http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2021.v13.n4.e373
ISSN on-line 2178-8332

Colloquium Exactarum

Submetido: 01/06/2019 Revisado: 28/10/2021 Aceito:16/02/2022

O USO DE SERVIÇO RESTFUL PARA GERENCIAR E GERAR CONTEÚDO TURÍSTICO PARA APLICATIVO MÓVEL

The use of service restful to manager and generate turist content to mobile application

Maycon Rayone Rodrigues Xavier; Francisco de Assis da Silva; Leandro Luiz de Almeida; Danillo Roberto Pereira; Mario Augusto Pazoti

Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática de Presidente Prudente – FIPP, Presidente Prudente, SP.
E-mail: maycon.rayone@gmail.com, chico@unoeste.br, llalmeida@unoeste.br, danilopereira@unoeste.br, mario@unoste.br

RESUMO – Na atualidade presencia-se uma grande evolução tecnológica, que torna fundamental prover informações para diferentes aparelhos móveis com plataformas diferentemente desenvolvidas. Esta evolução proporcionou ao ramo turístico a expansão de seu meio de comunicação através de todas as mídias tecnológicas disponíveis. Porém, por se tratar de um ramo gigantesco e que possui diferentes soluções para diferentes situações, existe um grande volume de dados que precisa ser filtrado, e assim tornou-se maçante utilizar uma variedade de soluções que busque destinos, verifique reputação e crie um itinerário. Como forma de atender a esta demanda, buscou-se, apresentar com este trabalho, uma solução computacional que centralize toda a informação turística. Foi construído um aplicativo móvel e um Web Service que, por meio da arquitetura REST (*Representational State Transfer*) utiliza protocolos de transferência para se comunicarem. Os resultados obtidos, demonstram uma eficiência significativa de tempo resposta e agilidade com relação as demais soluções discutidas. Todos os dados foram validados em laboratório baseados em um estudo de caso real em um pequeno munícipio turístico.
Palavras-chave: Aparelhos Móveis; Web Service; REST; Aplicativo Móvel.

ABSTRACT – Nowadays there is a great technological evolution making it fundamental to provide information for different mobile devices with differently developed platforms. This evolution has given to the tourist area the expansion of its means of communication through all the technological media available. However, because it is a gigantic area and has different solutions for different situations, there is a large amount of data that needs to be filtered, so it has become slow to use a variety of solutions that search for destinations, verify reputation and create an itinerary. We look for with this work as a way to solve this demand the development of a computational solution that centralizes all tourist information. A mobile application and a Web Service were built that, through the Representational State Transfer (REST) architecture, uses transfer protocols to communicate. The results obtained demonstrate a significant efficiency of response time and agility in relation to the other solutions discussed. All data were validated in the laboratory based on a real case study in a small tourist town.
***Keywords*:** Mobile Devices; Web Service; REST; Mobile Application.

1. INTRODUÇÃO

De acordo com a divulgação do Ministério do Turismo, o mapa turístico atual registra o aumento de munícipios turísticos cadastrados. Em 2016, eram 2.175 cidades em 291 regiões, e no ano seguinte o mapa registrou 3.285 municípios em 328 regiões turísticas, ou seja, em um ano houve um crescimento de 51,3% em alternativas turísticas no país (BRASIL, 2017a).

Segundo dados da ANAC (Agência Nacional de Aviação Civil), em 2017 o Brasil teve em sua totalidade a somatória de mais de 92 milhões de desembarques domésticos no país. Com esse fato é possível notar que os cidadãos brasileiros circulam o país, seja a turismo, negócios ou a lazer, desta forma gerando receita cambial em milhões ao país, tornando esse segmento um fator econômico importante (BRASIL, 2017b).

Por se tratar de um ramo que possui um fluxo de informações gigantesco, o turista se depara com a necessidade de filtrar todas as informações, com o objetivo de realizar processos para tornar sua estadia a mais proveitosa. Para isso, o turista utiliza ferramentas disponíveis de busca, como: Google, Facebook, Google+. Essas alternativas oferecem aplicações que necessitam de uma conexão com a internet para se comunicar com algum Web Service (Serviço da Web) através do protocolo HTTP[1].

De acordo com a W3C – *World Wide Web Consortium* (W3C, 2004) um Serviço da Web é um software capaz de fornecer uma comunicação entre máquina-a-máquina, através de uma rede. Ele possui uma interface WSDL[2] formada para ser processável pela máquina. Outros sistemas interagem com o serviço Web pela prescrição da mesma interface, porém através de mensagens SOAP[3], que normalmente são transmitidas usando HTTP, e serializadas em XML[4] juntamente com outros padrões relacionados à Web. Os Serviços da Web utilizam muitas tecnologias em camadas que são inter-relacionadas, as tecnologias que são consideradas em relação à arquitetura, são: XML, SOAP, WSDL e UDDI[5] conforme demonstrado o exemplo na Figura 1.

Segundo Fielding (2000), existe uma alternativa interessante para construção de aplicações Web distribuídas ou integração de sistemas independentes através da Web, que é denominada como REST (*Representational State Transfer*). Este modelo arquitetural tem como objetivo fornecer um conjunto de regras que, quando aplicadas como um todo, disponibilize um serviço confiável, seguro e escalável, e assim simplifique a comunicação entre sistemas computacionais. A sua estrutura pode ser visualizada na Figura 2.

