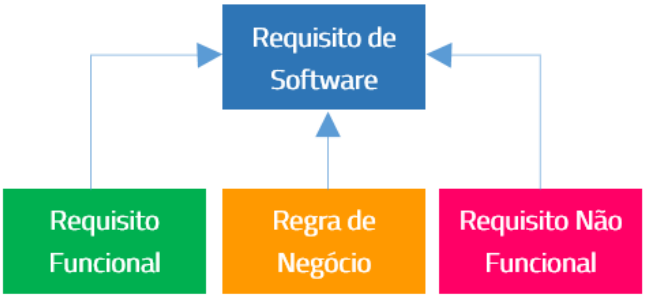
3.5.20 Socket.io

'Socket.io' é uma biblioteca JavaScript para aplicativos da web em tempo real, com ele é possível uma comunicação bidirecional em tempo real de clientes para servidores da Web. O Socket.io funciona com uma biblioteca do lado do cliente que é executada no navegador e outra biblioteca do lado do servidor para o Node.js. [26] Seu uso foi necessário para implementação do websocket, primordial para comunicação em tempo real.

4 Modelagem da Plataforma

Para a modelagem e planejamento da plataforma a ser desenvolvida, começa-se pela definição dos requisitos do software. De Acordo com Sommerville [27] , "Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações".

Figura 15 – Diagrama de Requisito de Software



Fonte: https://www.ateomomento.com.br/.

4.1 Requisitos de Usuário

Os requisitos de usuário dizem respeito a que serviços a plataforma fornecerá a seus usuários finais. Nos itens a seguir, a palavra "usuário"diz respeito a um jogador, entusiasta ou próprio estabelecimento.

- Um usuário deve poder registrar sua conta, incluindo seus dados de contato;
- Um usuário deve poder acessar sua conta através do seu e-mail;
- Um usuário deve poder alterar seus dados;
- Um usuário deve poder alterar deletar seu cadastro;
- um usuário deve poder buscar estabelecimentos por geolocalização;
- um usuário deve poder redirecionar para página web do estabelecimento;
- Um usuário deve poder filtrar sua busca;
- Um usuário deve poder avaliar o estabelecimento.

4.2 Requisitos Não-Funcionais

Conforme Sommerville [27], "Os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área".

No caso deste aplicativo, uma vez que almeja-se um grande fluxo de usuários, a confiabilidade e estabilidade são primordiais. Porém, além destes requisitos, grande atenção foi dada à questão da velocidade de uso e precisão da geolocalização, tendo em vista que a base da aplicação é conectar pessoas a lugares. Assim, foram definidas os seguintes requisitos não funcionais:

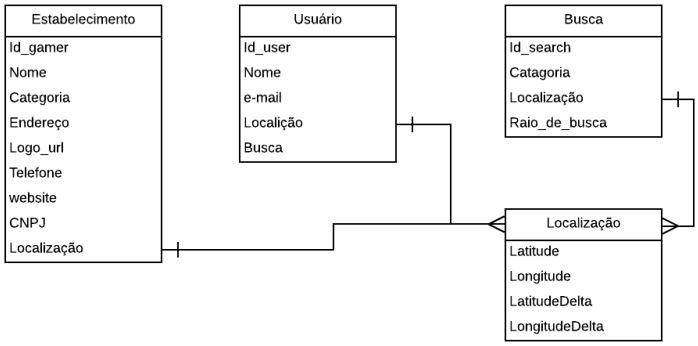
- A plataforma deve ser de fácil e prática operação;
- A plataforma deve ser precisa e rápida interação;
- A plataforma deve ter um visual minimalista;
- A plataforma não deve apresentar erros que atrapalhem o seu uso diário;
- A plataforma deve ser segura e confiável em relação ao dados de usuários.

4.3 Modelagem dos Dados

Uma vez que esta plataforma é primariamente um sistema de registro e consulta de informações, a modelagem se iniciou pelo detalhamento das estruturas de dados da plataforma. Através de estudo dos requisitos definidos na seção anterior, foi definido que a plataforma teria quatro entidades fundamentais, o "estabelecimento", "usuário", "localização"e "busca". Ao longo do desenvolvimento a complexidade e interações entre as entidades foram alteradas para satisfazer um bom funcionamento da aplicação.

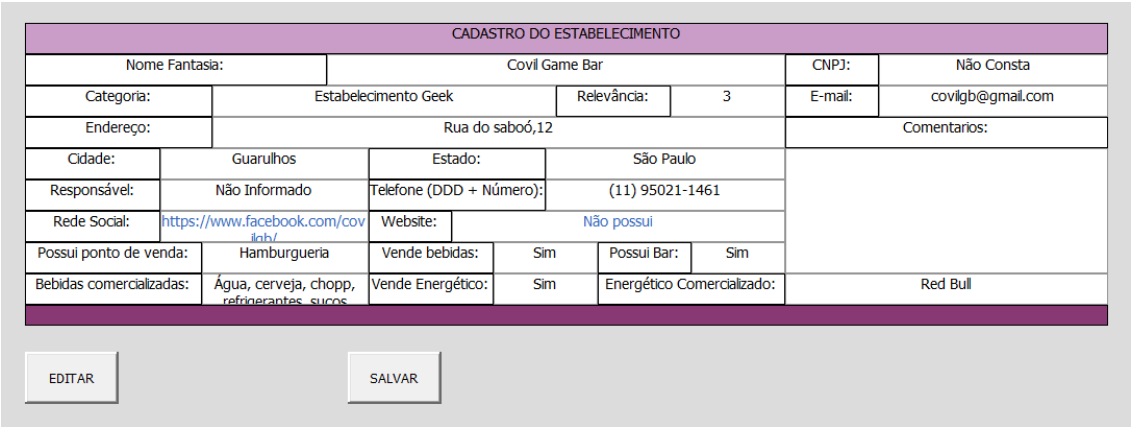
A entidade "estabelecimento"já tinha sua base de dados provinda de outro projeto, descrito no relatório de estágio. Neste projeto foi mapeado de maneira automática e inteligente todos os estabelecimentos do Brasil, extraindo o máximo de informações possíveis afim de atingir os objetivos propostos pela contratante, a multi-nacional Red Bull Brasil LTDA. Para está aplicação foram alterados e tratados esses dados para um melhor direcionamento da aplicação,por exemplo nesse projeto não há necessidade de saber qual tipo de energético se vende no estabelecimento, e também por questões contratuais.

Figura 16 – Modelagem de Dados



Fonte: Arquivo pessoal.

Figura 17 – Mapeameanto gamer - Cartão de Contato



Fonte: Arquivo pessoal.

