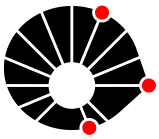

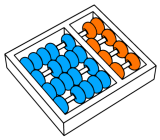

Smart Parking app - um aplicativo móvel para visualização de vagas em Estacionamento Inteligente

V. K. Aoki *L. F. Gonzalez* *J. F. Borin*

Relatório Técnico - IC-PFG-20-13
Projeto Final de Graduação
2020 - Agosto

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Smart Parking app - um aplicativo móvel para visualização de vagas em Estacionamento Inteligente

Vitor Kaoru Aoki* Luis Fernando Gonzalez† Juliana Freitag Borin‡

Resumo

O projeto de estacionamento inteligente da Unicamp, desenvolvido dentro do contexto da iniciativa Smart Campus, utiliza aprendizado de máquina para identificar a quantidade de vagas disponíveis a partir de fotos tiradas dos estacionamentos. O sistema inclui um totem, instalado na entrada do bolsão de estacionamento, para informar aos motoristas o número de vagas disponíveis. Com o objetivo de melhorar e ampliar a disponibilização desta informação à comunidade do campus, este trabalho apresenta o projeto e implementação de um aplicativo móvel capaz de informar, em tempo real e com segurança para o motorista, o número de vagas disponíves nos bolsões de estacionamento mais próximos do destino do usuário.

1 Introdução

A Internet das Coisas (do inglês, *Internet of Things* - IoT) tem sido cada vez mais adotada em soluções de automação e monitoramento no setor produtivo bem como em soluções para facilitar o dia-a-dia das pessoas. O mesmo tem ocorrido com algumas cidades que vêm desenvolvendo projetos com o uso de IoT com o objetivo de aumentar sua produtividade e torná-las mais sustentáveis.

Com a Unicamp esse processo não foi diferente; com a criação do projeto Smart Campus [5] várias soluções baseadas em IoT estão sendo desenvolvidas no campus em uma colaboração entre funcionários, docentes e alunos. Estas soluções visam tornar o ambiente da universidade mais sustentável e melhorar a vida da comunidade que a frequenta diariamente.

Algumas soluções já estão sendo implantadas pelo campus. Entre elas está a solução de Estacionamento Inteligente [3] implantada inicialmente no estacionamento do IC - Instituto de Computação. Por meio da utilização de aprendizado de máquina e IoT, ela se propõem a disponibilizar, em tempo real, a quantidade de vagas de estacionamento existentes. Para tanto, foi utilizada uma câmera que tira fotos do estacionamento e por meio da utilização de um algoritmo de aprendizado de máquina, é identificado o número de vagas livres. Por questões de segurança de dados, essas fotos são tratadas localmente e apenas os dados da

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP
†Konker Labs
‡Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

quantidade de vagas é enviado para a nuvem, que utiliza a plataforma da Konker Labs. Com estes dados coletados, é utilizado um totem na entrada do estacionamento, que mostra aos motoristas o número de vagas livres.

Embora a utilização de totens nas entradas dos bolsões de estacionamento contribua na busca por uma vaga - dado que evita a entrada em um bolsão lotado, ainda não é a solução ideal, pois: a) o motorista só recebe a informação de que não há vagas no estacionamento mais próximo do seu destino ao visualizar o totem e, então, precisa ir em busca de outro bolsão com vagas livres; b) é possível que haja congestionamento nos arredores de bolsões de estacionamento que apresentam alta procura por vagas em horários de pico, como, por exemplo, próximo do horário de início das aulas. Assim sendo, este projeto propõe uma nova interface para a visualização de vagas de Estacionamento inteligente por meio de um aplicativo móvel.

2 Trabalhos relacionados

Durante o processo de levantamento bibliográfico, foi possível encontrar diversos trabalhos relacionados a estacionamentos inteligentes. Dentre eles, encontram-se trabalhos teóricos e produtos vendidos no mercado; ambos com a finalidade de diminuir os problemas (tempo gasto, congestionamentos e poluição gerados) relacionados à procura por vagas de estacionamento.

Uma das aplicações práticas que pode ser vista no dia-a-dia, encontra-se em estacionamentos de shoppings e supermercados. Neles, nas entradas dos estacionamentos, é possível encontrar placares indicando a quantidade de vagas existentes em cada andar, proposta similar à utilizada no projeto de Smart Parking da Unicamp. Outra proposta utilizada por estacionamentos de estabelecimentos, é o uso de LEDs posicionados sobre cada vaga para indicar vagas vazias ocupadas. Ao estacionar, um sensor faz com que a luz fique vermelha e ao deixá-la, a luz volta a ficar verde. Essa alternativa apresenta algumas limitações, como a confusão que pode causar ao condutor do veículo caso existam muitas vagas, por conta da poluição visual causada.

Outra solução encontrada, e a que se assemelha mais ao trabalho aqui realizado, é o aplicativo *Smart Parking (Novas Software)* [8] que consiste em uma plataforma onde é possível encontrar no mapa, estacionamentos cadastrados, com dados de horário de funcionamento, localização e quantidade de vagas disponíveis. Também é possível realizar reservas de vagas e fazer check-in e check-out da vaga, com pagamento pelo uso do estacionamento.

Dentre os trabalhos teóricos estudados, dois foram destacados. O primeiro [1] utilizava um banco de dados MySQL para o armazenamento dos dados enviados por sensores instalados no estacionamento. A interface utilizada pelo usuário consistia em uma página web que permitia ao usuário a reserva de vagas e o pagamento do valor referente ao tempo utilizado no estacionamento. O sistema mostrava ao usuário as vagas livres, as reservadas e as ocupadas. A partir desta informação, era possível escolher a vaga desejada.

