CENTRO UNIVERSITÁRIO UNIFACVEST
CIÊNCIA DA COMPUTAÇÃO
DIEGO TISCHER

CONSTRUÇÃO DE APLICATIVO PARA VERIFICAR VAGAS DE ESTACIONAMENTOS LIVRES EM GARAGENS CADASTRADAS POR SITE OU APP

LAGES – SC

2020

DIEGO TISCHER

CONSTRUÇÃO DE APLICATIVO PARA VERIFICAR VAGAS DE ESTACIONAMENTOS LIVRES EM GARAGENS CADASTRADAS POR SITE OU APP

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Unifacvest como requisito básico para a aprovação no curso de Ciência da Computação (TCCII)

Orientador (a): Marcio Sembay e Willen Leolatto Carneiro e Igor Muzeka

LAGES- SC
2020

CONSTRUÇÃO DE APLICATIVO PARA VERIFICAR VAGAS DE ESTACIONAMENTOS LIVRES EM GARAGENS CADASTRADAS POR SITE OU APP

RESUMO

O tema tem relação com a construção de um aplicativo, capaz de verificar vagas de estacionamento livres em garagens cadastradas pelo site ou no aplicativo, onde de forma prática e simples, o usuário acessa o aplicativo e verifica se no local próximo de destino possui uma garagem de estacionamento e a sua disponibilidade de vaga e faz sua reserva. Objetivo: Avaliar a situação de vagas disponíveis dos estacionamentos privativos, sugerindo a implantação de um programa para uso de um estacionamento e controle de vagas.

Palavras-chave: Estacionamento. Aplicativo. Geolocalização.

CONSTRUCTION OF APPLICATION TO CHECK FOR FREE PARKING SPACES IN GARAGES REGISTERED BY SITE OR APP

ABSTRACT

The theme is related to the construction of an application, capable of checking free parking spaces in garages registered by the website or in the application, where in a practical and simple way, the user accesses the application and checks if there is a garage at the destination and parking availability and make your reservation. Objective: To assess the situation of available parking spaces in private parking lots, suggesting the implementation of a program for the use of parking and parking control.

Keywords: Parking. App. Geolocation.

LISTA DE FIGURAS

LISTA DE SIGLAS

SP – São Paulo

iOS - Sistema operacional do iPhone

EUA – Estados Unidos da América

MG – Minas Gerais

APIs - Application Programming Interface

CPF – Cadastro de Pessoa Física

LPWAN - *low-power wide-area network*

LRWPAN - Low-Rate Wireless Personal Area Network

IDE – Ambiente de desenvolvimento integrado

SQL - Structured Query Language

SGBD - Sistema de Gerenciamento de Banco de Dados

XML - eXtensible Markup Language

JSON - JavaScript Object Notation

MVC - Model-View-Controller

APP - Application

UF – Unidade Federativa

SUMÁRIO

1. INTRODUÇÃO

1.1 APRESENTAÇÃO

Atualmente a modernidade tem permitido que, em todas as áreas, haja rapidez em deslocamentos e procedimentos, onde o trânsito se torna um dos desafios de todas as pessoas no dia a dia. Diante disto, grandes congestionamentos estão presentes nas cidades e isto acaba por gerar o problema de falta de espaço para todos os veículos existentes.

A mobilidade não se trata apenas sobre o fluxo do trânsito mas também sobre a sua disponibilidade em estacionar os veículos, além de outras necessidades. Além dos estacionamentos públicos, feitos na via, uma outra solução são os estacionamentos privativos que aumentam a mobilidade do trânsito, criando mais espaço para estacionar.

Estes estacionamentos também garantem aos veículos algumas proteções extras como segurança contra furtos, temporais, riscos e arranhões menos frequentes, além da economia de combustível na procura pela vaga disponível.

A problemática do tema envolve justamente a questão de existir espaço suficiente para o número de carros que rodam no trânsito todos os dias e também a disponibilidade de tempo para procurar vagas livres neste espaço, o qual muitas vezes o motorista perde muitos minutos até conseguir estacionar na vaga que está disponível.

Para ajudar neste controle, muitas ferramentas virtuais estão sendo criadas para auxiliar o motorista, indicando a disponibilidade de estacionamento, tornando a vida mais prática de quem necessita utilizar seu veículo todos os dias.

1.2 JUSTIFICATIVA

Atualmente é preciso gerar praticidade, comodidade e agilidade com se fala em trânsito e mobilidade. Portanto, de que forma isto é possível, envolvendo a tecnologia e contribuindo para a melhora de congestionamento e falta de mobilidade no trânsito e estacionamentos?

Para isto, a construção de um aplicativo, capaz de verificar vagas de estacionamento livres em garagens cadastradas pelo site ou no aplicativo, onde de forma prática e simples, o usuário acessa o aplicativo e verifica se no local próximo de destino possui uma garagem de estacionamento e a sua disponibilidade de vaga e faz sua reserva.

1.3 OBJETIVOS

1.3.1 GERAL

Avaliar a situação de vagas disponíveis dos estacionamentos privativos, sugerindo a implantação de um programa para uso de um estacionamento e controle de vagas.

1.3.2 ESPECÍFICOS

a) Buscar artigos científicos em bases de dados disponíveis, fazendo uma revisão literária sobre a temática escolhida
b) Analisar os resultados da busca, avaliando aplicativos existentes para controle de vagas em estacionamentos privativos e seu funcionamento.
c) Desenvolver um programa para um estacionamento privativo para controle de vagas disponíveis.

2. REFERENCIAL TEÓRICO

2.1 Estacionamento

O desafio para encontrar uma vaga para veículos dentro de estacionamentos públicos e privados é muito atual, sendo um problema em vários países do mundo, e por isso inúmeras empresas se preocupam em criar sistemas de controle e automação para realizar este tipo de controle. Um bom estacionamento é a primeira recepção aos clientes e reflete também seu bom atendimento. O estacionamento, geralmente, é o primeiro estágio ao qual o cliente tem acesso, possuindo impacto na opinião dele em relação ao atendimento da empresa (BANDEIRA et al, 2014 apud CHAVES, 2010).

Portanto, fornecer um bom estacionamento dentro para seus clientes, é o primeiro passo para que ele tenha uma impressão positiva sobre seu negócio, antes mesmo de ter acesso ao produto ou serviço em si, visto que a dificuldade em se chegar ao seu estabelecimento é um critério a ser considerado pelo usuário que muitas vezes deixa de usufruir de serviços em determinados locais, justamente pela dificuldade ou falta de bom acesso.

SOARES (2017, online) fala sobre as vagas disponíveis na cidade de São Paulo – SP, sendo

> Circulam nas ruas por dia cerca de 3,8 milhões de carros, enquanto a cidade oferece pouco mais de 500 000 vagas em estacionamentos privados, segundo cálculos do Sindepark, o sindicato que representa as empresas da categoria. Se toda a turma decidisse parar ao mesmo tempo nesses locais, teríamos uma disputa de quase oito veículos por vaga. O exercício matemático livre dá uma ideia de quanto é dramática a busca por esse tipo de espaço na capital.

Estacionamento é o conjunto de baias designadas para abrigo de veículos parados, por um determinado período de tempo, em um local dentro da área urbana. Os estacionamentos são construídos para o uso temporário das vagas, e não permanente. Há dois tipos de estacionamento: privado e público. O estacionamento privado é aquele que só pode ser utilizado em caráter particular. O estacionamento público pode ser utilizado por qualquer automóvel (BRASILEIRO, 2014).

O Código de Trânsito Brasileiro em seu anexo I, tem as seguintes definições:

a) Parada - imobilização do veículo com a finalidade e pelo tempo estritamente necessário para efetuar embarque ou desembarque de passageiros.

b) Estacionamento - imobilização de veículos por tempo superior ao necessário para embarque ou desembarque de passageiros.

2.2 Aplicativos de Estacionamento

Neste interim, algumas soluções estão sendo criadas para facilitar a vida no transito em várias cidades do Brasil. A tecnologia tem invadido várias áreas e a da mobilidade urbana não é diferente.

Alguns aplicativos de celular são até gratuitos e mapeiam a região onde o motorista está e informam quais os preços cobrados para estacionar o carro. Além disso, a maioria coloca o horário de funcionamento, sendo que alguns ainda reservam a vaga para o usuário.

Um destes aplicativos é o Lets Park, que mostra sua localização e quais os estacionamentos mais próximos do lugar onde o motorista está. Cada estacionamento é sinalizado no aplicativo por um balão, que pode ser verde, azul ou cinza. Os verdes indicam as garagens que estão abertas, os azuis mostram quais estão fechadas naquele horário e os cinzas indicam os estacionamentos que ainda estão sem informações sobre horário de funcionamento. No botão no centro da tela, o motorista pode escolher por quantas horas quer usar o estacionamento para que o aplicativo indique os valores (KONKERO, 2019).

Outro exemplo é o Parkopedia que está disponível para sistemas operacionais Android e iOS. Com ele, motoristas conseguem localizar os estacionamentos mais próximos e a quantidade de vagas disponíveis para estacionar em garagens privadas. O aplicativo funciona em diferentes cidades e países, contudo, facilita a vida de turistas na hora de procurar vagas sem ter preocupações com o carro estacionado em ruas ou avenidas. O aplicativo também traz os preços disponíveis para as vagas de estacionamento e quais os tipos de pagamentos as garagens aceitam (KONKERO, 2019).