4.4 Casos de Uso

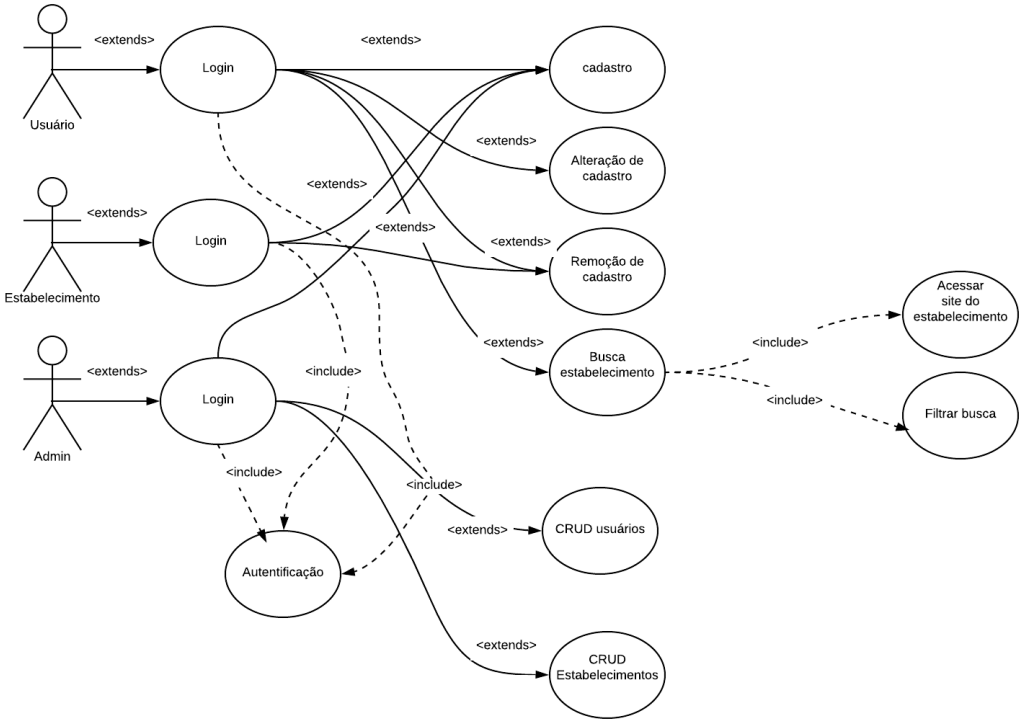
Após a conclusão da análise, iniciou-se uma nova fase do projeto chamada Casos de Uso, na qual desenvolveu-se diagramas UML e histórias de utilização do projeto. Pressman [28] define essa etapa como história estilizada sobre como um usuário final interage com o sistema sob um conjunto de circunstâncias específicas.

Um caso de uso é um recurso para documentar com o máximo de detalhes uma funcionalidade do sistema. Na documentação é descrito quem irá utilizar a funcionalidade, que são os atores; fluxos para utilização; pré-condições e o qual será o resultado final da funcionalidade.

Antes de começar a programação da aplicação em si, é necessário planejar o que realmente será preciso para o funcionamento correto da aplicação. Dessa forma foi feito então o diagrama de Caso de Uso da aplicação, como pode ser visto na figura abaixo. Conforme desenvolvimento do projeto e amadurecimento do planejamento o diagrama

sofreu várias alterações, onde alguns casos foram removidos, alterados e adicionados.

Figura 18 – Diagrama de Caso de Uso



Fonte: Arquivo pessoal.

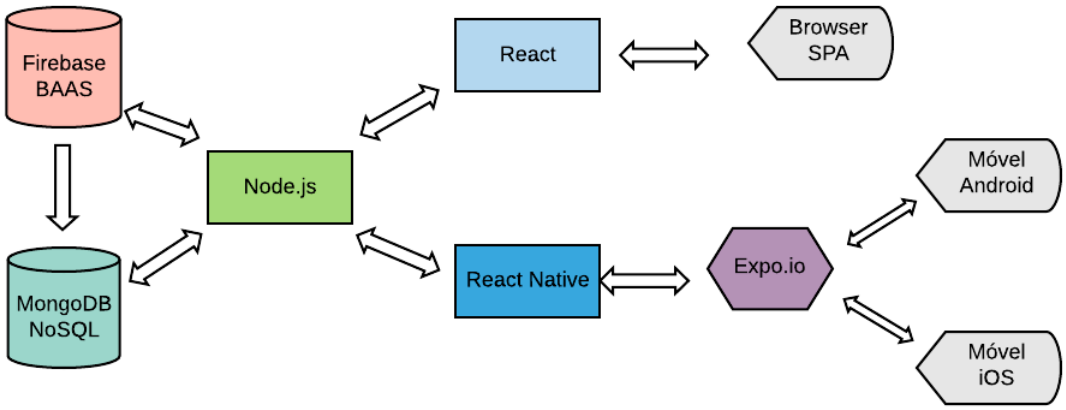
5 Desenvolvimento da Aplicação

Neste capítulo é apresentado o desenvolvimento do aplicativo desde o tratamento dos dados, passando por protocolos de comunicação, até a composição das telas no dispositivo do usuário final. O desenvolvimento foi *full-stack* pelo autor deste documento, somente auxiliado pelos colaboradores da empresa em partes visuais e de alinhamento funcional.

5.1 Arquitetura da Aplicação

A arquitetura global da aplicação é ilustrada na figura 19. O diagrama mostra as principais tecnologias utilizadas no desenvolvimento da plataforma. O banco de dados não relacional é hospedado no MongoDB e guarda boa parte dos dados. Este banco requisita as imagens salvas no Firebase quando necessário. Ambos conversam com o *back-end* que foi feito em Node.js, local onde as regras do negócio e rotas são aplicadas. Todas as requisições passam por esse processo e devem obter a autentificação fornecida pelo Firebase. A partir desde ponto, tem dois desenvolvimentos *front-end*, um para web e outro para *mobile*. A parte web foi desenvolvida em React, tendo uma estrutura SPA. Já a frente móvel foi desenvolvida em React Native através da ferramenta Expo.io, que possibilitou uma construção híbrida.

Figura 19 – Arquitetura global da aplicação



Fonte: Arquivo pessoal.

5.2 *Setup*

Após a aplicação bem estruturada iniciou-se o processo de programação e desenvolvimento propriamente dito, mas antes disso deve-se instalar e configurar as tecnologias no sistema operacional. Inicialmente foi configurado o VSCODE para um novo *workspace*, atualizando todas as extensões já usadas na IDE. Em seguida, seguindo as boas práticas, é fortemente recomendado o uso de gerenciadores de pacotes nesse momento, sendo assim utilizou-se o Yarn e Npm. Como usou-se o sistema operacional Windows 10, foi necessário alterar as definições de política de *script*, visto que no ambiente de desenvolvimento é necessário executar muitos comandos que violam as políticas padrões de proteção. Por fim, configurou-se o GIT para poder salvar os códigos e trabalhar de maneira mais segura, através da técnica do *gitflow*, sem se preocupar com backup ou alguma implementação dar errado e perder muito tempo tentando corrigir o problema.

5.3 Banco de Dados

5.3.1 Tratamento de dados

O tratamento dos dados foi realizado por questões práticas e judiciais, visto que o projeto anterior requeria sigilosidade das informações. Além disso, todos 370 estabelecimentos estavam no formato CSV. Sendo assim, tratou-se os dados reduzindo a quantidade e informações e converteu-se para JSON. Por último, como os dados continham também o endereço dos estabelecimentos, usou a API de geocodificação da Google para converter em latitude e longitude cada um desses JSON.