O segundo trabalho [2] apresentava proposta parecida ao realizado, pois utilizava um aplicativo Android para a comunicação com o usuário. Assim como o trabalho anterior, o sistema também era responsável por mostrar ao usuário as informações de vagas do es-

tacionamento, e realizar a reserva e liberação das vagas. Porém, diferente do anterior, ele também mostrava ao usuário, um mapa com estacionamentos próximos. No presente trabalho também buscou-se minimizar a necessidade de interação do usuário com o aplicativo como forma de garantir a segurança do motorista. Para tanto, o aplicativo mostra apenas ao usuário a localização do instituto de destino e do estacionamento próximo.

A proposta desse trabalho se assemelha aos trabalhos estudados, no contexto de facilitar ao condutor do veículo a procura por vagas. Porém, diferente das demais, ela não está relacionada a estacionamentos onde é possível reservar vagas, tirando assim a possibilidade de garantia de vagas. Para que ele possua esta informação antes de chegar ao estacionamento desejado, a solução do trabalho se baseou na utilização do "totem virtual", mesclando as propostas de utilização de totens a aplicações mobile, dando ao usuário uma maneira fácil de acesso aos dados atualizados em tempo real.

3 Modelo do sistema

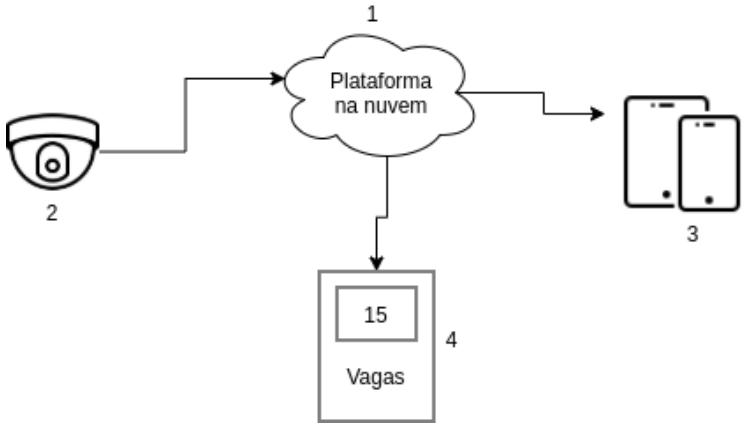

Figura 1: Fluxo de dados da solução Smart Parking [15]

O projeto é baseado na criação de eventos pelo aplicativo, que correspondem a compromissos que os usuários têm na Unicamp. Para cada evento, é possível verificar as informações de vagas nos estacionamentos por meio dos dados de localização armazenados dos institutos e do banco de dados na plataforma na nuvem, com o número de vagas coletado em tempo real.

O fluxo de dados desde a obtenção pelas câmeras, até os dispositivos móveis e totens está representado na Figura 1. As imagens são obtidas pela câmera e a quantidade de vagas é processada localmente, a partir dela (este processo é indicado pelo índice 2 na imagem). Este dado então é enviado para uma plataforma de armazenamento em nuvem (índice 1 na figura). Os dados de vagas são obtidos então pelos dispositivos móveis (índice 3) e pelos totens (índice 4) para que sejam apresentados aos usuários.

A definição de eventos e de estacionamentos próximos utilizada neste projeto é detalhada a seguir.

3.1 Eventos - Únicos e Repetitivos

No dia-a-dia da faculdade diversos tipos de eventos ocorrem. Eles podem ser aulas, reuniões, atividades esporádicas em algum instituto, entre outros. O aplicativo armazena os eventos do usuário em um banco de dados, e assim, ele pode acessá-los. Eles são compostos por um nome do evento, o instituto em que ocorrerá, a data (que para os eventos repetitivos, é o dia da semana e para os eventos únicos é uma data do mês) e a hora.

Os eventos únicos são utilizados pelo sistema como eventos que não ocorrem com frequência, como alguma reunião, ou alguma atividade que possar estar acontecendo em algum instituto. Assim sendo, após ser acessado pelo usuário, ele é apagado do banco de dados pois não voltará a acontecer. Por exemplo, podemos ter um evento com a seguinte configuração: Nome: Palestra sobre IoT, Local: IC - Instituto de Computação, Data: 12/06/2020, Hora: 14:00.

Já os eventos repetitivos são tratados pelo sistema como eventos que ocorrem frequentemente, como aulas para docentes e alunos, o horário de entrada no trabalho para os funcionários, entre outros. Como eles ocorrem regularmente, após serem acessados pelo usuário, ele continua salvo no sistema, a não ser que seja excluído pelo mesmo. Como um exemplo, podemos ter: Nome: Aula de Cálculo 1, Local: IMECC - Instituto de Matemática, Estatística e Computação Científica, Dia da semana: Segunda-feira, Hora: 10:00.

3.2 Estacionamentos Próximos

O sistema armazena informações de latitude e longitude dos estacionamentos de cada instituto. Assim, é possível calcular a distância entre o instituto de destino e os institutos com estacionamentos próximos.

Para o cálculo dessas distâncias foi utilizada a fórmula de Haversine [13] [14], que é utilizada para cálculos de distâncias entre dois pontos, em superfícies esféricas. Como a Terra é uma esfera muito grande, é possível utilizar equações planas para cálculos de distâncias pequenas. Porém, quando essas distâncias aumentam (maiores que 20 km) o resultado começa a apresentar discrepâncias. Assim, a fórmula de Haversine é uma das mais precisas para esses cálculos em esferas, utilizando como parâmetros a latitude e longitude dos pontos. A fórmula é proveniente da função seno ao quadrado, como indicado a seguir:

$$Haversine(\theta) = sin^2(\theta) \tag{1}$$

Para que sejam utilizadas as informações de latitude e longitude, a fórmula pode ser reescrita da seguinte maneira:

$$\begin{gathered} a = sin^2(\Delta\phi/2) + cos(\phi 1) * cos(\phi 2) * sin^2(\Delta\lambda/2) \\ c = 2 * atan2(\sqrt{a}, \sqrt{(1-a)}) \\ d = R * c \end{gathered} \tag{2}$$

onde $\phi 1$ é a latitude em radianos do ponto 1, $\phi 2$ é a latitude em radianos do ponto 2, $\Delta\phi$ a diferença entre $\phi 2$ e $\phi 1$; $\Delta\lambda$ é a diferença em radianos entre as longitudes dos dois pontos ($\lambda 2$ e $\lambda 1$) e R o raio da Terra em metros.