O aplicativo ParkMe segue a mesma linha do aplicativo anterior onde procura vagas para estacionar em diversas cidades, dentro e fora do Brasil. O aplicativo de localização também faz o monitoramento das vagas da Zona Azul – estacionamentos urbanos disponíveis na cidade de São Paulo. O ParkMe mostra os centros de estacionamento disponíveis e faz a liberação dos cartões e talões da Zona Azul (KONKERO, 2019).

Outra modalidade de ajuda tecnológica na área dos estacionamentos é o Smart Parking, o estacionamento inteligente que é um formato inovador de estacionamento que vem sendo utilizado como uma boa solução para a mobilidade urbana, ao incluir mais vagas de estacionamento e a facilitar a busca por elas por meio da tecnologia ao alcance das mãos (PAREBEM, 2019).

Surgido em São Francisco, no estado da Califórnia (EUA), e adotado por diversas cidades ao redor do mundo, o estacionamento inteligente é um sistema que funciona por meio de uma combinação de tecnologias, instaladas no estacionamento, e mesmo diretamente nas vagas e que oferece mais comodidade aos motoristas e melhora o dia a dia nas grandes cidades (PAREBEM, 2019).

O sistema já foi implantado em oito cidades e é uma automação dos processos de Zona Azul. Funciona por meio de etiquetas inteligentes instaladas nos veículos que se comunicam com sensores colocados na superfície do asfalto e conectados a uma central de gerenciamento. A combinação dessas tecnologias permite ao motorista ver um mapa de vagas disponíveis, acessá-las facilmente e ser tarifado automaticamente, conforme o tempo de utilização (PAREBEM, 2019).

O sistema de estacionamento inteligente começou a ser utilizado há poucos anos no Brasil e já está disponível em algumas cidades brasileiras. Estacionar nunca é tarefa simples quando se está em uma grande cidade. Muitas vezes, a disputa por vagas atinge até mesmo os municípios menores (PAREBEM, 2019).

O mercado oferece uma vasta gama de empresas especializadas na gestão de estacionamento, uma delas é o Grupo PareBem que vigora como uma das maiores e melhores gestoras de estacionamentos do Brasil. A empresa conta com a mais avançada tecnologia para implementar as soluções necessárias para que o estacionamento inteligente seja aderido pelas

cidades, disponibilizando tecnologia própria e know-how, além de treinamento para que tudo ocorra perfeitamente (PAREBEM, 2019).

2.3 Mobilidade urbana aplicada à tecnologia e informática

Muitos estudos tem surgido atualmente sobre o tema e neste tópico será exposto alguns dos diferentes resultados já obtidos no Brasil e no mundo.

Neto e Boaventura (2015) realizaram um estudo em Uberlândia (MG), sobre a inclusão da funcionalidade Busca de Vagas Disponíveis nos aplicativos de estacionamento que rodam nos *smartphones*, aproveitando a capacidade dos mesmos de geolocalização e permitindo desta forma que os usuários do sistema Zona Azul economizem tempo na busca por vagas disponíveis, tanto públicas quanto em estacionamentos privados. A implementação desta funcionalidade consistiu em consultar uma base de dados que contém informações das áreas de estacionamento Zona Azul (coordenadas, vagas totais, vagas disponíveis etc) por meio de web services e, através das APIs do Google Maps e renderizar as informações coletadas no mapa que será apresentado aos usuários.

Concluíram que a arquitetura proposta para a nova funcionalidade levou em consideração a simplicidade no desenvolvimento, tecnologias Open Source e a facilidade de integração com os aplicativos existentes (NETO e BOAVENTURA, 2015).

Silva (2018) elaborou um projeto que foi desenvolvido em duas partes: na primeira, utilizou-se o microcontrolador Arduino para atualizar o banco de dados do estacionamento e, na segunda, utilizou-se a tecnologia Android para realizar a comunicação entre o usuário do aplicativo e os estacionamentos cadastrados. A ideia do projeto é diminuir o tempo gasto na busca de vagas, possibilitando ao usuário procurar um estacionamento, remotamente, por meio do aplicativo, antes mesmo de sair de casa.

Neste mesmo sentido, Santos et al (2018) pensaram em desenvolver um aplicativo baseado em tecnologias embarcadas em *smartphones*, em que, de um lado o cliente interessado pudesse escolher o estacionamento e o período de tempo necessário; e de outro lado, o prestador do serviço pudesse aceitar ou não a requisição da referida vaga. A ideia é que o aplicativo trabalhe em um plataforma simples e intuitiva, onde os usuários terão uma interação amigável e de fácil assimilação. O usuário poderá usar as contas já existentes em seu smartphone (Facebook, E-mail, Google, Twitter) na realização do primeiro acesso, posteriormente acrescentando o CPF e os dados bancários somente no ato da reserva da vaga.

Orrie et al (2015) propõem um sistema sem fio para localização de vagas de estacionamento que para tal faz uso de uma rede de sensores sem fio que verificam se um local de estacionamento

está ocupado ou vago. O sistema proposto consiste de uma aplicação para *smartphones* e uma rede de sensores *wireless*.

Sulamain et al (2013) cita que maioria dos visitantes de prédios comerciais podem gastar de 30 a 45 minutos apenas para encontrar um espaço de estacionamento vazio. Na tecnologia mais recente, algum sistema de estacionamento pode oferecer um sistema que podia contar automaticamente quando o carro entra no espaço vazio e bloqueia um sinal infravermelho, notificando o sistema para contar. No entanto, esse tipo de sensor realmente tem um custo muito alto para instalar e manter. Portanto os autores desenvolveram um projeto, fornecendo uma solução econômica usando a tecnologia Zigbee na tecnologia de sistema de estacionamento. Em vez de usar e manter cabos que precisam ser instalados no teto do estacionamento, desenvolveram um sistema que usa a tecnologia sem fio do Zigbee e que pode notificar os visitantes sobre estacionamento vazio e não vazio.

Barriga et al (2019) contribuem dizendo que as soluções de estacionamento inteligentes têm uma arquitetura definida com componentes específicos (sensores, protocolos de comunicação e soluções de software). Embora haja apenas três componentes que, para compor uma solução inteligente de estacionamento, é importante mencionar que cada componente tem muitos tipos que podem ser usados na implantação dessas soluções. Seu estudo identifica os tipos mais utilizados de cada componente e destaca as tendências de uso no período de análise estabelecido.

Com base na literatura revisada, existem algumas soluções em que as tecnologias LPWAN são usada para implementações de estacionamento inteligente, mas os trabalhos de pesquisa mais revisados pelos autores usaram o LR-WPAN, especificamente o ZigBee. Existem vários tipos de sensores que podem implantar em uma solução de estacionamento inteligente. A pesquisa identificou que, em termos de vantagens tecnológicas, existem quatro características que devem ser consideradas, são: invasão, facilidade de instalação, sensores por slot e autonomia de detecção (BARRIGA et al, 2019).

3. FERRAMENTAS DO PROJETO

A produção deste trabalho terá como base os seguintes aplicativos e softwares.

3.1 Visual Studio 2017 Community

Visual Studio Community é um ambiente de desenvolvimento, IDE, gratuita e extensível para a criação de sistemas para Android, IOS e Windows, além de sistemas web e serviços de nuvem. Os recursos do Xamarin integrados ao Visual 17 Studio possibilitam o desenvolvimento

integrado de sistemas móveis (VISUAL STUDIO, 2017).

Esse ambiente possibilita o desenvolvimento em várias linguagens de programação para web como: ASP.NET, Node.js, Python e JavaScript. Além de frameworks e estruturas como AngularJS, jQuery, Bootstrap, Django e Backbone.js. O suporte multilíngue abrange C#, Visual Basic, F#, C++, JavaScript, TypeScript, Python e outras (VISUAL STUDIO, 2017).

3.2 C#

C# (pronunciado "C sharp") é uma linguagem de programação que pode ser utilizada para criar uma variedade de sistemas executados no .NET Framework. C# é fortemente tipada e orientada a objeto. C# permite o desenvolvimento rápido de sistemas mantendo a expressividade e a elegância das linguagens de estilo C (MICROSOFT, 2017).

3.3 SQL Server Express

O Microsoft SQL Server é um Sistema Gerenciador de Banco de Dados (SGBD) relacional desenvolvido pela Microsoft. O Microsoft SQL Server fornece suporte a dados estruturados e não estruturados armazenando dados corporativos com suporte nativo para dados relacionais, XML e dados espaciais. Esse SGBD também possibilita adicionar informações geográficas aos aplicativos de negócios e criar aplicativos com geolocalização. A granularidade dos dados temporais é aumentada com dados do tipo date e time (DATABASE DESIGN RESORCE, 2017).

3.4 JSON(Java Script Object Notation)

A notação JSON possibilita a composição de artefatos com estruturação leve e simplificada. Ao compará-lo com XML, e notória a vantagem de que em JSON não ocorrem redundâncias de *tags*, tornando menor o volume de dados e, consequentemente, reduzindo a quantidade de memória necessária para o seu armazenamento, manipulação e transmissão (FLORES, 2019).