Na estrutura abordada no estudo de caso deste trabalho, as informações turísticas do local são representadas através do Google Places API Web Service, que acessa os mesmos dados usado pelo Google Maps e o Google+. O usuário ao interagir com o buscador Google, que possui a integração com essas soluções, insere uma busca, que é disparada ao Web Service por meio do Protocolo HTTP, que ao verificar a sua existência na base de dados, retorna uma listagem de lugares cadastrados, com avaliações de usuários do Google+, endereço, telefone e rota para GPS. A solução do Facebook, retorna a "fã page" (página direcionada para empresas, que possui horário de funcionamento, local e mídia para enviar mensagem ou efetuar ligação) do local equivalente a busca, ou filtra por publicações de usuários que contenham a palavra-chave digitada.

Essas soluções apesar de utilizarem o conceito de Web Service para prover informações coerentes sobre turismo tem um grande volume de informações, que podem estar desatualizadas ou incorretas, já que necessita de interações de usuários para mantê-las consistentes, o que pode afetar a integridade das informações solicitadas. Outra funcionalidade vantajosa, que seria armazenar as informações buscadas, não se encontra presente, fazendo com que o usuário utilize papel ou até mesmo um bloco de anotações do dispositivo para armazená-las, e assim utilizá-las no momento desejado.

[1] *Hyper Text Transfer Protocol* (HTTP), é um protocolo de comunicação entre sistemas de informação que permite a transferência de dados principalmente na *World Wide Web* (FIELDING, 2000).

[2] *Web Services Description Language* (WSDL), é um formato XML para descrever serviços de rede como um conjunto de nós de extremidade que operam em mensagens que contêm informações orientadas a documentos ou orientadas a procedimentos (W3C, 2007)

[3] *Simple Acess Protocol* (SOAP), é um protocolo de arquitetura orientada a serviços, que fornece uma estrutura padrão extensível e composta para a troca de mensagens em XML entre dispositivos conectados a uma rede (SANT'ANNA, 2015).

[4] *Extensible Markup Language* (XML), é uma recomendação parar gerar linguagens de marcação para necessidades especiais. XML é capaz de descrever diversos tipos de dados, e seu objetivo principal é a facilidade de compartilhamento de informação através da Internet (W3C, 2016).

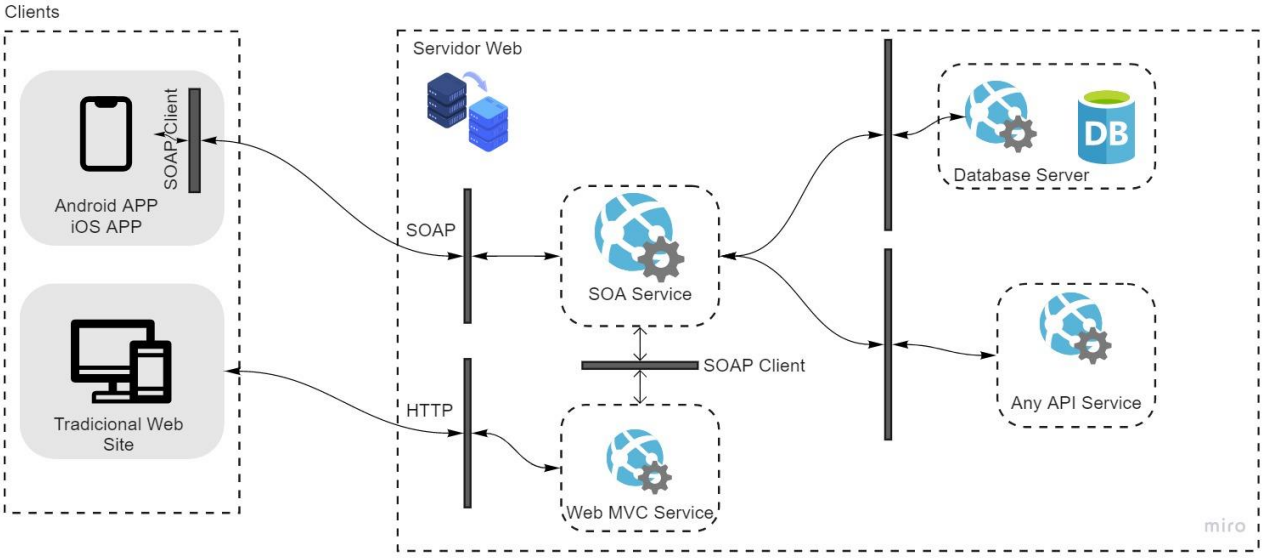
[5] *Universal Description, Descovery and Integration* (UDDI), são especificações que definem um serviço de registro para serviços da Web (OASIS, 2017).

Este trabalho apresenta uma solução para o gerenciamento do conteúdo turístico de um município, que através dos padrões fornecidos pela arquitetura REST, proporciona a comunicação uniforme entre diversas aplicações, especialmente as aplicações móveis, e então gerencie os usuários autenticados, dados de geolocalização, avaliações de usuários e itinerários de usuários.