Desse modo, para este projeto a equação (2) foi utilizada para o cálculo das distâncias, sendo d (2), o resultado final. Considerando um possível deslocamento do condutor do veículo, do estacionamento de um instituto até o instituto de destino, tomamos como 300 m a distância ideal para recomendação dos estacionamentos próximos.

4 Metodologia

A partir de análises de trabalhos já realizados, a ideia inicial era a de utilizar uma forma gráfica de apresentar as vagas existentes, assim como no trabalho de Santos e Passos [2]. Porém, ao analisarmos o uso do aplicativo no contexto da universidade, a ideia foi modificada para uma forma mais simples de visualização, pois seria utilizada durante a condução do veículo e não poderia comprometer a segurança do usuário.

Com isso, a primeira ideia para informar ao condutor a informação de vagas, foi a utilização de notificações. No aplicativo, antes de seu compromisso, o usuário deveria criar um evento, que ficaria armazenado em seu aparelho. A notificação então seria disparada 5 minutos antes do evento ocorrer mostrando ao usuário, a quantidade de vagas existentes no estacionamento mais próximo.

Porém, a solução apresentou algumas dificuldades que resultaram na escolha de outra forma, que melhor se adequaria ao projeto, para mostrar as informações. A primeira delas foi a de que dispositivos Android, por conta das modificações feitas pelas diversas empresas que produzem aparelhos com o sistema, podem apresentar como configuração padrão o bloqueio de notificações de aplicativos instalados. Desse modo, todos os usuários teriam que alterar as configurações de seus aparelhos para que pudessem receber as notificações, o que fugia da proposta do trabalho, causando uma dificuldade a mais para a utilização pelo usuário. A segunda foi a de que a utilização de notificações limitava a informação em tempo real, já que somente uma única notificação seria enviada ao usuário, podendo haver, dentro desse período de 5 minutos até o evento ocorrer, modificações na quantidade de vagas.

Como consequência, a proposta de utilização de notificações foi descartada, dando lugar ao uso dos "totens virtuais"que mostram em tempo real a quantidade de vagas em cada estacionamento de institutos próximos ao instituto de destino.

4.1 Tecnologias utilizadas

Para o desenvolvimento do projeto, foram utilizadas tecnologias de IoT para a obtenção e armazenamento dos dados de vagas de estacionamento, por meio da plataforma da Konker Labs e também tecnologias de desenvolvimento de aplicações mobile, no caso do projeto, a plataforma Android. As subseções seguintes apresentam as tecnologias utilizadas.

4.1.1 Konker Labs

A plataforma desenvolvida pela Konker Labs se encarrega de toda a estrutura relacionada à "Internet"em *Internet of Things*. Por meio de uma plataforma em nuvem, ela é responsável por todo o processo de coleta dos dados obtidos pelo hardware (no caso do projeto de Estacionamento Inteligente, uma câmera para a coleta de imagens do estacionamento, e um Raspberry Pi para o processamento das imagens), armazenamento, processamento, replicação e backup, além da comunicação entre os diferentes dispositivos.

A plataforma trabalha com a utilização de alguns conceitos. O primeiro deles é o de *Dispositivo*. Eles são representações dos dispositivos utilizados em soluções IoT. Assim, eles podem receber e enviar dados. O envio e recebimento desses dados pode ser feito de duas maneiras: por meio do protocolo HTTP utilizando requisições e também por inscrição/subscrição, por meio da utilização do protocolo MQTT.

O segundo conceito é o de *Canal*. Por meio de canais é possível agrupar mensagens que são semelhantes para serem processadas conjuntamente. Ele está diretamente ligado aos dispositivos, pois, um dispositivo pode ter diferentes sensores, motores, ou outras ferramentas, cujos dados são semelhantes. Assim, cada um desses dados pode ser obtido ou enviado por meio dos canais.

O terceiro conceito é o de *Rota*. Como em IoT os dispositivos normalmente conversam entre si, o conceito de rotas é utilizado para fazer essa comunicação. Por meio dele, é possível ligar um dispositivo de entrada, como um sensor, a um dispositivo de saída, como por exemplo um LED.

O quarto conceito é o de *Transformação*. Mensagens obtidas por meio de dispositivos de entrada muitas vezes precisam ser tratadas para serem utilizadas por um dispositivo de saída. Assim, as transformações ocorrem nessa rota entre um dispositivo e outro. O processamento dessas mensagens é feito por meio de plataformas externas que as obtém, processam os dados e as devolvem para a plataforma da Konker para que o processo de troca de dados entre dispositivos continue.

No projeto foram utilizados somente os dispositivos, onde cada um representa o estacionamento de um instituto e armazena a quantidade de vagas disponíveis.

4.1.2 Android

O sistema operacional Android é um dos mais utilizados do mundo para dispositivos mobile. Foi inicialmente lançado em 2007 como um sistema beta e desde então vem sendo utilizado em diversos aparelhos, sendo o mais comum, os smartphones. O Android foi desenvolvido baseado no sistema Linux, e é um projeto open-source. O projeto é mantido por um conjunto de empresas, sendo a Google a que encabeça o grupo.

4.1.3 Android Studio e Android SDK

Para o projeto foi desenvolvido um aplicativo Android, utilizando o *Android SDK (Software Development Kit)*, conjunto de ferramentas disponibilizada para o desenvolvimento de aplicativos do sistema. Ele disponibiliza diversas bibliotecas que ajudam no desenvolvimento, além de outras que são necessárias ao funcionamento do aplicativo. Cada nova versão do

Android lançada possui um SDK diferente, que possibilita que os aplicativos criados utilizando esse kit, sejam compatíveis com todos os smartphones que utilizem essa versão do sistema.