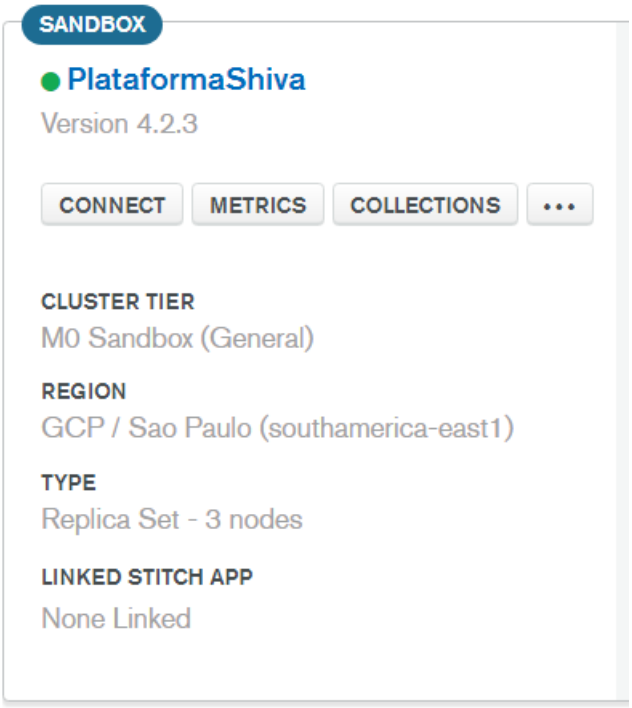
Figura 20 – Captura da API de geocodificação

Fonte: Arquivo pessoal.

5.3.2 MongoDB

O banco de dados da aplicação está localizado na nuvem, através do serviço de acesso remoto da MongoDB, visto na figura abaixo. O banco de dados projetado segue o modelo não-relacional, satisfazendo os requisitos do projeto. Outra facilidade é seu pacote gratuito de servidor, essencial nessa fase inicial do aplicativo, etapa que não tem como dimensionar de maneira precisa quais recursos irão necessitar de upgrade ou não. Por fim, MongoDB também oferece uma boa integração com o *back-end* em Node.js, tornando o processo de desenvolvimento mais fácil e escalável.

Figura 21 – Servidor Plataforma Shiva

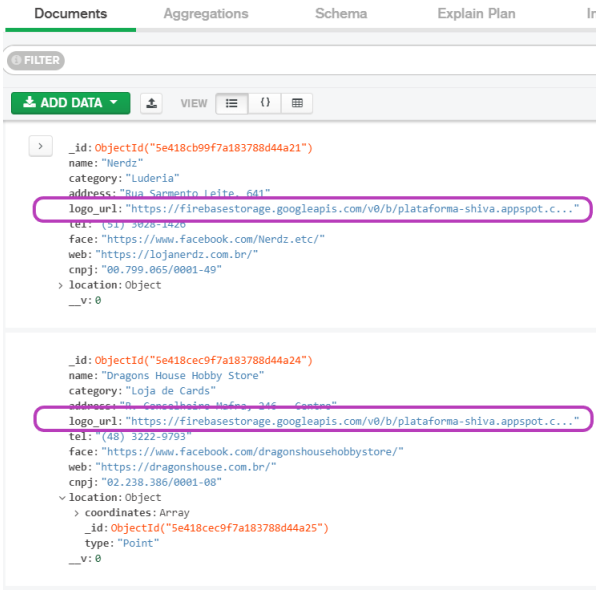


Fonte: Arquivo pessoal.

5.3.3 Firebase

Como existia muita discussão na comunidade de desenvolvedores sobre qual banco de dados era melhor e também existia a vontade de aprender as tecnologias de ponta que o mercado oferece, optou-se por também usar o Firebase, utilizando das vantagens que oferece sobre o MongoDB. Como o MongoDB não oferece o armazenamento de imagens, utilizou-se o serviço de *data storage* do Firebase, como mostra a figura 22. Também nessa etapa já deixou pré-configurado o sistema de login e autentificação, que será conectado posteriormente.

Figura 22 – Redirecionamento de dados

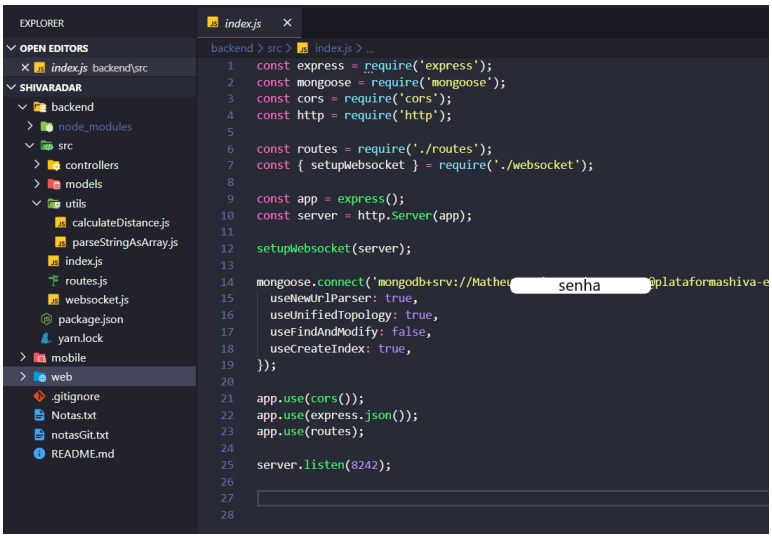


Fonte: Arquivo pessoal.

5.4 *Back-end*

O projeto teve início no *Back-end* para seguir o fluxo dos dados, construindo uma boa base da aplicação, utilizando ao máximo os recursos que o JavaScript fornece. Dessa forma, como citado anteriormente, optou-se por usar a plataforma Node.js, que facilitou o desenvolvimento e a integração com outras tecnologias. A primeira parte dessa etapa foi a conexão com o MongoDB, vista em detalhe no código na figura 23.

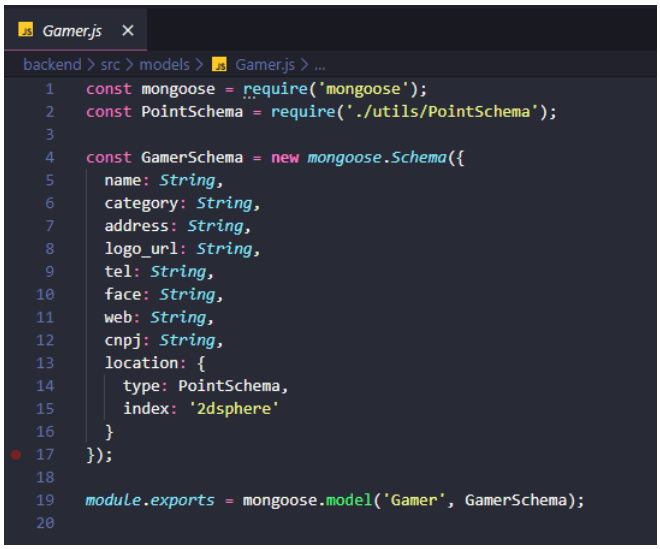
Figura 23 – index.js do backend



Fonte: Arquivo pessoal.

Com o banco de dados conectado, o próximo passo foi definir o modelo. Através do Mongoose utilizou-se *schemas* para modelar os dados da nossa aplicação. Tudo no Mongoose começa com um *schema*, e cada *schema* mapeia uma *collection* no MongoDB. Em detalhe pode ser visto na figura 24.