Foi proposto o desenvolvimento de um Web Service para diminuir o tempo no processo de busca por informações turísticas e agregar novas funcionalidades para a persistência dos itinerários. Esta solução possui ferramentas que facilitam a representação das informações turísticas para reduzir as interações do cliente (aplicativo para Android desenvolvido neste trabalho) na obtenção de dados. O mecanismo de busca utilizado possui a estratégia de compor informações turísticas de estabelecimentos em seus segmentos, e em conjunto com a API do Google Places para fornecer dados de geolocalização, caso um GPS esteja presente no dispositivo móvel. Por mais que se tenha um grande volume de dados, todas as informações de variados estabelecimentos precisam ser persistidas previamente no banco de dados, por possuírem um valor significativo para validar a lógica de negócio do estudo de caso. O trabalho consiste em aplicar os conceitos que a arquitetura REST fornece, a fim de sanar os problemas encontrados no ramo turístico.

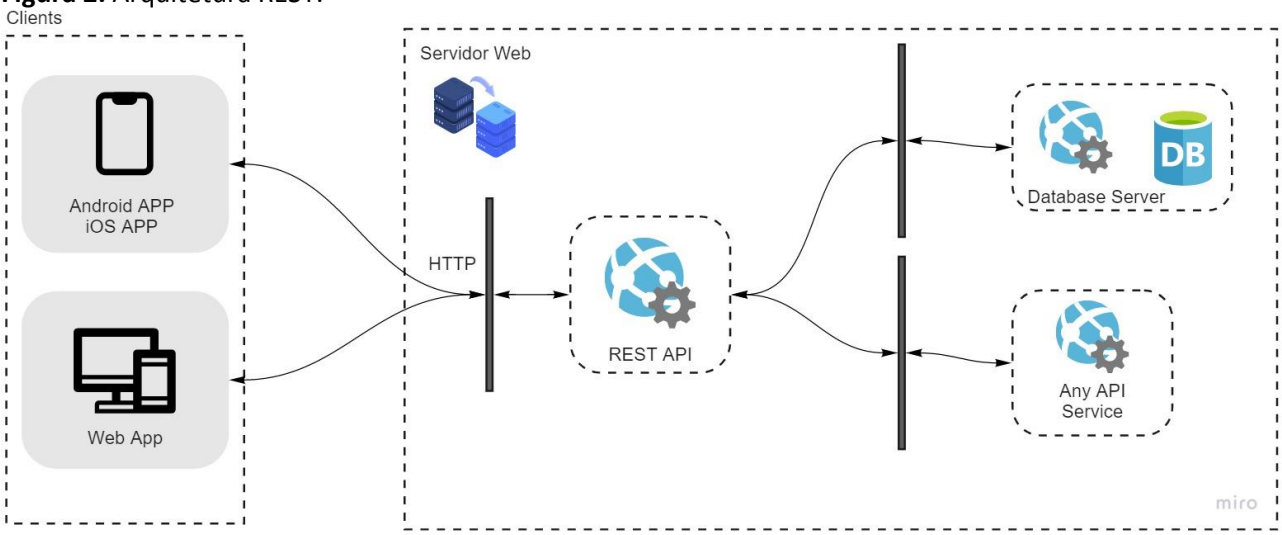
As demais seções deste trabalho estão segmentadas na seguinte maneira: na Seção 2 serão apresentados trabalhos relacionados; na Seção 3 é apresentada a metodologia utilizada neste trabalho; na Seção 4 são apresentados os experimentos realizados e os resultados obtidos por meio de testes; por fim, na Seção 5 são apresentadas considerações finais do trabalho.

Figura 1. Arquitetura SOAP.



Fonte: Os autores
Nota: Modificado de Microsoft (2021).

Figura 2. Arquitetura REST.



Fonte: Os autores
Nota: Modificado de Microsoft (2021).

2. TRABALHOS RELACIONADOS

Eggea (2013) desenvolveu uma aplicação turística baseada em geolocalização para ser utilizada em *smartphones* e *tablets* para Android. O autor empregou conceitos do protocolo REST em um serviço Web capaz de fornecer uma interface que utilize as funcionalidades do protocolo HTTP, e utilizou um modelo de dados JSON para o consumo do aplicativo móvel desenvolvido por ele. A solução promoveu uma comunicação cliente/servidor entre o aplicativo móvel e o Web Service para obter dados de localização e persistir itinerários a usuários autenticados. Utilizando o Google Maps API na versão 2, o autor empregou uma interface customizada ao usuário para filtrar estabelecimentos por localização através do "Serviço de localização do Google Maps" ou por segmento. As informações de estabelecimentos eram exibidas no plano geográfico do Google Maps e foram customizadas com marcadores próprios, ou seja, relativo ao tipo de segmento. O autor realizou um estudo aprofundado para conferir as funcionalidades que o protocolo SOAP e REST forneciam, e desta forma optou pela utilização do modelo REST para a implementação do serviço Web, pelo fato de seguir as melhoras práticas de implementação com menos complicações se comparadas ao SOAP. O trabalho teve como foco utilizar das ferramentas disponíveis para preencher lacunas de informações turísticas da cidade de Curitiba, para que os usuários tivessem acesso a informações consistentes e seguras, fornecidas pelo serviço REST.