Por possibilitar instanciar documentos menores, com estruturas simples e de fácil leitura, a notação JSON fora amplamente adotada, principalmente como formato escolhido para a transmissão de dados entre aplicações via Internet. O JSON é uma notação para a composição de documentos textuais contendo dados estruturados. A notação provém dos *objectliterals* do *Java Script*, sua sintaxe é formada basicamente por uma estrutura hierárquica composta pelos artefatos estruturais de vetores, matrizes e objetos, e pares de elementos nome e valor, estes dos tipos primários de cadeias de caracteres, números, booleanos e nulos (FLORES, 2019).

Assim como em *Java Script*, JSON representa os valores de forma simples, atribuindo a eles nomes que os identificam. Embora se assemelhe com a sintaxe de *objectliterals* do *Java Script*, JSON é independente de linguagem de programação. A estruturação de um documento na notação JSON é composta de forma hierárquica por uma sequência de vetores, matrizes e objetos. As matrizes são estruturas multidimensionais delimitadas pela abertura e fechamento de colchetes, sendo seus elementos separados por vírgulas. Tal como ocorre com as matrizes, os vetores são delimitados por colchetes e seus elementos também separados por vírgulas, porém os vetores são estruturas unidimensionais. Já os objetos são delimitados pela abertura e fechamento de chaves, admitem múltiplos pares de elementos nome e valor, assim como matrizes e até mesmo outros objetos (FLORES, 2019).

3.5 Android

Sistema Operacional Móvel desenvolvido pela Google, baseado no kernel Linux e projetado principalmente para dispositivos móveis sensíveis ao toque, como *smartphones* e tablets.

A interface do usuário do Android é baseada, principalmente, na manipulação direta, fazendo uso de gestos que correspondem vagamente a ações do mundo real para manipular objetos na tela e também de um teclado virtual para entrada de texto. Além de celulares e tablets, o Google desenvolveu ainda o Android TV para televisores, o Android Auto para carros e o Android Wear para "dispositivos vestíveis"(como *smartwatches*), cada um com uma interface de usuário específica. Em 2003, na cidade de Palo Alto, na Califórnia foi fundada por Andy Rubin, Rich Miner, Nick Sears e Chris White a Android Inc (PRADO, 2017). A companhia se concentrou na evolução de código aberto de muitas das ideias que começaram na Danger (antiga empresa de Rubin), almejando oferecer a melhor experiência conectada à Web possível para usuários *mobile* e criando um ambiente em que qualquer desenvolvedor pudesse colaborar (DOBIE et al.,2019).

3.6 MVC (Model View Controller)

MVC é o acrônimo de Model-View-Controller (em português, Modelo-Visão Controle) e que, como o próprio nome supõe, separa as camadas de uma aplicação em diferentes níveis. MVC não é um padrão de projeto mas sim um projeto de arquitetura de software. Existam outros padrões de arquitetura de software – como Pipeline, Blackboard, Microkernel e Reflection – o MVC é um dos mais difundidos e utilizados pelos desenvolvedores principalmente pela funcionalidade e objetividade. O MVC define a divisão de uma aplicação em três componentes: Modelo, Visão e Controle. Cada um destes componentes tem uma função específica e estão conectados entre si. O

objetivo é separar a arquitetura do software para facilitar a compreensão e a manutenção (DA SILVA, 2019).

3.7 Asp.net

O ASP.NET é uma tecnologia de desenvolvimento consolidada no mercado já há alguns anos. Essa tecnologia se divide em dois grandes subtipos: o ASP.NET Web Forms e o ASP.NET MVC. Esse último é o grande foco do artigo. Essa tecnologia, basicamente, faz uso de um padrão de design, o MVC, implementado na forma de um framework pela Microsoft. É esse Framework, com todos os seus recursos, o grande responsável pela criação de excelentes aplicações web utilizando o padrão MVC manutenção (DA SILVA, 2019).

O padrão MVC é um padrão responsável pela apresentação da aplicação. Basicamente, ele visa criar um código que não possui uma conexão forte entre as partes (loosely coupled). Essa é uma das bases do desenvolvimento de código atual, e facilita muito a manutenção e adição de funcionalidades ao código posteriormente. Dentro do padrão MVC, há três elementos principais: o *model*, responsável por representar as entidades da lógica de negócios da aplicação; a *view*, responsável por apresentar uma interface para o usuário; e o *controller*, que realiza o controle dos outros elementos, fornecendo uma ligação entre eles (DA SILVA, 2019).

3.8 Google Maps API (Geolocalização)

O Google Maps é um mapa desenvolvido e fornecido pela empresa Google que é utilizado para visualização do planeta terra via satélite de forma gratuita e pública, onde qualquer usuário pode encontrar lugares no mundo, verificar rotas de uma cidade para outra e principalmente, se localizar no globo, tudo através da tecnologia GPS. É possível também que usuários com uma conta Google possam criar mapas privados, marcando pontos de interesse e fazendo anotações (SOUSA, 2016).

O Google Maps possui uma API de desenvolvimento que é um serviço público, onde qualquer desenvolvedor pode utilizar em seus sites ou aplicações, desde que não seja cobrada nenhuma taxa para tal uso, nesses casos, é preciso adquirir uma versão paga dessa API. De modo geral, a Google Maps API possui diversas APIs que possuem especificações diferentes, aumentando as possibilidades de desenvolvimento e as possibilidades de aplicação (SOUSA, 2016).

3.9 Github

GitHub é um website que fornece serviço de hospedagem online e gerenciamento de código fonte, bem como controle de versão distribuído aos usuários. Atualmente, o GitHub possui mais de 53 milhões de repositórios (Fevereiro 2017) e 14 milhões de usuários (Abril 2016)(BATISTA, 2017).

Muitos estudos utilizam os dados do GitHub para analisar sentimento, estudar a correlação entre as mensagens dos commitse a tendência a defeitos nos projetos de Java, mostrar práticas típicas que programadores usam para lidar com exceções, entre outros (BATISTA, 2017).

3.10 Entity Framework

O Entity Framework é um conjunto de tecnologias no ADO.NET que dão suporte ao desenvolvimento de aplicativos de software orientados a dados. Os arquitetos e desenvolvedores de aplicativos orientados a dados lutam com a necessidade de realizar dois objetivos muito diferentes.

Precisam modelar as entidades, as relações e a lógica dos problemas de negócios que estão solucionando e também precisam trabalhar com os mecanismos de dados usados para armazenar e recuperar os dados. Os dados podem se estender por vários sistemas de armazenamento, cada um com seus próprios protocolos (DA SILVA, 2019).

3.11 Postman

Uma aplicação que permite realizar requisições HTTP a partir de uma interface simples e intuitiva, facilitando o teste e depuração de serviços REST. O Postman está disponível como uma aplicação para o browser Google Chrome e possui diversas funcionalidades úteis no desenvolvimento de projetos (DA SILVA, 2019).

3.12 Somee.com

Em linguagens de programação Web que executam no lado servidor, como ASP (*Active Server Pages),* ASP.NET, PHP, JSP e outras, é necessário um servidor Web dedicado à execução destes scrips, e o Somee.com permite a hospedagem do *website* e sistemas *web* baseados em ASP, ASP.NET e PHP com banco de dados SQL Server (BATISTA, 2017).

4 METODOLOGIA

Este trabalho tem cunho técnológico, se baseando em uma pesquisa do tipo qualitativa, visto que a pesquisa responde a questões muito particulares. Ela se preocupa com um nível de realidade que não pode ser quantificado.

Para Minayo e Sanchez (1994, p.37):

> A investigação qualitativa trabalha com valores, crenças, hábitos, atitudes, representações, opiniões e adequa-se a aprofundar a complexidade de fatos e processos particulares e específicos a indivíduos e grupos. A abordagem qualitativa é empregada, portanto, para a compreensão de fenômenos caracterizados por um alto grau de complexidade interna.

A partir desse estudo, será aplicado um questionário, a fim de embasar e complementar as observações realizadas a criação do programa, seguiu os itens abaixo relacionados:

a) Levantamento de informações e dados

Para encontrar o problema foi pesquisado sistemas semelhantes em algumas cidades do Brasil e do mundo e pesquisa em artigos em bases de dados online. Isto ajudou a definir quais itens fariam parte do programa e que funcionalidades o sistema terá.

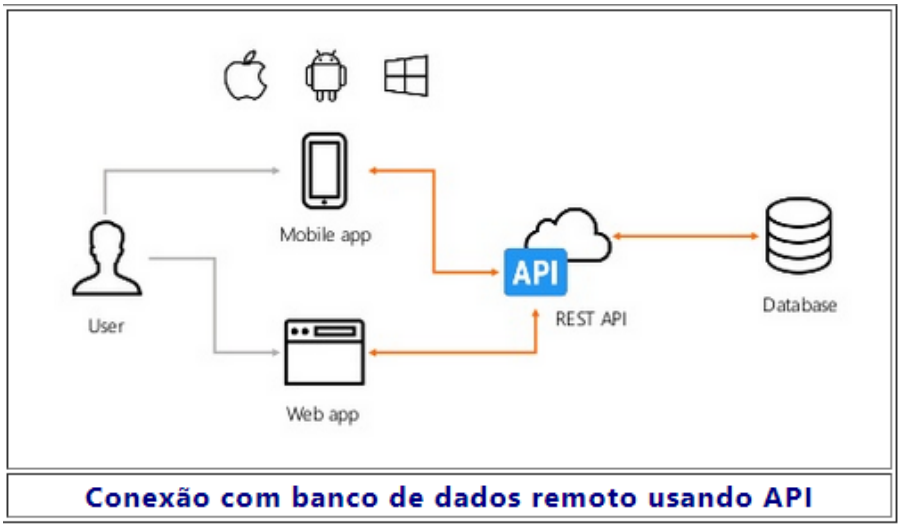
b) Implementação

A codificação do sistema foi realizada utilizando tecnologias e conceitos constantes sem realizar planos de testes formais. Apenas foram testados os itens com a finalidade de verificar erros dentro das funcionalidades definidas previamente.