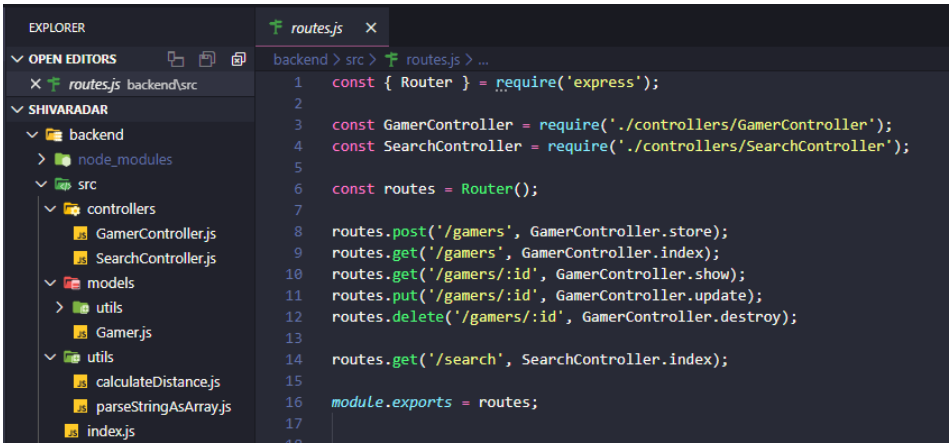
Figura 24 – Estabelecimento *Gamer Schema*



Fonte: Arquivo pessoal.

Após isso, implementou-se as rotas que representam os clássicos métodos HTTP (GET, POST, DELETE, PUSH) e também a rota principal de busca por latitude e longitude, parte onde a regra de negócio foi implementada. Para manter o código mais organizado e limpo, separou-se em dois controladores, parte do código é mostrada na figura 25.

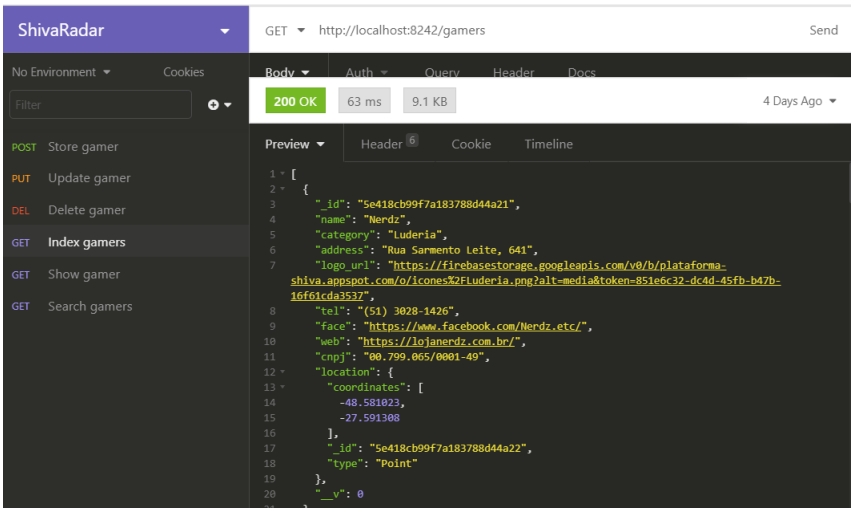
Figura 25 – rotas do *back-end*



Fonte: Arquivo pessoal.

Os testes de rotas e requisição, bem como a comunicação entre o banco de dados e o *back-end*, foram rodados através do software Insomnia Rest. Com uma interface bem visual e intuitiva, sua manipulação era fácil e eficiente, como mostra-se na figura 26.

Figura 26 – Testes de rota no Insomnia Rest



Fonte: Arquivo pessoal.

5.5 *Front-end* - SPA

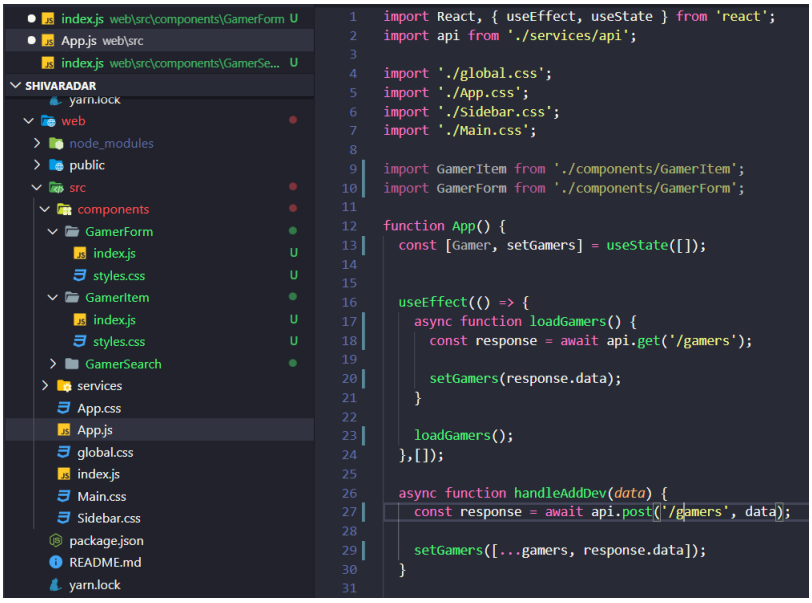
Nesta etapa do projeto iremos focar mais na parte visual e na experiência que o usuário vai ter ao usar a aplicação. Primeiro passo é a criação do projeto, como base foi utilizado do comando pré-existente CRA (*Create React App*), que cria um ambiente com configurações de *webpack* e babel básicas para a inicialização rápida de um projeto. Para o desenvolvimento do SPA foi necessário mudar a *view* básica oferecida pelo comando. Após isso, limpou-se todos os arquivos inúteis e organizou-se as pastas.

O próximo passo foi desenhar os blocos que irão dividir a tela, organizando a disposição dos componentes presentes na SPA. Sendo assim, começou pela <aside> com a criação dos componentes 'formulário' e 'busca'. Ambos componentes tiveram o uso do Hook de efeito através da combinação do *useEffect* com *useState*, não havendo a necessidade de escrever uma única classe. Parte dessa lógica pode ser vista na figura 27.

Já falando da parte mais visual e da UX, é importante explicar sobre a responsividade da aplicação. Pensando que o usuário pode acessar a página web com dispositivos de diferentes tamanhos de tela, programou-se a adaptação através da *flexbox* e *media query*, conforme mostrado na figura 28.

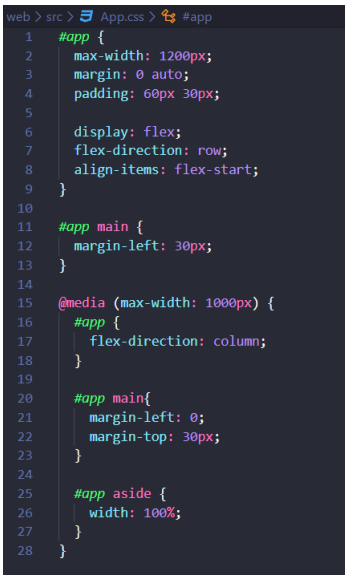
Como encontrou-se dificuldades de implementação do mapa na aplicação web, optou-se por deixar a SPA como um painel de função administrativa. Para isso criou-se

Figura 27 – Hook de efeito



Fonte: Arquivo pessoal.

Figura 28 – *Media Query* da aplicação



Fonte: Arquivo pessoal.

componente na <main> para substituir o mapa, denominado 'GamerItem', tendo um funcionamento semelhante a um índice, detalhado no código da figura 29.