Cortez (2014) desenvolveu uma aplicação capaz de gerenciar pedidos de uma pizzaria, de sua produção até a entrega. Ele criou três aplicações, uma capaz de fornecer um serviço Web utilizando os recursos disponíveis no *framework* JSF do pacote Java Web, uma segunda para dispositivo móvel do sistema operacional Android e uma solução Web desenvolvida na linguagem de programação C# para fornecer uma interface ao estabelecimento, capaz de obter os dados gerenciados pelo serviço Web. Por meio da aplicação móvel, os usuários realizam seus pedidos, que através de uma requisição HTTP, interage com o recurso de pedidos do serviço Web. Este recurso recebe a requisição da aplicação móvel, e então, após a devida validação e processamento, persiste os dados no banco de dados. Após este processamento, a aplicação Web desenvolvida para o estabelecimento recebe a atualização do pedido, altera sua situação para produção ou envia conforme a sua ordem na lista de pedidos recebidos.

3. METODOLOGIA

O Web Service proposto neste trabalho possui a responsabilidade de prover a interface para persistir dados turísticos, interfaces de acesso a esses dados, e comunicação com API Google Places para obtenção de informações sobre geolocalização. Esta interface, por meio de interações de uma aplicação móvel, fornece

informações atualizadas referentes aos dados turísticos de um munícipio.

As informações são persistidas em um banco de dados SQLServer, cuja tecnologia aplica os conceitos de relacionamentos entre entidades. As entidades utilizadas (Estabelecimentos, Categorias, Períodos, Comentários, Usuários e Itinerários) foram convertidas em tabelas do SQLServer, para armazenar seus respectivos dados.

A gestão das informações foi realizada através de um Web Services baseado na arquitetura REST. O consumo da informação é feito por meio do protocolo HTTP que representa um sistema de solicitações e respostas, o processo se baseia em um cliente de chamada (navegadores, aplicativos) que envia uma solicitação a um *endpoint* e o *endpoint* a responde.

Todo o processo espera receber solicitações de usuários autenticados, para que possa ter o armazenamento de itinerários e avaliações de acordo com as escolhas de estabelecimentos pesquisados.

Este trabalho foi baseado em um estudo de caso real, que de fato proporcionou a descoberta de muitas informações do âmbito turístico. Buscou-se criar um banco de dados com as informações mais primordiais e assim gerar entidades bem completas na qual pudesse armazenar essas informações.

O Web Service foi desenvolvido utilizando o ASP.NET, que é um *framework*[6] voltado para o desenvolvimento Web, que utiliza das melhores práticas disponíveis no .NET Framework, para fornecer um serviço RESTFul (capacidade de implementar os conceitos de REST), também conhecida como API[7].

Por se tratar de um subconjunto de uma arquitetura maior que é o .NET Framework, foi utilizada a linguagem de programação C# (CSharp), que é orientada a objeto e fortemente tipada. A sua sintaxe é altamente expressiva e possui uma grande familiaridade com as linguagens Java, C++ e C. A grande vantagem de utilizar o framework .NET é que o mesmo inclui um sistema de execução virtual chamado de CLR (*Common Language Runtime*) e um conjunto unificado de bibliotecas. O CLR é uma implementação Microsoft através de um padrão CLI (*Common Language Infrastructure*), que é responsável pela criação de ambientes de execução e de desenvolvimento nos quais linguagens e bibliotecas funcionem de forma integrada (MICROSOFT, 2022).

Os mecanismos sociais (Facebook e Google+) adotados permitem que as informações turísticas se mantenham disponíveis e alimentadas por seus usuários. Porém, o excesso de informações ocasionado pelo crescimento do setor, e a necessidade de intervenção humana na manutenção dos dados afetam os seguintes requisitos:

- Confiabilidade: com várias informações disponíveis por diferentes aplicações sem a devida atualização pode trazer dois tipos de dados distintos para um estabelecimento. Desta forma ocasionando informações divergentes;
- Manutenibilidade: todas as informações disponíveis sobre um determinado local, pode se encontrar em no mínimo mais de uma plataforma de busca ou rede social, e esse excesso de dados cadastrados na internet, quando necessário uma atualização precisa ser propagada em todos os meios disponíveis, causando assim um retrabalho e demanda de tempo;
- Integridade: Para o turista persistir dados de itinerário, buscar endereços e navegar com aplicações de GPS, necessita da utilização de até duas aplicações de diferentes contextos para prover seu propósito e assim comprometendo a informação por perda de dados e uso maior do tempo.

Como forma de contornar os problemas citados e ainda garantir consistência dos dados fornecidos através de um Web Service, apresentou-se, neste trabalho uma arquitetura sustentável, que consiste em aplicar um conjunto de técnicas de busca e uso de APIs com o padrão REST.

O padrão arquitetural REST visa criar uma API que utiliza o HTTP como seu método de comunicação subjacente. O HTTP é um sistema de solicitação e respostas que possui vários detalhes importantes que necessitam de atenção, segundo Kearn (2016), sendo:

- *Resource* (Recursos): define a estrutura da API e seus recursos, que são acessados

[6] Um conjunto de bibliotecas, que fornece o necessário para simplificar a execução de determinadas tarefas no desenvolvimento de software.