5 PROJETO

Através da aplicação Web os estacionamentos com as devidas Geolocalização serão cadastradas.

Figura 1 – Conexão com banco de dados



Fonte: Autor (2020)

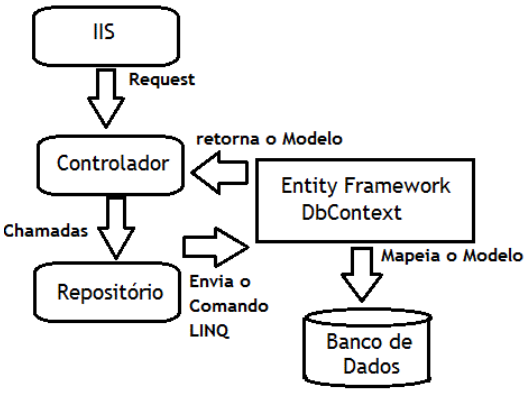
Fluxograma do processo da aplicação:

a) Web app: Usuário vai cadastrar na aplicação Web que após o cadastro.
b) REST API : Essa Api disponilizará a localização e os estacionamentos livre e disponível.
c) Mobile App: Os usuários poderão utilizar o celular para buscas desses estacionamentos, e fazer o ckech in.

5.1 Web APP

Utiliza-se para desenvolvimento da aplicação Web Asp.net MVC, este recurso após usuário colocar os dados do estacionamento através do endereço irá consultar api de endereço do Google para saber a questão da geolocalização do estabelecimento. O MVC é nada mais que um padrão de arquitetura de software, separando sua aplicação em 3 camadas. A camada de interação do usuário(*view*), a camada de manipulação dos dados(*model*) e a camada de controle(*controller*).

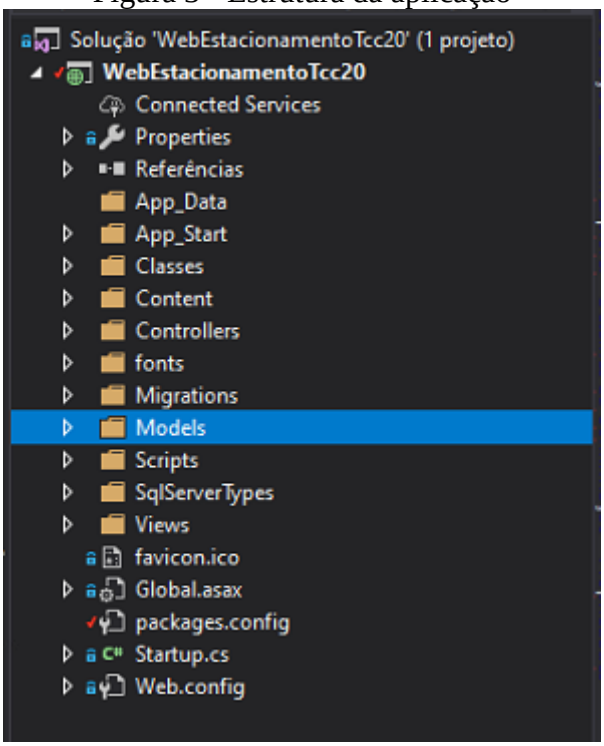
Figura 2 – Desenvolvimento aplicação Web Asp.net



Fonte: Autor (2020)

A estrutura do MVC seguirá o exemplo abaixo na aplicação na parte do cadastro do Usuário. Interessante entram verificar os detalhes no conceito básico.

Figura 3 - Estrutura da aplicação

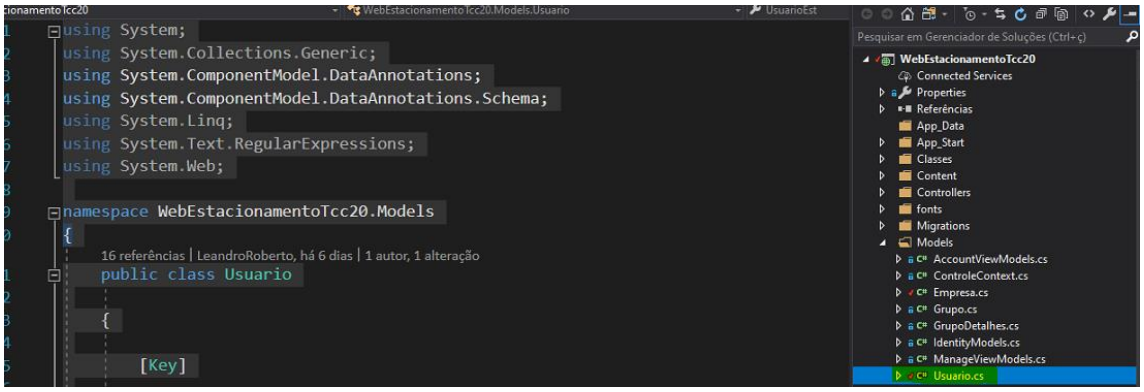


Fonte: Autor (2020)

5.2 Programação

Abaixo apresenta-se a programação e seus códigos para criação da Classe de usuário (Apêndice 1)

Figura 4 – Classe do usuário

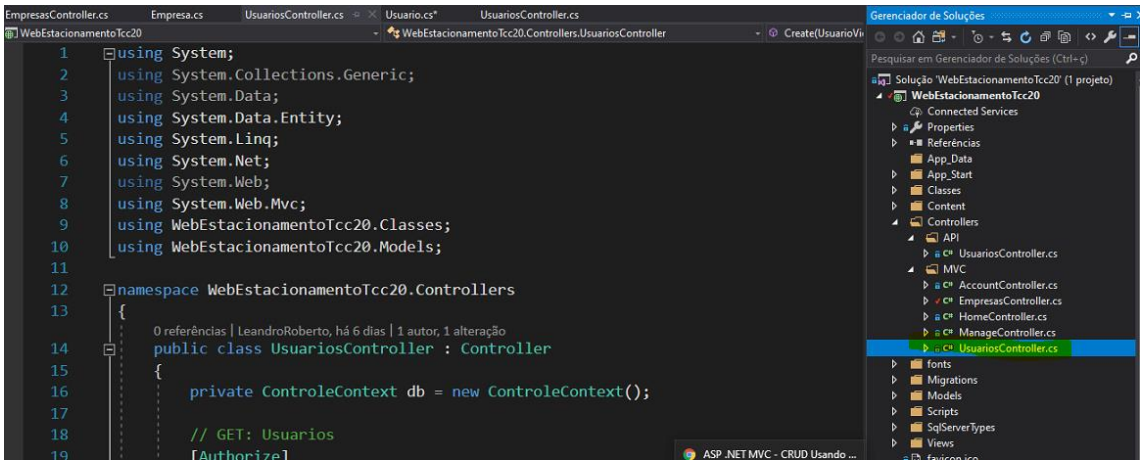


Fonte: Autor (2020)

5.3 Controller

A Controladora (*controller*), como o nome já sugere, é responsável por controlar todo o fluxo de informação que passa pelo site/sistema. É na controladora que se decide “se”, “o que”, “quando” e “onde” deve funcionar. Define quais informações devem ser geradas, quais regras devem ser acionadas e para onde as informações devem ir, é na controladora que essas operações devem ser executadas. Em resumo, é a controladora que executa uma regra de negócio (modelo) e repassa a informação para a visualização (visão). (Apêndice 2)

Figura 5 - Controller



Fonte: Autor (2020)

As funções do sistema são:

1. Create
2. Delete
3. Update
4. Edit

a. Dentro da aplicação tem-se na aplicação web as devidas funções, estas são criadas através MVC Web e elas podem ser alteradas conforme a necessidade. ActionResult Index..: retorna os detalhes da lista dos usuários cadastrados

b. ActionResult Detalhes: retorna os detalhes do usuário ActionResult Create: cria o usuário com detalhes que foram fornecidos na tela de cadastro

c. ActionResult Edit : Edita o cadastro os usuários através do código que foi ActionResult