5.6 *Front-end - Mobile*

O desenvolvimento Mobile teve como premissa continuar trabalhando com JavaScript e ter um desenvolvimento híbrido, ou seja, montar um *app mobile* que funcionasse

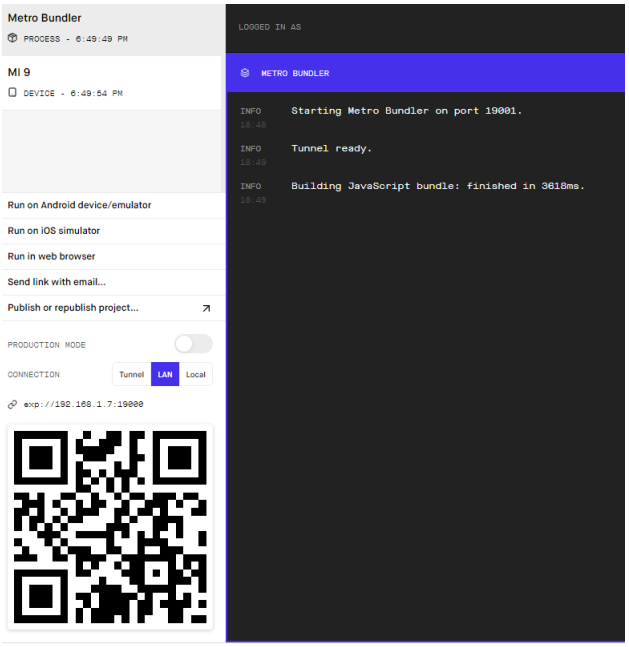
Figura 29 – 'GamerItem' - <main> da SPA

```jsx
import React from 'react';

import './styles.css';

function GamerItem({ gamer }){
  return (
    <li className="gamer-item">
      <header>
        <img src={gamer.avatar_url} alt={gamer.name}/>
        <div className="user-info">
          <strong>{gamer.name}</strong>
          <span>{gamer.category}</span>
        </div>
      </header>
      <p>{gamer.address}</p>
      <a href={uri: Gamer.web}>Acessar página do local</a>
    </li>
  );
}

export default GamerItem;
```

Fonte: Arquivo pessoal.

tanto para Android quanto iOS, mas que não fosse necessário se preocupar com especificidades dos sistemas operacionais. Dessa forma, a melhor maneira encontrada foi usar a plataforma Expo.io. Conforme na figura 30 é possível acompanhar o desenvolvimento do código de maneira interativa.

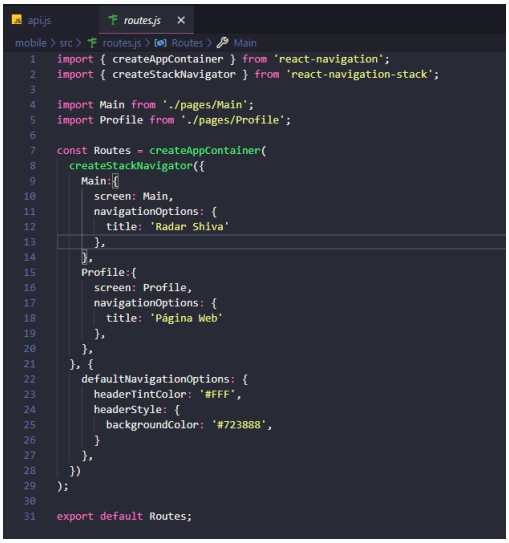
Figura 30 – Menu Expo *Developer Tools*



Fonte: Arquivo pessoal.

Após as instalações iniciais de ferramentas de desenvolvimento e suas configurações, inicia-se o projeto fazendo a parte visual primeiramente. Escolheu-se começar com duas páginas principais: 'Main' e 'Profile'. Sendo assim, criou-se o arquivo de rotas, responsável por coordenar a navegação entre as telas. A forma como a navegação foi configurada é por pilhas, como mostrar o código da figura 31.

Figura 31 – Rotas de tela com *stack navigation*



Fonte: Arquivo pessoal.

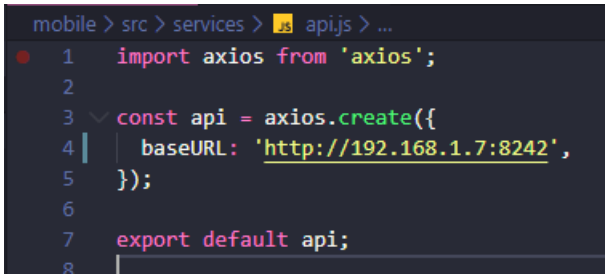
A página 'Main' vai ser onde a aplicação roda o layout do mapa e faz as buscas de estabelecimento por meio de filtros. Dessa forma, o próximo passo foi implementar o mapa dentro da página 'Main'. Sua importação e configuração foi feita através da biblioteca 'react-native-maps'. Também nessa etapa foi separado a parte de estilização da página fora do código principal, conforme mostra a figura 32.

Figura 32 – Tela 'Main' - Código fonte

Fonte: Arquivo pessoal.

Depois dessa parte de visualização, foi desenvolvido a parte funcional, onde os estabelecimentos são carregados no mapa. Através do Axios é feito a requisição para o Node.js que chama o MongoDB, visto na figura 33.

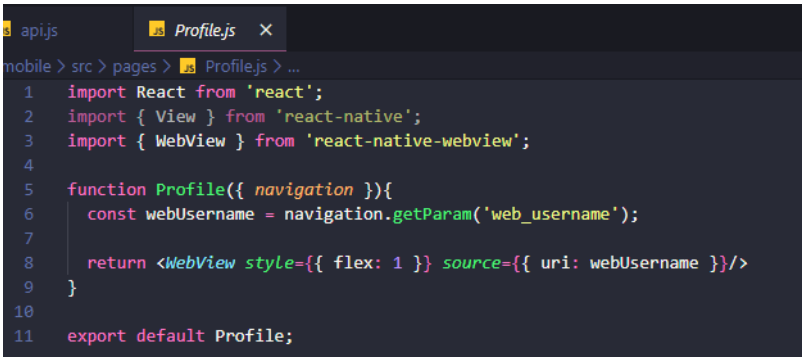
Figura 33 – Tela 'Main' - Conexão Axios



Fonte: Arquivo pessoal.

Depois do mapa já renderizado de maneira rápida, foi aplicado os *markers* na página 'Main'. Este elemento em conjunto com o *callout* será responsável pelo gatilho de troca de página. A página 'Profile' busca no MongoDB qual URL acessar e mostrar através do modo *WebView*, em detalhe na figura 34.

Figura 34 – Tela 'Profile' - *WebView*



Fonte: Arquivo pessoal.

5.7 Integração

Após testar todas as rotas e o funcionamento tanto do *mobile* quanto da web, inicou-se a implementação do protocolo *websocket*. O mesmo foi necessário pois quando um usuário tentasse cadastrar um estabelecimento que estivesse em um raio de 20 quilômetro de outro usuário, este dado novo deveria aparecer em tempo real para todos que estivessem procurando pela categoria em questão. Algumas modificações foram necessárias, nas figuras 35 e 36 podemos ver a implementação da configuração de comunicação e também da fórmula de *Haversine* para o cálculo de distância entre dois pontos através da latitude e longitude.