Em conjunto com o SDK, foi utilizado também a IDE (*Integrated development environment*) Android Studio, programa oficial utilizado para desenvolvimento de aplicativos do sistema. O Android Studio é uma IDE desenvolvida baseada na IDE existente Intellij, que é utilizada para o desenvolvimento Java. Ela foi desenvolvida pela *JetBrains* e oferece todas as ferramentas para desenvolvimento desde o backend do aplicativo até o frontend do mesmo.

4.1.4 Kotlin

Kotlin é uma linguagem desenvolvida pela *JetBrains*, e que foi escolhida em 2019 pela Google como a linguagem recomendada para o desenvolvimento de aplicativos Android. Ela foi pensada com o objetivo de interagir completamente com a linguagem Java e também ser uma alternativa a ela, sendo mais concisa, porém, sendo ainda fortemente tipada. Utiliza como plataforma de execução, também a JVM utilizada pelo Java.

Para o projeto foi utilizada a linguagem Kotlin, que é uma linguagem que segue principalmente o conceito de orientação a objetos (utilizando também, conceitos de programação funcional) e que permitiu o projeto ser modularizado de forma a trabalhar com microserviços que compõe o serviço final.

4.1.5 HTTP Requests e biblioteca OkHttp

O aplicativo criado pelo projeto funciona, pelos conceitos da plataforma Konker, como um dispositivo de saída. Assim, os dados precisavam ser obtidos por uma das duas formas que a plataforma disponibiliza e a escolhida foi por meio das requisições HTTP, já que o sistema não possui a necessidade de ficar o tempo todo verificando alterações nas vagas de estacionamento, sendo esse processo necessário somente quando o usuário solicita o acesso a esses dados.

O protocolo HTTP trabalha com a troca de mensagens entre clientes e servidores por meio da rede de internet, para realizar ações sobre recursos identificados por URL's. Para isso são utilizadas requisições enviadas pelos clientes e respostas a essas requisições devolvidas pelos servidores.

Para essas ações, são utilizados alguns métodos que possuem diferentes propósitos. São eles: GET, POST, HEAD, PUT, DELETE, PATCH e OPTIONS. Para o projeto foram utilizados somente os métodos GET e POST. O método GET é utilizado para a obtenção dos recursos identificados por uma URL. Por outro lado, o POST é utilizado para o envio de dados do cliente para o servidor, para a criação ou atualização de um recurso em uma URL.

Para a realização dessas requisições foi utilizada a biblioteca OkHttp que é uma biblioteca open source que lida com requisições HTTP nas linguagens Java e Kotlin. Ela utiliza o protocolo HTTP/2.0 que é uma atualização do antigo protocolo HTTP e que consegue

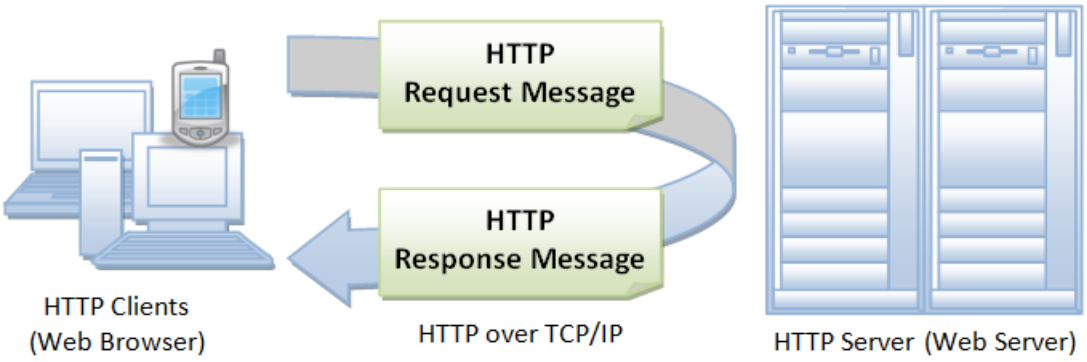

Figura 2: Fluxograma de uma requisição HTTP [4]

realizar as requisições de forma eficiente e segura. A Figura 2 apresenta o fluxo de uma requisição HTTP.

4.1.6 SQLite

O SQLite [9] é um banco relacional open source utilizado em aplicações mobile, sites de pequeno porte ou aplicações desktop que não necessitam de um banco de dados SQL completo. Ele se caracteriza por ter um tamanho reduzido, não necessitar de um servidor para funcionar, já que vem embutido à aplicação, e ser de fácil utilização. Por ser um banco relacional de pequeno porte, ele não apresenta todas as funcionalidades de um banco relacional completo, como por exemplo, não tem acesso a chaves estrangeiras, delete em múltiplas tabelas, entre outras. Porém, apresenta funções básicas para que algumas informações sejam armazenadas pela aplicação.

O sistema Android apresenta suporte para o SQLite [10], apresentando um pacote nativo para a utilização do banco (*android.database.sqlite*). Como o banco utiliza o armazenamento do sistema para salvar os dados, o mesmo ocorre com o Android. Assim, o banco de dados SQLite é armazenado no armazenamento interno do dispositivo Android, na pasta privada do aplicativo. Com isso, o banco fica seguro pois, por padrão, esta pasta não pode ser acessada por outros aplicativos e nem pelo próprio usuário.

Para o aplicativo do projeto foi utilizada a classe *SQLiteOpenHelper* do Android para o gerenciamento do banco. Por meio dela é possível realizar consultas no banco de maneira mais fácil. O banco foi utilizado no aplicativo como forma de armazenar os eventos criados pelo usuário.

4.1.7 Google API

A Google disponibiliza uma série de API's para o desenvolvimento mobile e web. Para a sua utilização, deve-se realizar um cadastro na plataforma de mapas da Google Cloud [11] para a obtenção de uma chave de acesso. No console da plataforma, é possível realizar todo o monitoramento de uso das API's escolhidas para o sistema.