[7] *Application Programming Interface* (API), é uma interface de programação de aplicativos elaborada para fornecer um meio de comunicação com serviços on-line da Web que aplicativos podem usar para recuperar e atualizar dados (KEARN, 2016).

através de URLs (*Uniform Resource Locator*) que são aquelas que são utilizadas para acessar páginas da Web comumente;

- *Request Verbs* (Verbos de Solicitação): são verbos que descrevem o que está sendo solicitado no momento para o *Resource.* Para obter dados normalmente é emitido o verbo GET para instruir um *endpoint* (ponto final) de que se deseja obter dados. Porém também existe outros verbos incluindo POST, PUT e DELETE;
- *Request Headers* (Cabeçalhos de Solicitação): são as instruções adicionais que são enviadas juntamente com a solicitação, que podem conter outras configurações;
- *Request Body* (Corpo de solicitação): que contém os dados que são enviados através da solicitação que podem ser pelo verbo POST (criar item) ou PUT (atualizar um item). É importante salientar que essas operações com o *Request Body* necessitam que o modelo de dados enviado seja no formato JSON[8] ou XML;
- *Response Body* (Corpo de Resposta): que é o corpo da resposta, pois o protocolo HTTP é uma mão de via dupla, ou seja, toda a requisição tem uma resposta, e essa resposta pode ser uma página da Web, JSON ou XML;
- *Response Status Code* (Código de Estado da Resposta): que retornam ao cliente os detalhes da solicitação com o estado em que ela se encontra através de códigos.

O desenvolvimento foi realizado utilizando o modelo Web API 2 presente no ASP.NET, que segundo Kearn (2016) possuí uma abordagem de fácil implementação para um serviço Web RESTFul, utilizando todos os recursos que a estrutura .NET tem a oferecer. O Web API é construído baseado no modelo de *pipeline*[9] modular do .NET, e isso faz com que, toda solicitação ao Web Service passe pelo *pipeline* de solicitação .NET primeiro, que então permite que seja adicionado facilmente módulos customizados a solicitações.

O Web API usa conceitos de *Controller* e *Action* também presentes no MVC (*Model*, *View*, *Controller*). Os recursos são mapeados diretamente das *controllers*, normalmente é criado uma *controller* diferente para cada entidade de dados (Estabelecimento, Categoria, Itinerario, etc). Para definir o mecanismo de roteamento, o Web API utiliza funções do .NET, que mapeia as URLs para as *controllers*. As APIs são mantidas em uma rota "/api/" que ajuda a distinguir controladoras API das que não são API do mesmo projeto (KEARN, 2016).

As ações presentes nas *controllers* são usadas para mapear um verbo específico do HTTP conforme descrito anteriormente. Como, por exemplo, a rota "/api/Estabelecimento/All", que responde uma solicitação do tipo GET com todas as entidades do tipo Estabelecimento no formato de dados JSON.

O Web API é apenas uma parte do ASP.NET, e por se tratar de um *framework*, possui várias bibliotecas para agilizar o processo de desenvolvimento, e neste projeto foram utilizadas algumas das tecnologias disponíveis no ASP.NET, tais como:

- *Entity Framework*: É um *framework* para realizar operações de banco de dados que também utiliza o conceito de ORM (*Object Relational Management*). Foi utilizado para mapear as classes programadas, e transformá-las em entidades de banco de dados, configurando relacionamentos, chaves primárias, estrangeiras, campos de auto incremento e outras particularidades;
- ASP.NET *Identiy*: É um sistema identidade que possibilita a autenticação de usuários através de cadastros realizados no próprio serviço, ou através de login externos, utilizados por contas do Google, Twitter, Facebook entre outros. No produto foi utilizado o *Identity* com o padrão OAuth 2.0, que realiza as autenticações dos usuários cadastrados, por meio de um *token* fornecido no processo de login;
- *Scaffolding*: Essa tecnologia refere-se às classes com dados dinâmicos, que gera automaticamente páginas para realizar operações de banco de dados como: criar, ler, atualizar e excluir (CRUD) para cada tabela de acordo com a configuração desejada. No Web Service foi utilizado para gerar o CRUD de

[8] JSON (*JavaScript Object Notation*) é um modelo para armazenamento e transmissão de dados no formato texto, de forma bem compacta (JSON, 2017).

[9] *Pipeline* é uma técnica que permite a busca de instruções além da próxima a ser executada.

estabelecimentos, e categorias a fim de ter uma interface amigável para cadastro desses dados;

- *Routing*: Existe uma tabela de roteamento com uma rota padrão "api/{controller}/{id}", sendo que o id é configurado como parâmetro opcional. Essa rota é direcionada a uma classe controladora (*controller*) que trata as solicitações HTTP. Os métodos públicos da controladora são chamados de métodos de ação, e quando a Web API recebe uma solicitação, ela encaminha para uma ação. Para executar as ações não depende somente da rota do recurso, mas também do *Request Verb* para determinar o que está sendo solicitado.

  No produto, foram empregados esses conceitos para acessar os recursos disponíveis de estabelecimentos, categorias, autenticar usuários, persistir itinerários, filtrar estabelecimentos por categoria e para determinar a latitude e longitude do estabelecimento através de chamadas a Google Place API.