Figura 35 – Cálculo de distância - *Haversine*

Fonte: Arquivo pessoal.

Figura 36 – Configuração do *socket*

Fonte: Arquivo pessoal.

6 Resultados

6.1 Resultado da Aplicação Mobile

O *app* conta com um mapa, uma barra de busca na área superior da tela e um ícone de buscar ao lado. O usuário ao digitar a categoria do estabelecimento e pressionar o ícone de buscar, a tela do aplicativo irá mostrar os locais correspondentes da requisição em uma área de até 20 quilômetros. Após esta consulta aparecerá os ícones correspondentes aos estabelecimentos (figura 37) e ao tocar em cima abrirá uma caixa com as informações de cada estabelecimento com a possibilidade de acessar, com segundo toque, a rede social ou site registrado no banco de dados da plataforma.

Assim que conclui-se o código do aplicativo *mobile*, ele teve seu funcionamento testado na região da grande Florianópolis. No entanto não existiam estabelecimentos suficientes para realizar um teste piloto, por isso foi criado um banco de dados artificial para testar o funcionamento do aplicativo e suas funcionalidades. O banco de dados fictício possui 50 estabelecimentos *gamers* sendo eles distribuídos igualmente entre Estabelecimento Gamer, Luderia, Lan House Gamer, Loja de Cartas e Estabelecimento Geek na região da grande Florianópolis. Através do protocolo *websocket* o aplicativo era atualizado em tempo real assim que um novo estabelecimento era adicionado no banco de dados artificial. Os testes foram realizados internamente pelos funcionários da empresa Shiva e o objetivo dos exames eram encontrar estabelecimentos com perfil *gamer* na região de Florianópolis.

Em primeiro lugar os usuários tentaram encontrar estabelecimentos perto de suas casas buscando pela palavra chave "estabelecimento gamer"e obtiveram êxito na pesquisa, os resultados foram 3 comércios em uma área de 20 quilômetros do usuário. Após encontrar estes locais era possível obter mais informações dos estabelecimentos ao pressionar em cima dos ícones, abrindo uma caixa de informações onde os dados podiam ser vistos, os integrantes do teste tentaram acessar as redes sociais dos estabelecimentos pressionando em cima da URL mostrada na caixa de informações e (figura 37), o aplicativo obteve exito em redirecionar para o site desejado sem erros. A extensão máxima do aplicativo neste estado piloto foi redirecionar um cliente até o Web Site dos estabelecimentos cadastros e obter informações como endereço e número de telefone, apesar de parecer simples, não existem produtos hoje no mercado que fazem essa função para este nicho de mercado. O usuário consegue buscar por 5 tipos de estabelecimentos, sendo estes:

6.1.1 Estabelecimento gamer

Se enquadra neste tipo de estabelecimentos os locais onde possuem consoles, jogos de realidade virtual, além dos computadores convencionais de uma lan house.

6.1.2 Luderias

Se enquadra neste tipo de estabelecimentos os locais onde o foco do comércio é a utilização do espaço para jogos de tabuleiro e cartas.

6.1.3 Lan House Gamer

Se enquadra neste tipo de estabelecimentos os locais onde possuem computadores gamers capaz de rodar os jogos da atualidade.

6.1.4 Loja de Cartas

Se enquadra neste tipo de estabelecimentos os locais onde acontece a compra e venda de cards de jogos.

6.1.5 Estabelecimento Geek

Se enquadra neste tipo de estabelecimentos os locais onde possuem temáticas geek como por exemplo um restaurante temático de Star Wars.

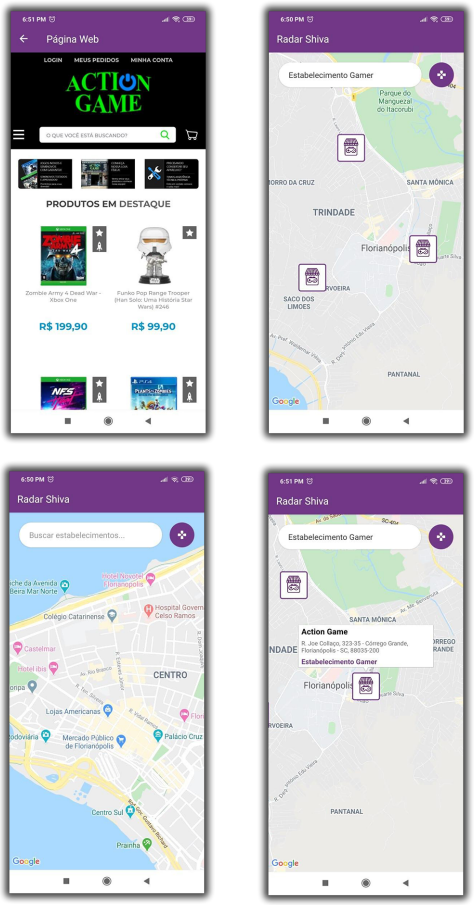
Figura 37 – Telas dos Estabelecimentos

Fonte: Arquivo pessoal.

6.1.6 Visual do Aplicativo Mobile

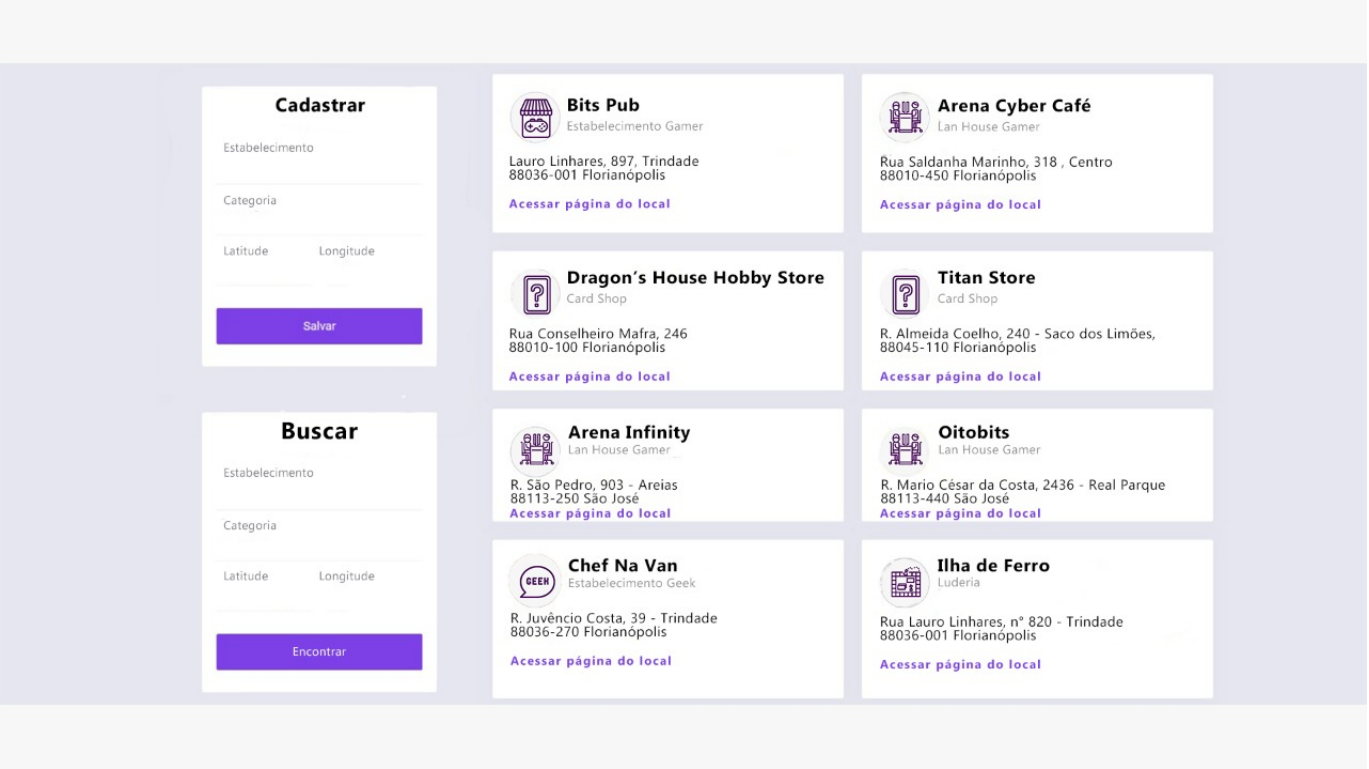
Ao longo do desenvolvimento da aplicação não foi possível se conectar com API gráfica do Facebook, por alguns problemas de autenticação que não foram resolvidos até o momento da conclusão do projeto, sendo assim não foi possível obter de maneira automática o logo dos estabelecimentos, desse modo optamos por manter as categorias já apresentadas (Estabelecimento Gamer, Luderia, Loja de Cartas, Lan House Gamer, Estabelecimento Geek). Assim foi criado um ícone para cada categoria, esses ícones foram armazenados na parte *data storage* do Firebase, pois o MongoDB não havia a mesma opção de maneira fácil. Nas bordas do aplicativo foram utilizadas tons de roxo que fazem parte da identidade visual da empresa onde o projeto foi realizado.