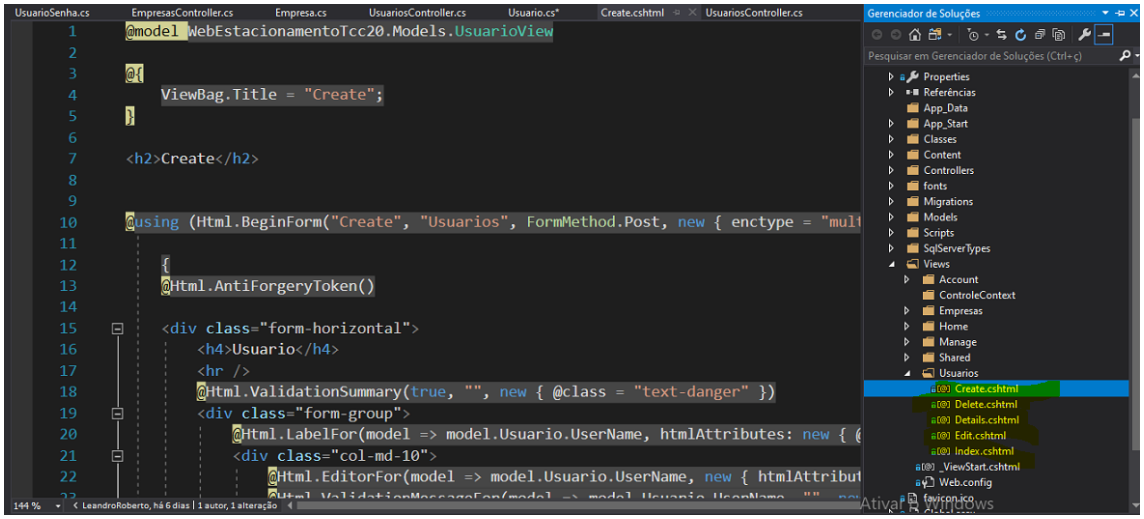
d. Delete: Deleta através do ID do usuário

5.4 View

Este é o início da captura de todas as funções para iniciar os processos internos da aplicação. No sistema terá um VIEW para cada entidade da aplicação, ou seja para cada cadastro de Usuario. (Apêndice 4)

Abaixo segue.

Figura 6 - View

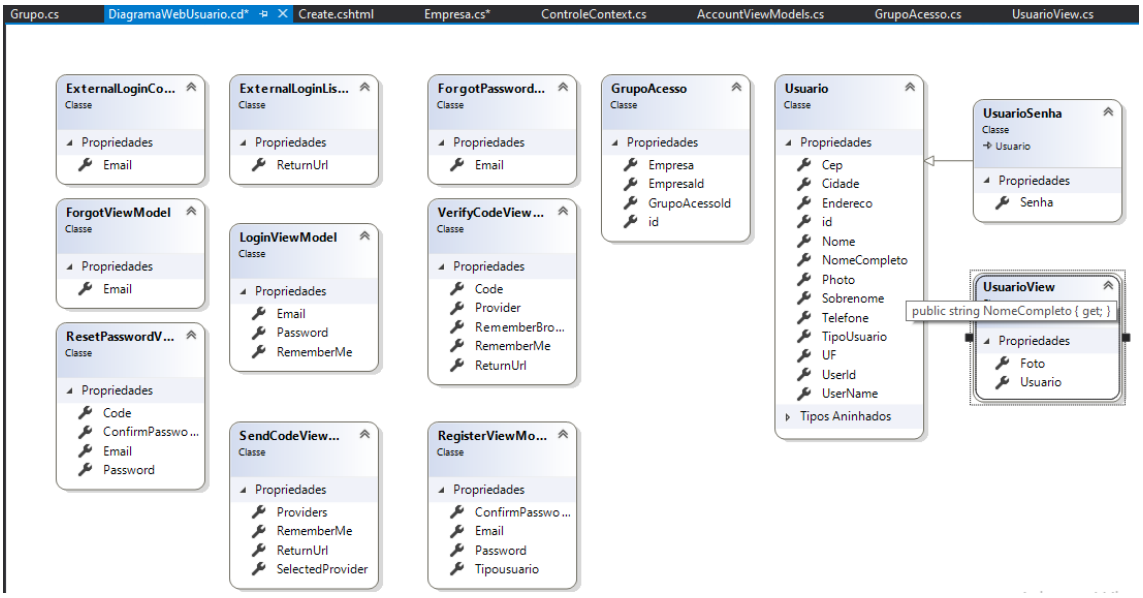


Fonte: Autor (2020)

5.5 Diagrama de Classe Acesso da Aplicação Web

Diagrama com toda a estrutura da soluçao Web, criaçao do registro do usuário, criação do registro para a vaga de estacionamento, tabelas de permissão de login, definiçao de grupos de acesso, tickets emitidos.

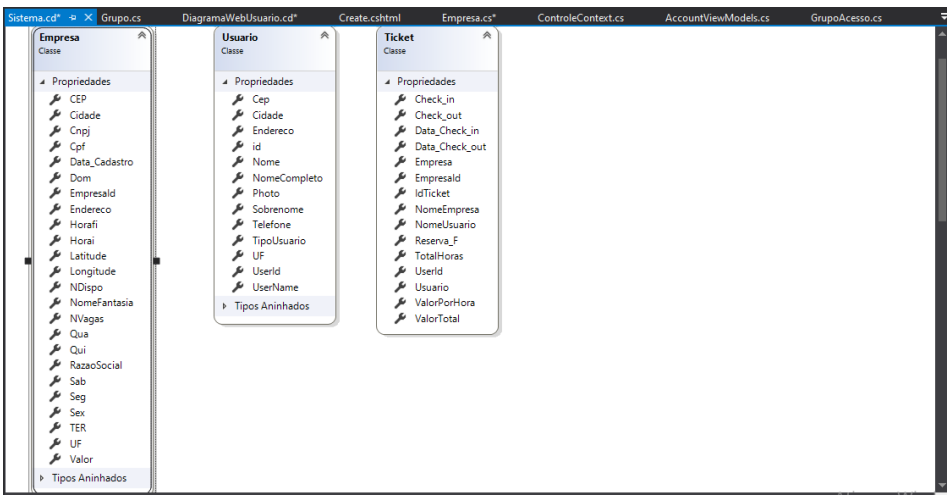
Figura 7 - Diagrama de Classe Acesso da Aplicação Web.



Fonte: Autor (2020)

Diagrama com referências aos registros das classes Empresa, Usuário, Ticket e seus atributos.

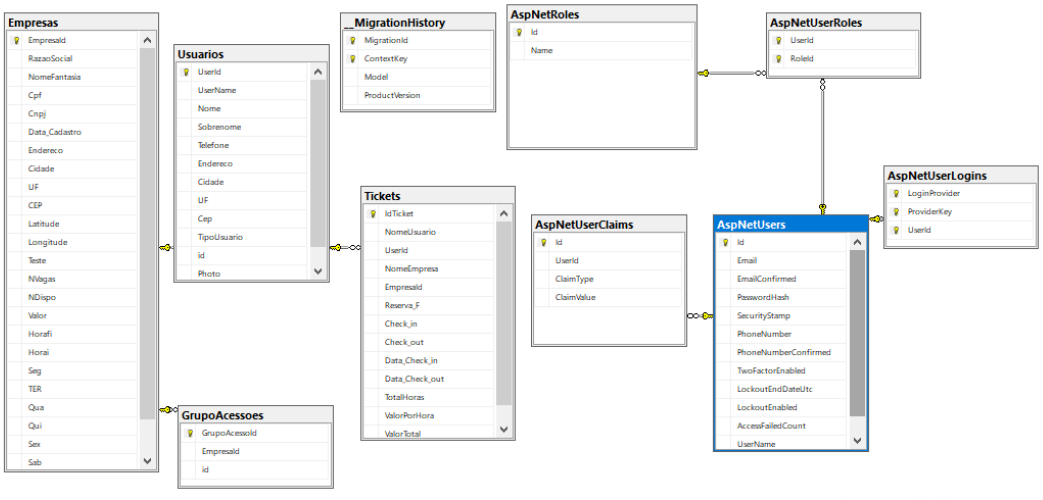
Figura 8 - Diagrama de Classe Sistema da Aplicação Web.



Fonte: Autor (2020)

Diagrama referente a estrutura da parte Web, onde é possivel visualizar os registros do usuário no banco de dados.

Figura 9 - Diagrama do banco de dados.



Fonte: Autor (2020)

5.6 Aplicação ApptccvagaLivre

O aplicativo irá consultar e listar aos usuários cadastrados os estacionamentos que estão, próximos a ele. Através da Api.com acesso pelo link http://www.appestapiloto.somee.com/api/Empresas/Pesquisavaga

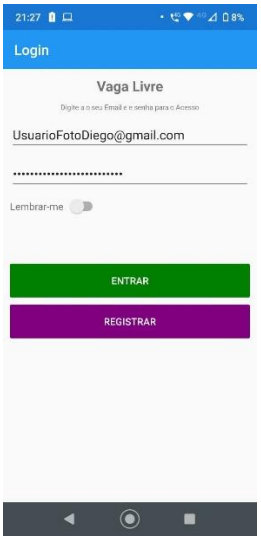
A Aplicação foi desenvolvida em C# Xaramin. Consulta as api que estão na aplicação web para cadastrar consultar usuários e buscar estacionamentos e ela é feita via Postman.