Para o projeto, foi utilizada a API de Mapas. Com ela, é possível adicionar ao aplicativo, ou site, mapas assim como os oferecidos pelo Google Maps. Em sistemas Android, essa utilização é feita por meio do SDK do Maps, que faz parte do SDK utilizado para o desenvolvimento do aplicativo. Como para o projeto não era necessário todas as funcionalidades do Google Maps, foram utilizados somente mapas estáticos, que permitiam a utilização de marcadores, para localizar os institutos e seus respectivos estacionamentos. Mesmo sendo possível navegar pelo mapa e utilizar o zoom para aproximação de localizações, o mapa é estático, pois não é possível realizar nenhuma interação com ele, no sentido de procurar por novas localidades e traçar rotas.

4.2 Desenvolvimento

4.2.1 Eventos

O desenvolvimento inicial do projeto se deu com a criação dos eventos. Para isso, foram criadas duas telas com formulários para que o usuário pudesse adicionar seus eventos no banco de dados do sistema e posteriormente pudesse acessar os eventos criados. Para isso, foi projetado o seguinte fluxo de funcionamento:

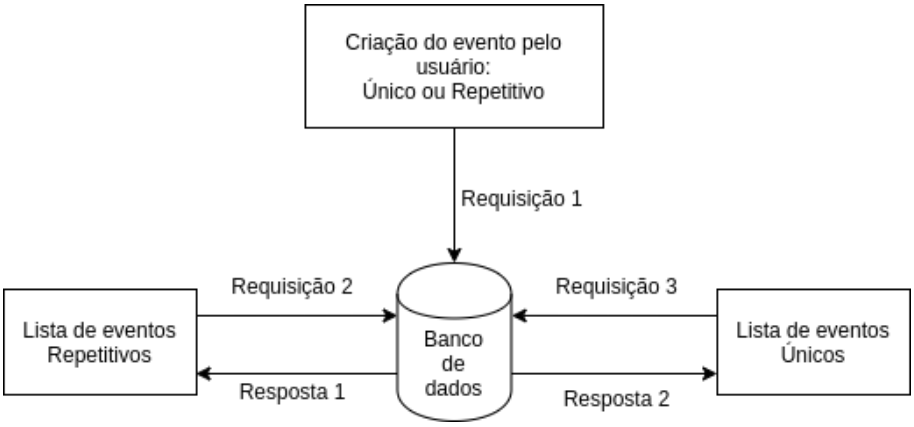

Figura 3: Fluxo de criação e visualização de eventos

No fluxograma da Figura 3, a Requisição 1 representa a requisição feita ao banco para o armazenamento dos eventos. Como explicado na seção de conceitos do projeto, os eventos possuem estruturas parecidas, diferenciando-se apenas na parte do armazenamento das datas, onde eventos únicos armazenam as datas completas, com dia, mês e ano e os eventos repetitivos armazenam o dia da semana na qual o evento ocorre. Assim, como o banco SQLite não permite a utilização de campos do tipo date, este campo foi armazenado como uma string, permitindo assim que uma única tabela fosse utilizada para o armazenamento dos eventos. A diferenciação entre eles é feita com a utilização do campo *repetitive*, que em caso de valor 0, representa os eventos únicos e em caso de valor 1, os eventos repetitivos.

As Requisições 2 e 3, representam as requisições feitas para buscar a lista de eventos repetitivos e únicos, respectivamente, para que sejam mostradas ao usuário nas telas de

visualizações dos eventos cadastrados. As Respostas 1 e 2 são as respostas do banco com as listas, para as telas de visualização de eventos repetitivos e únicos, respectivamente.

Dentro das listas de eventos, o usuário pode então acessar os eventos para a consulta dos estacionamentos próximos. Os eventos repetitivos, como são recorrentes, não são deletados após seu acesso. Já os eventos únicos, como só ocorrem uma vez, são deletados do banco de dados. O fluxo de funcionamento para o acesso dos dois eventos está descrito nos fluxogramas a seguir:

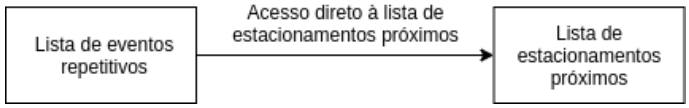

Figura 4: Fluxo de acesso aos estacionamentos próximos de um evento repetitivo

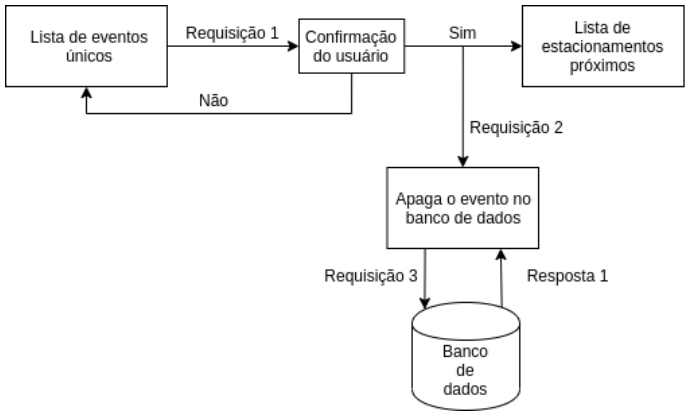

Figura 5: Fluxo de acesso aos estacionamentos próximos de um evento único

Na Figura 4, temos que o acesso dos eventos repetitivos à lista de estacionamentos próximas é direta. Já na Figura 5, o acesso à essa lista passa por uma confirmação do usuário. Ao clicar em um dos eventos únicos, o usuário é perguntado se deseja acessar o evento, pois, uma vez acessado, ele é apagado do banco de eventos. Caso a resposta seja não, nada acontece. Caso a resposta seja sim, o evento é apagado, com a Requisição 2 a classe que gerencia o banco e com a Requisição 3 que envia a query ao banco para apagar o evento. A Resposta 1 representa a resposta de que a query foi realizada com sucesso. Em paralelo, o usuário tem acesso à lista de estacionamentos próximos.