Figura 38 – Telas do ShivaRadar



Fonte: Arquivo pessoal.

Figura 39 – SPA versão 1.0



Fonte: Arquivo pessoal.

6.2 Resultado da Aplicação Web

A aplicação web da ShivaRadar ficou de uso exclusivo administrativo pois não foi possível desenvolver um mapa funcional para esta plataforma, por isso foi definido que nela seria possível cadastrar novos estabelecimentos e verificar os dados já existentes em forma de lista como demonstra a figura 39.

Apesar da frustração de não conseguir implementar um mapa na versão web, ela foi muito bem utilizada para encontrar todos os tipos de estabelecimentos de uma determinada categoria, como não existe um "raio de alcance"pela inexistência de um mapa, ao fazer uma pesquisa por estabelecimentos o aplicativo web retorna todos os correspondentes em todo o brasil, facilitando assim uma futura melhoria para grandes empresas que querem atingir vários comércios em todo o território nacional.

7 Conclusões e Perspectivas

O aplicativo tem como objetivo desenvolver uma aplicação multiplataforma utilizando recursos como Node.Js, React e React Native para geolocalização de estabelecimentos *gamers*, de forma que o engajamento entre jogadores de *e-sports*, patrocinadores e realizadores de eventos presenciais de campeonatos semi profissionais possam se comunicar de maneira efetiva e direta. No protótipo apresentado o usuário consegue consultar endereço, rede social, nome fantasia e telefone por estabelecimentos *gamers* exibidos no mapa na versão *mobile* e cadastrar novos estabelecimentos na aplicação web. Para atingir este objetivo foi utilizado um desenvolvimento híbrido, através da plataforma Expo.io, esta escolha se mostrou imprescindível pela facilidade em utilizar e testar o aplicativo tanto em Android como em iOS para acompanhar o desempenho do projeto em diferentes ambientes, já na aplicação web foi utilizado JavaScript por possui uma grande variedade de *frameworks* e suporte ativo.

Por fim o aplicativo vai além de conectar jogadores a empresas e cria um ecossistema de desenvolvimento do cenário *gamer* em grau regional sem concorrentes no mercado brasileiro. Os resultados percentuais oriundos desta etapa se distanciam da fase final, porém, a validação do aplicativo já foi confirmada através da venda do banco de dados presentes na versão piloto, o que reforça seu potencial de crescimento a longo prazo.

7.1 Limitações

O aplicativo conta com algumas limitações por estar na sua versão piloto, uma delas é suportar uma grande quantidade de estabelecimentos e usuários cadastrados, estes impasses são oriundos da utilização de softwares como o MongoDB e o Firebase de forma gratuita pois é possível utilizar somente uma pequena quantidade de armazenamento sem custos adicionais. O aplicativo *mobile* também não possui a opção de *login*, o que impede que seja gerado indicadores sobre o comportamento do usuário diante da aplicação, como por exemplo estabelecimentos mais acessados por determinada faixa etária, locais com a melhor avaliação da região e histórico de comércios já acessados. Outra limitação é a busca de estabelecimentos em todo o território nacional, pois o único modelo de exibição dos comércios são pelos ícones no mapa, ou seja, o usuário fica limitado a conhecer somente os locais em um raio de 20 quilômetros, impossibilitando a geração de uma lista com todos os correspondentes de determinada categoria no aplicativo.

Em relação a tecnologia empregada no aplicativo existem algumas limitações em vencer a informalidade no território brasileiro, pois vários estabelecimentos não estão nas redes sociais e não possuem CNPJ, o que impede a captura de dados de forma automática

e depende da ação dos donos dos estabelecimentos para se cadastrarem no aplicativo, porém com o decorrer do tempo a tendência é que as empresas cada vez mais saiam da informalidade e este problema seja minimizado. Para vencer algumas das limitações do aplicativo piloto, as possíveis soluções para estas modificações serão descritas no próximo capítulo.

7.2 Trabalhos Futuros

A versão final até então apresentada neste projeto pretende melhorar em diversos aspectos, começando pela questão de escalabilidade, armazenamento, indicadores de qualidade, expansão comercial, cadastro de eventos *gamers*, campeonatos de diversos jogos *online* e filtros de busca.

O começo de todo esse projeto se dá em um banco de dados com diversos estabelecimentos, porém como manter ele atualizado? Para sanar o problema de escalabilidade as próximas versões tem o objetivo de utilizar o CNPJ das empresas mapeadas e comparar seus CNAEs (Classificação Nacional de Atividades Econômicas), dessa forma seria possível gerar uma escalação dos CNAEs mais relevantes para cada tipo de estabelecimento e seu grau de confiança em predizer que um local com o registro de CNAEs da forma X, Y e Z correspondem a um determinado estabelecimento *gamer*. O potencial desse tipo de análise é enorme e pode ser feito toda vez que um CNPJ novo for registrado, fazendo com que o aplicativo tenha um banco de dados preciso e atualizado sem ter que depender da publicação do estabelecimento na internet para que ele possa ser achado por ferramentas de busca, porém como mencionado anteriormente a informalidade brasileira é um empecilho neste modelo de negócio.

Para resolver os problemas de armazenamento oriundos da utilização de softwares gratuitos o projeto necessitaria de investimento externo até que o aplicativo seja rentável suficiente para se manter em planos pagos que atendam as demandas da aplicação e possam oferecer uma experiência agradável ao usuário.