Figura 10 – Consulta Postman

Fonte: Autor (2020)

Aplicativo desenvolvimento em Xaramin multi plataforma faz o mesma consulta através do metodo (Apêndice 4)

Figura 11 – Tela aplicativo



Fonte: Autor (2020)

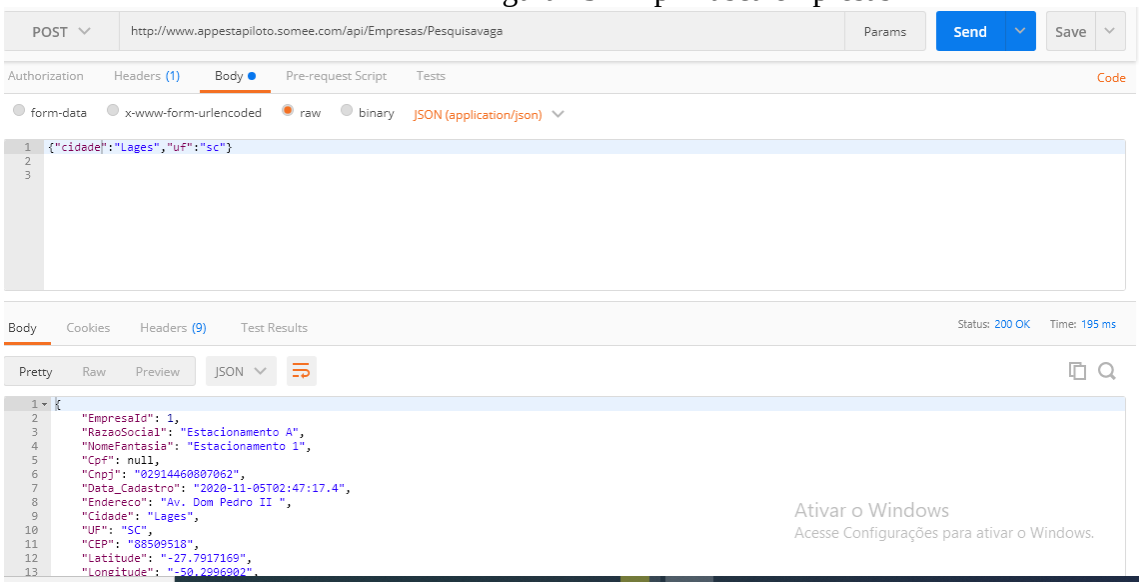
Após aplicação logar com sucesso ela retorna a tela de busca. O usuário deve inserir o nome da cidade e UF.

Figura 12 – Tela Busca Aplicativo

Fonte: Autor (2020)

A consulta é feita através da api busca empresas:

Figura 13 – Api Busca empresas



Fonte: Autor (2020)

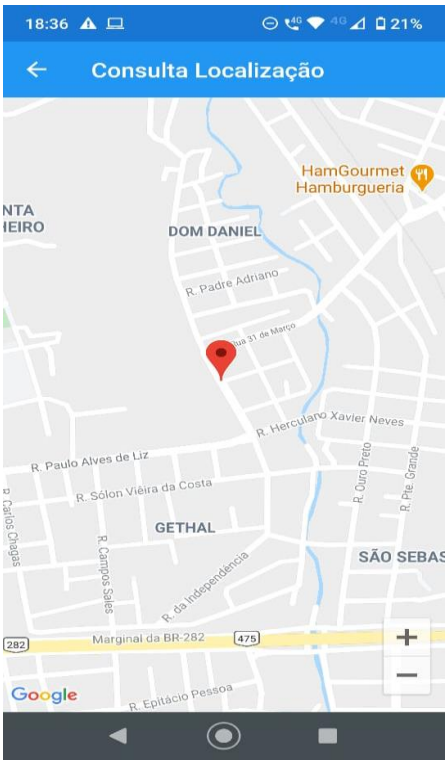
Figura 14– Tela Início App

Fonte: Autor (2020)

Xaramin faz a consulta na api através do código. (Apêndice 5) e também a chamada faz a consulta no mapa. (Apêndice 6)

O aplicativo trabalha com o recurso de geolocalização, permitindo que o usuário visualize sua posição no mapa. (Figura 15).

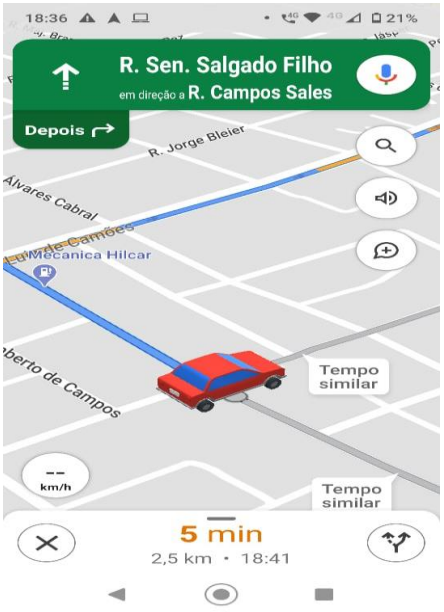
Figura 15– Geolocalização no aplicativo



Fonte: Autor (2020)

Além disto é possível verificar a rota para se chegar ao estacionamento mais próximo conforme verifica-se na Figura 16.

Figura 16 – Geolocalização no aplicativo



Fonte: Autor (2020)

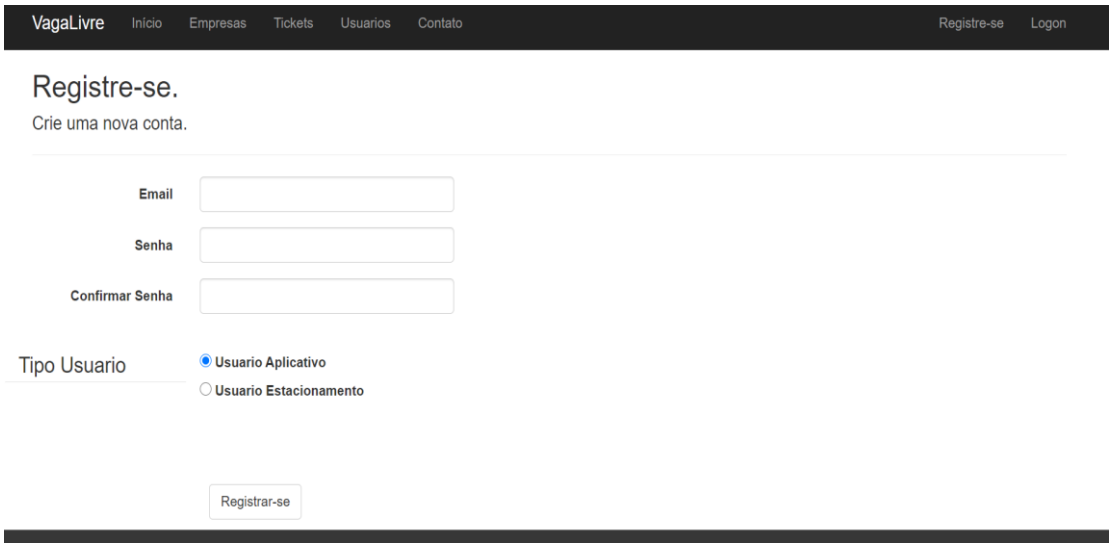
O usuário ao fazer seu cadastro seja como usuário, seja como empresa (estacionamento), terá a visualização da seguinte tela:

Figura 17 – Tela inicío Cadastro Site

Fonte: Autor (2020)

Ao entrar com usuário e senha, a pessoa terá acesso às empresas já cadastradas no aplicativo.

Figura 18– Página de registro de Usuário Aplicativo e Usuário Estacionamento



Fonte: Autor (2020)

Figura 19 – Lista de empresas cadastradas no aplicativo

Fonte: Autor (2020)

O usuário poderá cadastrar a própria empresa ou somente fazer seu cadastro de usuário para posteriormente fazer a consulta de estacionamento mais próximo a ele para verificar a existência da vaga.

6. CONSIDERAÇÕES FINAIS

Caso o presente trabalho seja aprovado, será disponibilizado o aplicativo de forma gratuita na Play Store para obtenção de feedback sobre o projeto. Após um período de testes e de melhorias com as informações coletadas, o app poderá ser lançado no mercado com intuito comercial.

Nesse sentido, ao instalar este sistema inteligente de estacionamento, é possível ter um maior controle mais eficiente oferecendo dessa forma uma maior democratização de uso das vagas. Ou seja, com isso mais motoristas terão a oportunidade de estacionar seus veículos. Reduzindo a falta de vagas em locais públicos do município.

O aplicativo trata-se de um protótipo, portanto correções serão feitas e atualizadas em próximas versões.

REFERÊNCIAS

BARRIGA, Jhonattan J. et al. Smart Parking: A Literature Review from the Technological Perspective. **Applied Sciences**, v. 9, n. 21, p. 4569, 2019.

BATISTA, Natércia A. et al. **GitSED: Um Conjunto de Dados com Informações Sociais Baseado no GitHub.**In: SBBD-DatasetShowcase Workshop. 2017. p. 224-233.

BRASIL. Lei n. 9.503, de 23 de setembro de 1997. **Institui o Código de Trânsito Brasileiro.** Disponível em: <http://www.planalto.gov.br/ccivil_03/LEIS/L9503.htm>. Acesso em: 14 março 2020.