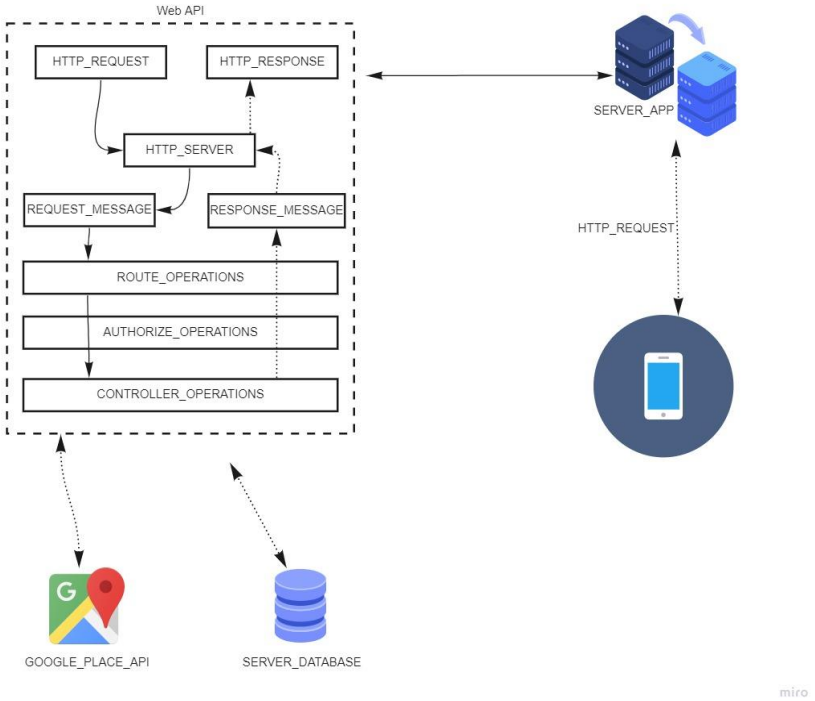
Para que o serviço funcione corretamente, é necessário um servidor com o .NET Framework devidamente instalado caso seja plataforma Windows Server. Para Linux e Mac precisa possuir o Mono instalado, pois também é um CLR que permite desenvolver e executar aplicativos do ASP.NET 5x. Porém independente de qual CLR instalado, a estrutura precisa conter o ambiente de execução KRE (*K Runtime Environment*), pois este fornece tudo que é necessário para hospedar e executar o aplicativo (ROTH, 2014).

O produto foi configurado no Microsoft Azure, em um servidor Windows na camada gratuita, destinada a testes. Foi criado um grupo de recursos, para armazenar o serviço de aplicativos e um servidor de banco de dados SQL Server.

Na Figura 3, é apresentado o cenário de serviço Web proposto neste trabalho, onde existem os seguintes componentes:

- SERVER_APP: É o servidor da aplicação responsável por receber as requisições HTTP e encaminhá-las para a aplicação WEB_API;
- WEB_API: Serviço Web proposto, desenvolvido no template Web API 2 utilizando o framework ASP.NET. A aplicação tem a responsabilidade de conter a lógica do negócio e gerenciá-la;
- GOOGLE_PLACE_API: Web Service do Google que retorna informações sobre locais (estabelecimentos, localizações geográficas ou pontos de interesse);

Figura 3. Componentes para execução do serviço Web.



Fonte: Os autores.

- SERVER_DATABASE: Serviço responsável por conter o banco de dados do projeto. Provê a interface de conexão com a aplicação e contêm as regras de autenticação e desempenho;
- CLIENTE_APP: Aplicativo móvel desenvolvido na plataforma Android, a fim de testar as comunicações com o Web Service proposto, e projetar os dados.

O produto conta com uma etapa de configuração, que deve armazenar as tabelas no banco de dados *serviceupcidade*. As informações referente estabelecimentos são armazenadas nas tabelas *Estabelecimentos*, *Categorias*, *Periodo* e *EstabelecimentoCategoria.* A tabela *Estabelecimentos* é responsável por armazenar as informações de estabelecimentos. Na tabela *Categorias* são armazenados os registros referentes a categorias de estabelecimentos, que contém uma relação de muitos através da tabela *EstabelecimentoCategorias*, que é utilizada em grande escala através do recurso "api/Estabelecimento/CategoriaFilter/{categoriaSlug}", para retornar estabelecimentos relacionados com a categoria inserida no parâmetro do recurso. Já a tabela *Periodo* é responsável por armazenar registros relacionados a horário de funcionamento do estabelecimento registrado.

O ambiente de autenticação disponível no ASP.NET com a biblioteca *Identity* necessita a criação das tabelas *AspNetUsers, AspNetUserClaims, AspNetUserLogins, AspNetUserRoles* e *AspNetRoles.* A tabela *AspNetUsers* é responsável por armazenar dados de usuários cadastrados no sistema, e configurações para autenticações falhas, de dois fatores ou de conta bloqueada. Na tabela *AspNetUserLogins* são armazenadas informações relativas a autenticação de usuários através de outras aplicações, como por exemplo, autenticar-se no sistema por contas do Google+, Facebook ou Twitter. As reinvindicações retornadas através do login externo são armazenadas na tabela *AspNetUserClaims* que guarda o id do usuário e o provedor externo atualizados automaticamente nesta tabela. Por fim, a tabela *AspNetRoles* tem a função de armazenar as atribuições que irão fornecer níveis de acesso ao sistema. Para armazenar os níveis de acesso de um determinado usuário é de responsabilidade da tabela *AspNetUserRoles* manter as informações de usuário cadastrado com determinada atribuição.