Em caso de acesso a qualquer um dos eventos, o fluxo de dados para o carregamento das listas de estacionamentos próximos com as respectivas quantidades de vagas, funciona como descrito a seguir:

Na Figura 6, a Lista de estacionamentos próximos é carregada a partir da Requisição 1 feita para a o método responsável pelo processamento dos dados. Assim, uma requisição HTTP é feita à plataforma da Konker para a obtenção dos últimos dados de vagas de todos os estacionamentos, cujo retorno é identificado como Resposta 1. Em seguida, é feita uma requisição ao método que calcula as distancias entre estacionamentos por meio

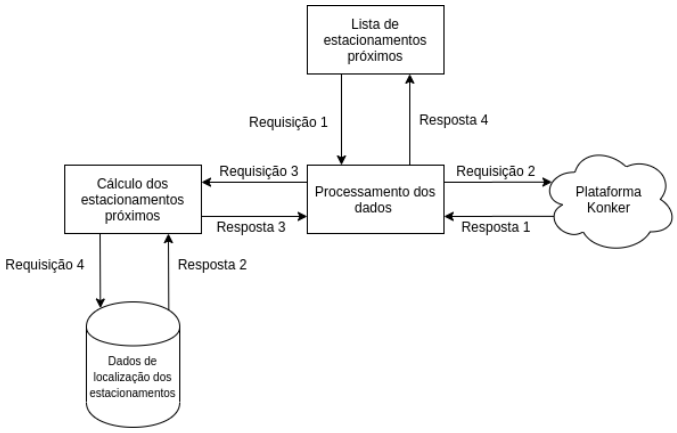

Figura 6: Fluxo de visualização de listas de estacionamentos próximos para eventos os eventos

da fórmula de Haversine e retorna os estacionamentos próximos. Para isso é feita uma requisição dos dados de latitude e longitude no banco de dados (Requisição 4 e Resposta 2) e em seguida as distancias são calculadas e uma lista de estacionamentos próximos é retornada na Resposta 3. Assim, o método processa os dados e retorna, na Resposta 4, a lista com os estacionamentos próximos e as respectivas quantidades de vagas. Esse processo é refeito a cada 30 segundos para a atualização da quantidade de vagas para o usuário.

Na página de estacionamentos próximos, ao clicar em um dos estacionamentos, o usuário é direcionado a um mapa com a localização marcada do instituto de destino e do estacionamento selecionado. Para isso, foi utilizada a API da Google para o Maps. Com ela, é possível adicionar ao aplicativo um mapa estático com os marcadores de localização. Assim, foram utilizadas as informações de latitude e longitude dos institutos, armazenadas, para criar os marcadores no mapa.

4.2.2 Visualização de dados dos institutos

O objetivo principal do aplicativo é a utilziação dos eventos para que o motorista tenha acesso aos dados de vagas em seu caminho até o instituto de destino. Porém, também foi criada uma ferramenta que mostra ao usuário as informações de localização do instituto (rua, número, cep), para que possa planejar seu caminho previamente, além da quantidade de vagas do estacionamento do instituto, no momento do acesso. Além disso, é possível ver os estacionamentos próximos e a localização do instituto, no mapa, por meio da API do Google descrita anteriormente. O fluxo de funcionamento dessa ferramenta é descrito a seguir:

Na Figura 7, a Requisição 1 e Resposta 1, representam as obtenção dos dados de localização do instituto, que estão armazenados no banco de dados. Caso o usuário decida visualizar a localização do instituto e dos estacionamentos próximos, no mapa, a Requisição 2 é feita. Assim, na tela com o mapa, é feita uma requisição ao método que calcula as distâncias dos estacionamentos ao instituto acessado (incluindo o próprio estacionamento

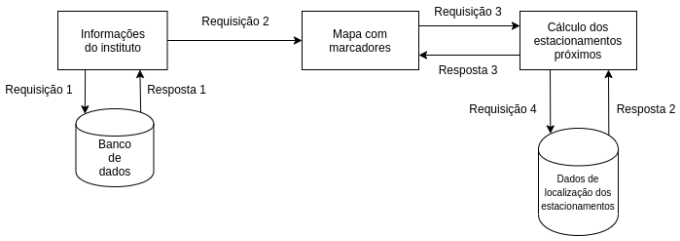

Figura 7: Fluxo de visualização de informações dos institutos

do instituto), e retorna a lista dos mais próximos, com as respectivas informações de latitude e longitude que são utilizadas para adicionar os marcadores ao mapa.

5 Resultados

A versão final do aplicativo e os exemplos de sua utilização estão representados por *screenshots* retirados do aplicativo rodando em um aparelho Android. A Figura 8 apresenta a tela inicial do aplicativo e as próximas subseções mostram as telas e exemplos de cada funcionalidade do aplicativo. O repositório com o projeto pode ser encontrado em [16]

Figura 8: Tela principal do aplicativo

5.1 Criação de eventos repetitivos e únicos

Na Figura 9 temos a tela de criação de eventos repetitivos. Todos os campos são necessários, apresentando uma mensagem de erro caso o botão Enviar seja pressionado com um dos campos vazios. O primeiro campo necessário é o título, utilizado para a rápida identificação do evento pelo usuário. O segundo é o instituto onde o evento ocorrerá. A partir dessa informação são calculados os estacionamentos próximos. Os outros dois campos são o dia da semana e o horário do evento, para o planejamento do usuário dos eventos que participará.