BRASILEIRO, Luzenira Alves; DE ASCENÇÃO, Camila Ferreira; ROSIN, Thales Alexandre. Áreas de Estacionamento para Veículos de Carga e Descarga. **Revista Nacional de Gerenciamento de Cidades**, v. 2, n. 10, 2014.

CHAVES, G. C. 2010**. Estacionamento –um negócio da China!** (edo Brasil, dos EUA, da Índia...). G9 Investimentos disponível em: https://www.g9investimentos.com.br/biblioteca/estacionamento-um-negocio-da-china-e-do-brasil-dos-eua-da-indiaAcesso em: 14 março 2020.

DA SILVA, Andrei Santos; DE VARGAS FLORES, Luciano; VIEGAS, Silvio Cesar. **ASP NET MVC E SITES COMERCIAS.**Revista Eletrônica em Gestão e Tecnologia, v. 5, n. 2, p. 88-96, 2019.

DATABASE DESIGN RESORCE. **SQL Server overview**. Disponível em: https://docs.microsoft.com/en-us/sql/relational-databases/databases/resource-database?view=sql-server-2017. Acesso em: 14 março 2020.

DOBIE, A. et al. **Android Central**. 2019. Disponível em: .www.androidcentral.com/ android-pre-history.

FLÔRES, Fabiano Niederauer et al. **Um processo para a geração automatizada de instâncias de esquemas JSON a partir de consultas SQL**. 2019.

KONKERO. **5 aplicativos para economizar com estacionamento**. 2019. Disponível em https://www.konkero.com.br/financas-pessoais/economizar/3-aplicativos-gratuitos-para-economizar-no-estacionamento. Acesso em 14 março 2020.

MINAYO, M. C. S.; SANCHES, O. **Quantitativo-Qualitativo: Oposição ou Complementaridade?** Caderno de Saúde Pública, Rio de Janeiro, 9 (3): p. 239-262, jul/set, 1994.

MICROSOFT. **Visão geral do ASP.NET MVC. 2017b**. Disponível em: https://docs.microsoft.com/pt-br/aspnet/mvc/overview/gettingstarted/introduction/getting-started . Acesso em: 14 março 2020.

NETO, Gumercindo Rodrigues Chaves; BÔAVENTURA, Ricardo Soares. **Geolocalização Mobile Aplicado No Sistema De Estacionamento Zona Azul Em Uberlândia–Um Estudo De Caso**. 2015.

ORRIE, O.; SILVA, B.; HANCKE, G. **A wireless smart parking system**. IECON2015-Yokohama, p. 9–12, 2015.

PARE BEM. **Estacionamento Inteligente**. 2019. Disponível em https://www.parebem.com.br/como-funciona-um-estacionamento-inteligente/. Acesso em 14 março 2020.

PRADO, Sérgio. **Introdução ao funcionamento interno do Android**. 2017.

SANTOS, Marcos et al. **Uma nova maneira de estacionar veículos de passeio em grandes centros urbanos: proposta do aplicativo "minha vaga"**/A new way of parking vehicles in large urban centers: proposal of the application" my vacancy". Brazilian Journal of Development, v. 4, n. 5, p. 1779-1788, 2018.

SILVA, Dhiego José. **Aplicativo para estacionamento**. Universidade Federal do Rio de Janeiro. Curso de Engenharia Eletrônica e de Computação da Escola Politécnica. 2018.

SOARES, Jussara. **Aplicativos que ajudam a encontrar estacionamentos se proliferam.** 2017. Disponível em: https://vejasp.abril.com.br/cidades/aplicativos-ajudam-a-encontrar-vagas-de-estacionamento/. Acesso em 14 março 2020.

SOUSA, Kássio Rômulo Lima. **Sistema colaborativo para criação de roteiros turísticos e Interativos utilizando a API Google Maps**. 2016.

SULAIMAN, HamzahAsyrani et al. Wireless based smart parking system using zigbee. **International Journal of Engineering and Technology**, p. 3282-3300, 2013.

VISUAL STUDIO. **Visual Studio**. Disponível em: https://visualstudio.microsoft.com/pt-br/vs/ . Acesso em: 14 março 2020.

APÊNDICES

Apêndice 1 – Código Classe de usuário

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;

namespace WebEstacionamentoTcc20.Models
{
   public class Usuario

   {

      [Key]

      public int UserId { get; set; } Declaração da Variavel e tipo de dado Inteiro esse campo será criado no banco de dados através da forma que utilizamos o MVC.

      [Display(Name = "E-Mail")]
      [Required(ErrorMessage = "O Campo {0} é Obrigatório!")]
      [StringLength(100, ErrorMessage = "O Campo {0} pode ter no máximo {1} e minimo {2} caracteres", MinimumLength = 7)]

      [DataType(DataType.EmailAddress)]
      [Index("UserNameIndex", IsUnique = true)]

      public string UserName { get; set; }
      [Display(Name = "Nome")]
      [Required(ErrorMessage = " O Campo {0} é Obrigatório!")]
      [StringLength(50, ErrorMessage = " O Campo {0} pode ter no máximo {1} e minimo {2} caracteres ", MinimumLength = 2)]

      public string Nome { get; set; }
      [Display(Name = "Sobrenome")]
      [Required(ErrorMessage = " O Campo {0} é Obrigatório!")]
      [StringLength(50, ErrorMessage = " O Campo {0} pode ter no máximo {1} e minimo {2} caracteres ", MinimumLength = 2)]

      public string Sobrenome { get; set; }
      [Display(Name = "Usuário")]

      public string NomeCompleto { get { return string.Format("{0} {1}", this.Nome, this.Sobrenome); } }
      [Required(ErrorMessage = " O Campo {0} é Obrigatório!")]
      [StringLength(20, ErrorMessage = " O Campo {0} pode ter no máximo {1} e minimo {2} caracteres ", MinimumLength = 7)]

      public string Telefone { get; set; }
      [Required(ErrorMessage = " O Campo {0} é Obrigatório!")]
```

```
        [StringLength(100, ErrorMessage = " O Campo {0} pode ter no máximo {1} e minimo {2} caracteres ", MinimumLength = 10)]

        public string Endereco { get; set; }
        [DataType(DataType.ImageUrl)]
        public string Photo { get; set; }

        [Display(Name = "User app")]
        public bool UsuarioApp { get; set; }

        [Display(Name = "User estacionamento")]
        public bool UsuarioEst { get; set; }
```

Apêndice 2 - Exemplo do Controller da Classe Usuario

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using WebEstacionamentoTcc20.Classes;
using WebEstacionamentoTcc20.Models;

namespace WebEstacionamentoTcc20.Controllers
{
   public class UsuariosController : Controller
   {
      private ControleContext db = new ControleContext();

      // GET: Usuarios
      [Authorize]
      public ActionResult Index()
      {
         return View(db.Usuarios.ToList());
      }

      // GET: Usuarios/Details/5
      public ActionResult Details(int? id)
      {
         if (id == null)
         {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         Usuario usuario = db.Usuarios.Find(id);
         if (usuario == null)
         {
            return HttpNotFound();
         }
         return View(usuario);
      }

      // GET: Usuarios/Create
      public ActionResult Create()
      {
         return View();
      }

      // POST: Usuarios/Create
      // To protect from overposting attacks, please enable the specific properties you want to bind
to, for
      // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
      [HttpPost]
      [ValidateAntiForgeryToken]
      public ActionResult Create(UsuarioView view)
```

```
{
    if (ModelState.IsValid)
    {
        db.Usuarios.Add(view.Usuario);
        try
        {

            if (view.Foto != null)
            {
                var pic = Utilidades.UploadPhoto(view.Foto);
                if (!string.IsNullOrEmpty(pic))
                {
                    view.Usuario.Photo = string.Format("~/Content/Fotos/{0}", pic);
                }
            }

            db.SaveChanges();

            Utilidades.CreateUserASP(view.Usuario.UserName);
            if (view.Usuario.UsuarioApp)
            {
                Utilidades.AddRoleToUser(view.Usuario.UserName, "Usuarioapp");
            }

            if (view.Usuario.UsuarioEst)
            {
                Utilidades.AddRoleToUser(view.Usuario.UserName, "UsuarioEst");
            }

            return RedirectToAction("Index");
        }
        catch (Exception ex)
        {

            ModelState.AddModelError(string.Empty, ex.Message);
        }
    }

    return View(view);
}

// GET: Usuarios/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Usuario usuario = db.Usuarios.Find(id);
    if (usuario == null)
    {
        return HttpNotFound();
    }
```

```
            var view = new UsuarioView
            {
                Usuario = usuario
            };

            return View(view);
        }

        // POST: Usuarios/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind
to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(UsuarioView view)
        {
            if (ModelState.IsValid)
            {

                var db2 = new ControleContext();
                var oldUser = db2.Usuarios.Find(view.Usuario.UserId);
                db2.Dispose();

                if (view.Foto != null)
                {
                    var pic = Utilidades.UploadPhoto(view.Foto);
                    if (!string.IsNullOrEmpty(pic))
                    {
                        view.Usuario.Photo = string.Format("~/Content/Fotos/{0}", pic);
                    }
                }
                else
                {
                    view.Usuario.Photo = oldUser.Photo;
                }

                db.Entry(view.Usuario).State = EntityState.Modified;
                try
                {
                    if (oldUser != null && oldUser.UserName != view.Usuario.UserName)
                    {
                      Utilidades.ChangeEmailUserASP(oldUser.UserName, view.Usuario.UserName);
                    }
                    db.SaveChanges();
                }
                catch (Exception ex)
                {

                    ModelState.AddModelError(string.Empty, ex.Message);
                    return View(view);
                }
                return RedirectToAction("Index");
            }
            return View(view.Usuario);
```

```
        }

        // GET: Usuarios/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Usuario usuario = db.Usuarios.Find(id);
            if (usuario == null)
            {
                return HttpNotFound();
            }
            return View(usuario);
        }

        // POST: Usuarios/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Usuario usuario = db.Usuarios.Find(id);
            db.Usuarios.Remove(usuario);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}


    }
}
```