Do mesmo modo as tomadas de decisões do usuário dentro do aplicativo também precisam ser melhoradas nos trabalhos futuros, como por exemplo a falta de um sistema de *login*, possibilidade de avaliar o estabelecimento e filtro de busca, como não houve tempo suficiente para desenvolver todas as interações será necessário despender um esforço para remover essas limitações em versões futuras.

Outra implementação para as próximas versões seria a opção de cadastrar eventos *gamers*, *geeks* e diversos torneios de jogos *online*, este tipo de interação geraria mais conexões para o aplicativo pois os locais sedes de eventos também estariam cadastrados na plataforma, fazendo com que o aplicativo se retroalimenta entre usuários buscando torneios e torneios sendo realizados em estabelecimentos cadastrados na aplicação. Por fim

o projeto mais ambicioso a longo prazo para o aplicativo ganhar visibilidade seria através da estratégia *growhacking*, onde o aplicativo poderia ser traduzido para as principais línguas do mundo como: inglês, mandarim, espanhol, hindi e árabe com o intuito de atingir o maior número de mercados possíveis.

Referências

1 NEWZOO; BAR, E. *eSports in Brazil: Key facts, Figures, and Faces*. 2019. Acessado em 10/01/2020. Disponível em: <https://newzoo.com/insights/trend-reports/esports-in-brazil-key-facts-figures-and-faces/>. Citado na página 19.

2 SILVA, T. A. S. *Processo Unificado*. 04/08/2016. Acessado em 13/01/2020. Disponível em: <http://tassinfo.com.br/gestao-de-produto/processo-unificado/>. Citado na página 27.

3 ROA, C. R. *Distância entre locais (Latitude e Longitude)*. 01/10/2017. Acessado em 17/01/2020. Disponível em: <http://academicosdoexcel.com.br/2017/10/01/distancia-entre-locais-latitude-e-longitude/>. Citado na página 29.

4 SANTOS, S. R. *Aplicativos Móveis um Negócio Rentável: Tudo sobre como Ganhar Muito dinheiro criando Apps!* [S.l.]: SSTrader, 2018. (1ª Edição). Citado na página 31.

5 GONçALVES, D. *Compreenda as diferenças entre o desenvolvimento nativo e híbrido*. 7 de março de 2018. Acessado em 01/02/2020. Disponível em: <https://blog.cronapp.io/compreenda-as-diferencas-entre-o-desenvolvimento-nativo-e-hibrido/>. Citado na página 31.

6 NASCIMENTO, W. *Android vs iOS: compare os sistemas operacionais para celular*. 09/02/2020. Acessado em 09/02/2020. Disponível em: <https://www.techtudo.com.br/noticias/2020/02/android-vs-ios-compare-os-sistemas-operacionais-para-celular.ghtml>. Citado na página 32.

7 BAILãO, J. *Por que utilizamos Single Page Applications – SPA?* 3 JAN, 2019. Acessado em 22/11/2019. Disponível em: <https://imasters.com.br/front-end/por-que-utilizamos-single-page-applications-spa>. Citado na página 34.

8 MACORATTI, J. C. *Visual Studio Code – Apresentando o editor multiplataforma da Microsoft*. 22 JUL, 2016. Acessado em 17/12/2019. Disponível em: <https://imasters.com.br/desenvolvimento/visual-studio-code-apresentando-o-editor-multiplataforma-da-microsoft>. Citado na página 36.

9 PANZOLINI, B. *Gerenciando seus branches com o Git Flow*. 10/11/2018. Acessado em 07/01/2020. Disponível em: <https://tableless.com.br/git-flow-introducao/>. Citado na página 37.

10 MDN colaboradores da. *JavaScript*. 9 de set de 2019. Acessado em 28/01/2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Citado na página 37.

11 HYSLOP, B.; CASTRO, E. *HTML 5 E CSS 3*. [S.l.]: Alta Books, 2013. (7ª Edição). Citado 2 vezes nas páginas 37 e 38.

12 SANTOS, G. *Node.js — O que é, por que usar e primeiros passos*. 14/06/2016. Acessado em 22/01/2020. Disponível em: <https://medium.com/thdesenvolvedores/node-js-o-que-é-por-que-usar-e-primeiros-passos-1118f771b889>. Citado na página 38.

13 HUDSON, P. *Hacking with React*. [S.l.]: Leanpub, 2016. (1ª Edição). Citado na página 38.

14 DABIT, N. *React Native in Action*. [S.l.]: Manning, 2019. (1ª Edição). Citado na página 39.

15 CRUZ, R. *React Native + Expo*. 27/05/2019. Acessado em 19/01/2020. Disponível em: <https://medium.com/@rogercruz/react-native-com-expo-um-exemplo-40e5574c6904>. Citado na página 39.

16 GIAMAS, A. *Mastering MongoDB 4.x*. [S.l.]: Packt Publishing, 2019. (2ª Edição). Citado na página 39.

17 ORLANDI, C. *Firebase: serviços, vantagens, quando utilizar e integrações*. 01/01/2019. Acessado em 20/01/2020. Disponível em: <https://blog.rocketseat.com.br/firebase/>. Citado na página 39.

18 CAMPOS, D. *Yarn: A evolução do NPM*. 03/11/2016. Acessado em 20/01/2020. Disponível em: <https://tableless.com.br/yarn-evolucao-do-npm/>. Citado na página 40.

19 L, A. *O Que É npm? Introdução Básica para Iniciantes*. 22/05/2019. Acessado em 10/02/2020. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-npm>. Citado na página 40.

20 MDN colaboradores da. *Introdução Express/Node*. 27 de ago de 2019. Acessado em 05/01/2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introduç˜ao>. Citado na página 41.

21 LEWIS, J. *How To Increase Development Efficiency With Nodemon And Opn*. 25 de outubro, 2018. Acessado em 08/01/2020. Disponível em: <https://medium.com/@jeffrey.allen.lewis/how-to-increase-development-efficiency-with-nodemon-and-opn-4027b96175a1>. Citado na página 41.

22 NIEDRINGHAUS, P. *Postman vs. Insomnia: Comparing the API Testing Tools*. 1 de setembro, 2018. Acessado em 08/01/2020. Disponível em: <https://itnext.io/postman-vs-insomnia-comparing-the-api-testing-tools-4f12099275c1>. Citado na página 41.

23 LUIZ, T. *Mongoose — Criando queries*. Feb 1, 2018. Acessado em 23/12/2019. Disponível em: <https://medium.com/@thiagoluiz.nunes/mongoose-criando-queries-d72d38e8fece>. Citado na página 41.

24 BERNARDES, M. *Como usar Axios como cliente HTTP*. Jul 16, 2015. Acessado em 03/11/2019. Disponível em: <http://codeheaven.io/how-to-use-axios-as-your-http-client-pt/>. Citado na página 42.

25 MDN colaboradores da. *Cross-Origin Resource Sharing (CORS)*. 09 de ago de 2019. Acessado em 13/01/2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Controle_Acesso_CORS>. Citado na página 42.

26 CADENHEAD, T. *Socket.IO Cookbook*. [S.l.]: Packt Publishing, 2015. (1ª Edição). Citado na página 42.

27 SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson Education, 2011. (9ª Edição). Citado 2 vezes nas páginas 43 e 44.

28 PRESSMAN, R. *Engenharia de Software: Uma abordagem Profissional*. [S.l.]: Amgh Editora, 2016. (8ª Edição). Citado na página 45.