Para armazenar as interações dos usuários com estabelecimentos, é utilizada a tabela *Comentario*, na qual são armazenadas as informações referentes às avaliações e comentários sobre o estabelecimento

selecionado. Com o objetivo de persistir dados de itinerários, é utilizada a tabela *Itinerario* que armazena as informações com relação a *estabelecimentos*, data de visita e descrição de atividade.

As tabelas que mantém dados relacionados a *estabelecimentos* são mantidos através de um campo definido como *foreign key*[10], e essas informações são utilizadas no momento de carregar cada entidade com todas as informações necessárias.

Para não ser necessário verificar as informações geográficas de cada estabelecimento cadastrado, foram utilizadas chamadas assíncronas à API Google Place por meio do recurso "/api/Place/{id}", que espera o id do estabelecimento para obter dados de geolocalização do estabelecimento e assim atualiza no banco de dados.

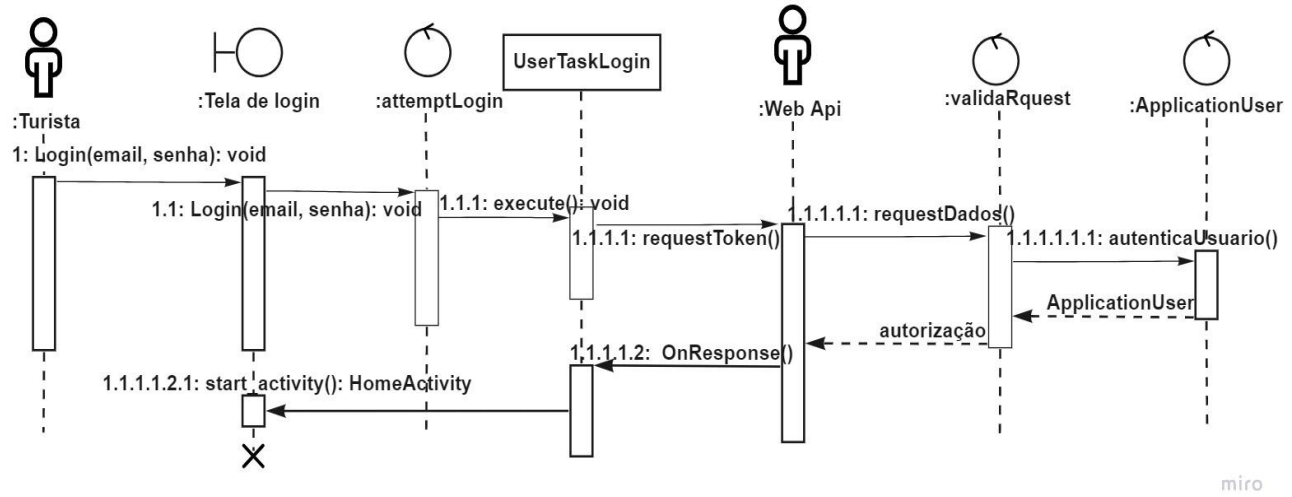
Nas Figuras 4 e 5 pode-se observar as principais etapas executadas pelo *Cliente_APP,* que tem como objetivo consumir os recursos disponíveis no serviço REST proposto (WEB_API).

É possível visualizar no diagrama de autenticação disponível na Figura 3 os métodos essências do processo:

- *attemptLogin*: Método que executa a tentativa de criar usuário, validando os campos e-mail e senha. Tem também como responsabilidade iniciar a ação da execute Classe *UserLoginTask*;
- *UserLoginTask:* Classe que implementa a biblioteca Volley do Android, para executar solicitação assíncrona para o Web Service (Web API), através de uma fila de tarefas criadas no software;
- *execute:* Envia uma requisição do tipo POST para o recurso "/api/Token" que após a devida validação retorna uma resposta com um objeto que contém o *token* de acesso, data de expiração, de login e o tipo da autenticação;