Apêndice 3 – Código View

a) Create:

```
@model WebEstacionamentoTcc20.Models.UsuarioView

@{
   ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm("Create", "Usuarios", FormMethod.Post, new { enctype =
"multipart/form-data" }))

   {
   @Html.AntiForgeryToken()

   <div class="form-horizontal">
      <h4>Usuario</h4>
      <hr />
      @Html.ValidationSummary(true, "", new { @class = "text-danger" })
      <div class="form-group">
         @Html.LabelFor(model => model.Usuario.UserName, htmlAttributes: new { @class =
"control-label col-md-2" })
         <div class="col-md-10">
            @Html.EditorFor(model => model.Usuario.UserName, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Usuario.UserName, "", new { @class =
"text-danger" })
         </div>
      </div>

      <div class="form-group">
         @Html.LabelFor(model => model.Usuario.Nome, htmlAttributes: new { @class =
"control-label col-md-2" })
         <div class="col-md-10">
            @Html.EditorFor(model => model.Usuario.Nome, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Usuario.Nome, "", new { @class =
"text-danger" })
         </div>
      </div>

      <div class="form-group">
         @Html.LabelFor(model => model.Usuario.Sobrenome, htmlAttributes: new { @class =
"control-label col-md-2" })
         <div class="col-md-10">
            @Html.EditorFor(model => model.Usuario.Sobrenome, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Usuario.Sobrenome, "", new { @class
= "text-danger" })
```

```
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Usuario.Telefone, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Usuario.Telefone, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Usuario.Telefone, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Usuario.Endereco, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Usuario.Endereco, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Usuario.Endereco, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Foto, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                <span class="btn btn-default btn-file">
                    @Html.TextBoxFor(model => model.Foto, new { type = "file" })
                </span>
            </div>
        </div>


        <div class="form-group">
            @Html.LabelFor(model => model.Usuario.UsuarioApp, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                <div class="checkbox">
                    @Html.EditorFor(model => model.Usuario.UsuarioApp)
                    @Html.ValidationMessageFor(model => model.Usuario.UsuarioApp, "", new {
@class = "text-danger" })
                </div>
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Usuario.UsuarioEst, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                <div class="checkbox">
                    @Html.EditorFor(model => model.Usuario.UsuarioEst)
```

```
                @Html.ValidationMessageFor(model => model.Usuario.UsuarioEst, "", new {
@class = "text-danger" })
            </div>
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-default" />
        </div>
    </div>
</div>


}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Apêndice 4 - Desenvolvimento Em Xaramin Multi Plataforma

```
private async void Logar()
      {
         WaitActivityIndicator.IsRunning = true;
         var loginRequest = new LoginRequest
         {
            Email = Emailentry.Text,
            Senha = passwordEntry.Text,
         };

         var jsonRequest = JsonConvert.SerializeObject(loginRequest);
         var httpContent = new StringContent(jsonRequest, Encoding.UTF8, "application/json");
         var resp = string.Empty;

         try
         {
            var client = new HttpClient();
            client.BaseAddress = new Uri("http://www.appestapiloto.somee.com");
           var Url = "/api/Usuarios/Login";


            var result = await client.PostAsync(Url, httpContent);

            await DisplayAlert("Error", client.BaseAddress + Url, "Aceitar 95");

            await DisplayAlert("Error", result.ToString(), "Aceitar 95");

            if (!result.IsSuccessStatusCode)

            {
               await DisplayAlert("Error", "Usuario ou Senha incorretos 89", "Aceitar");
               WaitActivityIndicator.IsRunning = false;
               return;
            }

            resp = await result.Content.ReadAsStringAsync();

         }
         catch (Exception ex)
         {
            await DisplayAlert("Error", ex.Message, "Aceitar 95");
            WaitActivityIndicator.IsRunning = false;
            return;

         }

         var user = JsonConvert.DeserializeObject<User>(resp);
         user.Password = passwordEntry.Text;
         WaitActivityIndicator.IsRunning = false;

         await DisplayAlert("Bem Vindo", user.Nome, "aceita");
         await Navigation.PushAsync(new Imenu(user));
```

Apêndice 5 – Código Xaramin Consulta Na Api

```
    public Imenu(User user)
    {
        InitializeComponent();

        this.Padding = Device.OnPlatform(new
Thickness(10, 20, 10, 10),
new Thickness(10),
new Thickness(10)
 );

        Pins = new ObservableCollection<Pin>();


        this.user = user;
    }
    protected override void OnAppearing()
    {
        base.OnAppearing();

        usernameLabel.Text = this.user.Nome;
        photoImage.Source = this.user.PhotoFullpath;
        photoImage.WidthRequest = 250;
        photoImage.HeightRequest = 250;

        buscarestacionamento.Clicked += Buscarestacionamento_Clicked;

    }

    private async void Buscarestacionamento_Clicked(object sender, EventArgs e)
    {
         if (string.IsNullOrEmpty(cidadeentry.Text))
         {

             await DisplayAlert("Erro", "Digite um nome de cidade", "Aceitar");
             cidadeentry.Focus();
             return;
         }

        ConsultaMapalocalizacao();


 }


    private async void  ConsultaMapalocalizacao()
    {


        WaitActivityIndicator.IsRunning = true;

        var localizacaoRequest = new LocalizacaoRequest
        {
```

```
    cidade = cidadeentry.Text,
    uf =  ufentry.Text,
};

var jsonRequest = JsonConvert.SerializeObject(localizacaoRequest);
var httpContent = new StringContent(jsonRequest, Encoding.UTF8, "application/json");
var resp = string.Empty;

try
{
    var client = new HttpClient();
    client.BaseAddress = new Uri("http://www.appestapiloto.somee.com");
    var Url = "/api/Empresas/Pesquisavaga";
    var result = await client.PostAsync(Url, httpContent);
    await DisplayAlert("Error", client.BaseAddress+Url, "Aceitar 95");

    await DisplayAlert("Error", result.Content.ToString(), "Aceitar 95");

    if (!result.IsSuccessStatusCode)

    {
        WaitActivityIndicator.IsRunning = false;
        await DisplayAlert("Error", result.ToString(), "Aceitar");
        WaitActivityIndicator.IsRunning = false;
        return;
    }

    resp = await result.Content.ReadAsStringAsync();

}
catch (Exception ex)
{
    await DisplayAlert("Error", ex.Message, "Aceitar 95");
    WaitActivityIndicator.IsRunning = false;
    return;

}

var empresa = JsonConvert.DeserializeObject<Empresa>(resp);
empresa.RazaoSocial = ufentry.Text;
WaitActivityIndicator.IsRunning = false;

await DisplayAlert("Bem Vindo", user.Nome, "aceita");
await DisplayAlert("Empresa Encontrada", empresa.NomeFantasia, "aceita");
await DisplayAlert("Latitude", empresa.Latitude, "aceita");
await DisplayAlert("Longetude", empresa.Longitude, "aceita");
await Navigation.PushAsync(new ConsultaMapa(empresa));
```

Apêndice 6 – Código Chamada Consulta Mapa

```
public ConsultaMapa(Empresa empresa)
      {

         InitializeComponent();

         var latitudec = Convert.ToDouble(empresa.Latitude, new CultureInfo("en-US"));
         var longetudec = Convert.ToDouble(empresa.Longitude, new CultureInfo("en-US"));


         var position2 = new Position(latitudec, longetudec);
         Mapa.MoveToRegion(MapSpan.FromCenterAndRadius(
                  new Position(position2.Latitude, position2.Longitude),
                  Distance.FromMiles(0.5)));


         //  var position1 = new Position(-27.7917169,-50.2996902);

         var position1 = new Position(latitudec, longetudec);
         try
         {
            var pin = new Pin
            {
               Type = PinType.Place,
               Position = position1,
               Label = "Nome Estacionamento..: "+empresa.NomeFantasia,
               BindingContext = "Quantidade Vagas..: " + empresa.NDispo,
               Address = "Endereco "+empresa.Endereco,
            };

            Mapa.Pins.Add(pin);


         }
         catch (Exception ex)
         {
            Debug.WriteLine("Unable to get location, may need to increase timeout: " + ex);
         }
      }

      private async void Locator()
      {
         var locator = CrossGeolocator.Current;
         locator.DesiredAccuracy = 50;

         var location = await locator.GetPositionAsync(timeoutMilliseconds: 10000);
         var position = new Position(location.Latitude, location.Longitude);
         Mapa.MoveToRegion(MapSpan.FromCenterAndRadius(position,
Distance.FromMiles(.3)));
      }

   }
}
```