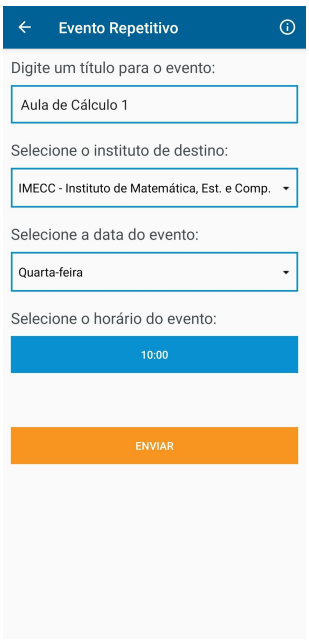

Figura 9: Tela de criação de eventos repetitivos

Na Figura 10 temos a tela de criação dos eventos únicos. Assim como a de eventos repetitivos, todos os campos são necessários para o evento ser armazenado. A única diferença existente para a criação de eventos repetitivos é que para eventos únicos é escolhida uma data, não o dia da semana em que o evento ocorrerá.

5.2 Lista de eventos repetitivos e únicos

Nas Figuras 11 e 12, temos as listas de eventos repetitivos e únicos, respectivamente. Ambas as listas são ordenadas pela ordem de adição do evento e apresentam todas as informações dadas pelo usuário ao criar os eventos. A partir delas é possível acessar os eventos e com isso a lista de estacionamentos próximos.

Na Figura 13 temos a caixa de diálogo aberta ao tentar acessar um evento único. Como discutido na seção de Desenvolvimento, os eventos únicos, ao serem acessados, são apagados do banco de dados. Assim, ao clicar em um evento único, antes de ir à tela de lista de estacionamentos próximos é pedida uma confirmação pelo usuário. No caso dos eventos repetitivos, ele apenas vai para a tela com a lista de estacionamentos próximos.

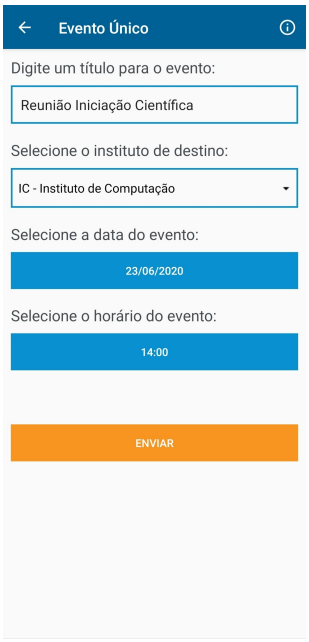

Figura 10: Tela de criação de eventos únicos

Figura 11: Tela de lista de eventos repetitivos

Os eventos podem ser apagados pelo usuário, a partir das listas. Assim, a Figura 14 mostra a caixa de confirmação apresentada ao usuário ao tentar apagar um evento (a ação

Figura 12: Tela de lista de eventos únicos

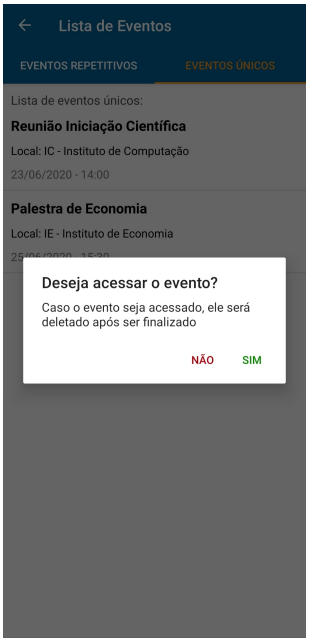

Figura 13: Mensagem de acesso a eventos únicos

de apagar um evento ocorre ao pressionar e segurar o evento desejado).

Figura 14: Mensagem ao apagar eventos

5.3 Lista de estacionamentos próximos

Na Figura 15 temos a tela com a lista de estacionamentos próximos ao instituto onde o evento ocorrerá. Ela mostra os estacionamentos ordenados pela proximidade, sendo o primeiro e mais destacado, o estacionamento do próprio instituto. A quantidade de vagas é mostrada em verde quando não são zero, e em vermelho, caso não hajam vagas. A tela é atualizada automaticamente a cada 30 segundos, podendo ser atualizada manualmente pressionando o botão de atualizar no canto inferior direito.

Esta tela deve ser acessada assim que o condutor ou condutora comece a utilizar o veículo. Assim como um GPS, o smartphone deve ser utilizado como uma forma de consulta, estando em um local de fácil visualização e que não prejudique a condução. Dessa maneira a informação é acessada de forma segura e fácil por quem utiliza o aplicativo.

5.4 Mapa com o instituto destino e o estacionamento selecionado

Ao clicar em um dos estacionamentos próximos, o usuário é redirecionado para a tela com o mapa e os marcadores de localização do instituto do evento (marcador em vermelho) e do estacionamento acessado (marcador em azul). Esta tela é indicada na Figura 16.

5.5 Informações do instituto

Outra funcionalidade citada na seção de Desenvolvimento, foi a de acesso a informações de localização dos institutos. Esta tela é representada pela Figura 17. Nela, o usuário escolhe o instituto desejado a partir do menu e ao efetuar a busca, lhe é mostrado o nome e logo do

Figura 15: Tela de lista de estacionamentos próximos

Figura 16: Tela com o mapa com o instituto destino e um estacionamento próximo

instituto e informações como rua, número e CEP, para o usuário se planejar previamente como chegar ao instituto e também a quantidade de vagas no momento da pesquisa.

Figura 17: Tela com as informações dos institutos

Existe também o botão "Ver Mapa"que leva o usuário a uma tela com um mapa, indicando o instituto pesquisado (marcador em vermelho) e todos os estacionamentos próximos (marcadores em azul). Este mapa tem como função ajudar o usuário a localizar as melhores opções para estacionar seu veículo, e assim poder se planejar previamente. A tela com este mapa está indicada na Figura 18

6 Trabalhos futuros

Como uma forma de contribuir ainda mais para o planejamento do dia-a-dia das pessoas que frequentam o campus, esta solução poderia ser estendida com a adição de um sistema para informar ao usuário os horários em que os estacionamentos apresentam seus picos de lotação e os melhores horários para que o veículo seja estacionado. Para isso seriam utilizadas técnicas de aprendizado de máquina, com os dados de quantidade de vagas, coletados diariamente. Por ser um projeto novo, ainda não é possível que essa ferramenta seja implementada, por conta da falta de dados disponíveis.

Outra funcionalidade possível é a de integrar aos mapas, a localização em tempo real, do veículo, para que ele tivesse uma melhor visualização de sua localização em relação ao instituto desejado e também pensando nas pessoas que estão visitando o campus pela primeira vez e não o conhecem tão bem. Para isso seria utilizado o GPS do aparelho para obter a localização do veículo e também algoritmos para que essa informação fosse mostrada no mapa com uma maior precisão.

Por fim, para abranger uma outra parcela da comunidade, é visada a criação do apli-

Figura 18: Tela com o mapa com o instituto destino e todos os estacionamentos próximos

cativo para o sistema IOS (desenvolvido pela empresa Apple para os smartphones Iphone), tendo assim praticamente todos os usuários de smartphones da universidade conectados ao sistema.

7 Conclusão

Assim como foi observado nas pesquisas para o desenvolvimento do projeto, a etapa de desenvolvimento de plataformas para a visualização dos dados de vagas de estacionamento pelos motoristas não é muito pesquisada e desenvolvida, sendo apresentada de maneira simples por meio de totens na entrada do estabelecimento ou led's de sinalização. Com isso, o projeto trouxe uma outra alternativa para trabalhos futuros.

Como foi desenvolvido para mostrar os dados de maneira simplificada e direta, o aplicativo se mostrou seguro para o acompanhamento durante a condução do veículo. Por evitar que as pessoas tenham que chegar ao estacionamento para descobrir se existem vagas ou não, há um impacto positivo na produtividade dos membros da comunidade, pois agora eles perdem menos tempo na busca por vagas. Adicionalmente, este projeto também tem potencial de contribuir para a diminuição do congestionamento dentro do campus, pois não há a necessidade de paralisações para procura por espaços para estacionar o veículo.

Por fim, conclui-se que, se utilizada de maneira ampla, a solução aqui proposta poderá resultar em um trânsito mais fluído e organizado dentro do campus contribuindo para o objetivo do projeto Smart Campus de melhorar a qualidade de vida da comunidade da universidade.

Referências

[1] Thiago Pires Romanelli Vicente - *Controle Inteligente de Vagas para Estacionamentos Utilizando o Conceito de Internet das Coisas*. Monografia - Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

[2] Samila Ruane Barboza Santos e Sílvio Rodrigo Lima Passos - *Smart Parking: Uma Aplicação Para Estacionamento em Cidades Inteligentes*. Trabalho de Conclusão de Curso – Universidade Federal de Sergipe Centro de Ciências Exatas e Tecnologia Departamento de Computação, 2017.

[3] João Victor Baggio, Luis Fernando Gonzalez e Juliana Freitag Borin - *Smart Parking - A Smart Solution Using Deep Learning*. Paper – Instituto de Computação da Universidade Estadual de Campinas, 2020.

[4] NTU. HTTP (HyperText Transfer Protocol). [S.I] [2009?]. Disponível em: <https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/http_basics.html>. Acesso em: 09 de jun. de 2020.

[5] SMART CAMPUS UNICAMP. Saiba mais sobre o projeto. [S.I] [2018?]. Disponível em: <https://smartcampus.prefeitura.unicamp.br>. Acesso em: 09 de jun. de 2020.

[6] KONKER LABS. Página inicial. [S.I] [2020?]. Disponível em: <http://www.konkerlabs.com>. Acesso em: 09 de jun. de 2020.

[7] KONKER LABS. Guia de Uso da Plataforma Konker. [S.I] [2020?]. Disponível em: <https://konker.atlassian.net/wiki/spaces/DEV/pages/28180518/Guia+de+Uso+da+Plataforma+Konker>. Acesso em: 09 de jun. de 2020.

[8] NOVAS SOFTWARE. Página de download do aplicativo na Play Store da Google. [S.I] [2019?]. Disponível em: <https://play.google.com/store/apps/details?id=com.novas.smartparking>. Acesso em: 09 de jun. de 2020.

[9] SQLITE CONSORTIUM. Página principal do programa. [S.I] [2000?]. Disponível em: <https://www.sqlite.org/index.html>. Acesso em: 17 de jun. de 2020.

[10] ANDROID. Salvar dados usando o SQLite. [S.I] [2018?]. Disponível em: <https://developer.android.com/training/data-storage/sqlite>. Acesso em: 17 de jun. de 2020.

[11] GOOGLE. Página principal do Google Cloud. [S.I] [2018?]. Disponível em: <https://cloud.google.com/maps-platform>. Acesso em: 18 de jun. de 2020.

[12] GOOGLE. Página principal do Google Maps Platform. [S.I] [2008?]. Disponível em:<https://developers.google.com/maps/documentation>. Acesso em: 18 de jun. de 2020.

[13] GEONET - THE ESRI COMMUNITY. Distance on a sphere: The Haversine Formula. [S.I] [2017]. Disponível em:<https://community.esri.com/groups/coordinate-reference-systems/blog/2017/10/05/haversine-formula>. Acesso em: 18 de jun. de 2020.

[14] MOVABLE TYPE SCRIPTS. Calculate distance, bearing and more between Latitude/Longitude points. [S.I] [2019?]. Disponível em:<https://www.movable-type.co.uk/scripts/latlong.html>. Acesso em: 18 de jun. de 2020.

[15] ICONS8. Free Icons, Photos, Vectors, Music, and Tools. [S.I] [2020?]. Disponível em:<https://icons8.com>. Acesso em: 04 de ago. de 2020.

[16] VITOR AOKI. Repositório Git do projeto. [S.I] 2020. Disponível em:<https://github.com/vitorkaoki/Smart-Parking>. Acesso em: 25 de jun. de 2020.