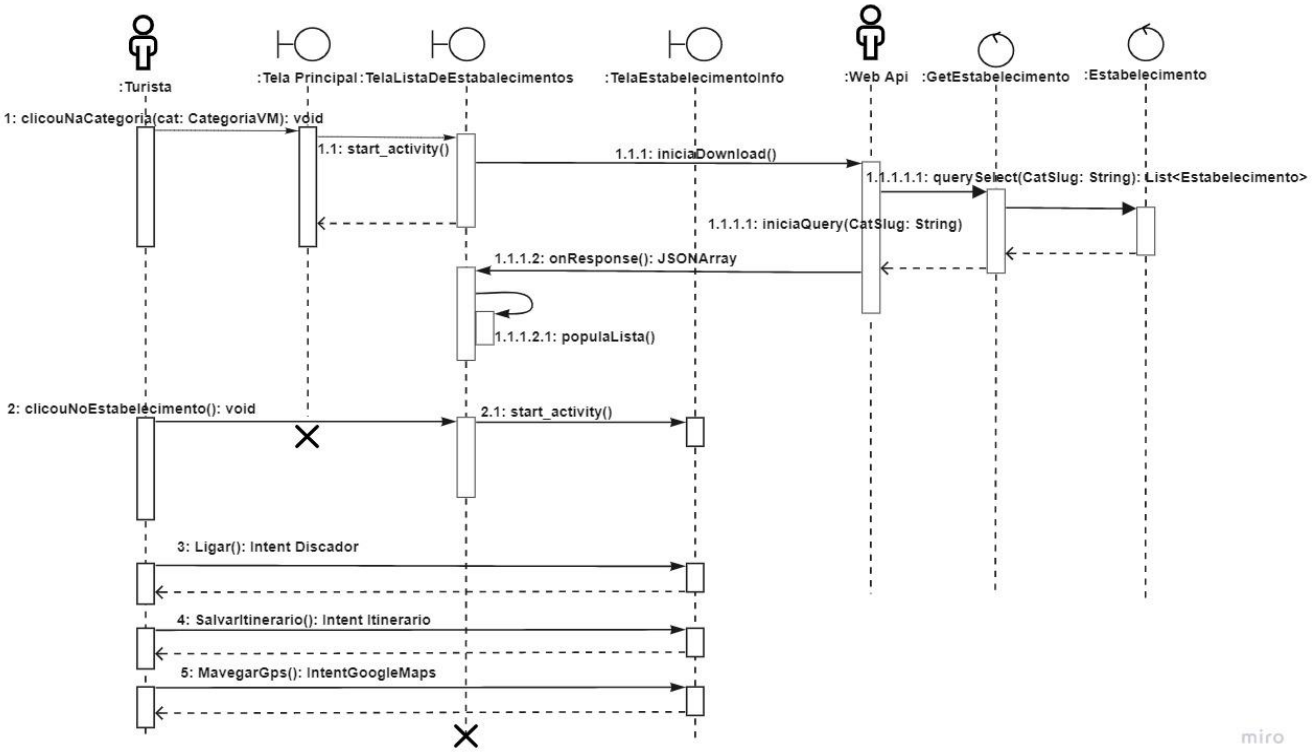
[10] *Foreign key* (chave estrangeira) tem como objetivo manter relacionamento entre duas tabelas do banco de dados (MICROSOFT, 2016).

Figura 4. Diagrama de sequência para autenticar o usuário.



Fonte: Os autores.

Figura 5. Diagrama de interação com usuário.



Fonte: Os autores.

- *startActivity:* Método que retorna a tela inicial para o usuário prosseguir com o acesso;
- *Authorized Object:* Retorna o objetivo da autenticação no formato JSON para interpretação e assinatura do *token* de acesso nas requisições futuras.

Na Figura 5 o diagrama de sequência exibe as interações que o usuário, após a devida autenticação, pode fazer o consumo dos dados gerenciados pelo Web Service:

- *clicouNaCategoria*: Método de seleção da lista de segmentos de estabelecimentos, entre esses, do ramo gastronômico, lazer e hospedagem;
- *iniciaDownload*: Faz o download dos estabelecimentos de acordo com a categoria selecionada, utilizando o

recurso "/api/Estabelecimento/CategoriaFilter/{categoria}" disponível no Web Service, que retorna uma lista de estabelecimentos serializados;

- *clicouNoEstabelecimento*: Método de seleção de item da lista de estabelecimentos que executa o método *startActivity* para exibir os detalhes do estabelecimento;
- *Ligar*: Permite que o usuário, ao interagir com o telefone do estabelecimento, abra o discador do smartphone;
- *SalvarItinerario*: Permite que o usuário salve em seu itinerário o estabelecimento com configurações de lembrete e realize comentários sobre o local;
- *NavegarGPS*: Abre o Google Maps com a localização do estabelecimento para traçar rotas.

O aplicativo desenvolvido na plataforma Android neste trabalho utiliza as bibliotecas da versão da API nível 25 (Android Nagaut). O processo de desenvolvimento foi utilizado através do Android Studio versão 3 a fim de verificar a usabilidade dos recursos do Web Service, que utiliza a linguagem de programação Java.

4. EXPERIMENTOS E RESULTADOS

Nesta seção é apresentado um experimento em laboratório baseado em um estudo de caso em Presidente Epitácio, cidade turística do interior de São Paulo. Essa cidade possui uma variedade de locais para lazer, hospedaria e gastronomia, e o meio de informação disponível para o acesso as informações se encontra em aplicações de busca, mídias sociais que possuem sua base de dados formada através de cadastros de seus usuários.

Para realizar a representação dos recursos turísticos proposto, foi utilizado o padrão REST (Seção 3).

Para demonstrar as funcionalidades do serviço REST proposto neste trabalho, foram mapeadas algumas informações de estabelecimentos e categorias relacionadas ao ramo turístico, que são essenciais, tais como: estabelecimentos, categorias de busca, autenticação de usuários, persistência de dados de estabelecimentos, verificação de avaliações sobre locais e consumo de dados geográficos. Com a solução proposta, será possível atender os seguintes cenários:

- O usuário realiza busca por estabelecimentos específicos e desconhecidos utilizando a mesma solução;
- O usuário realiza ligações e verifica avaliações positivas ou negativas feitas por outros usuários através do perfil do estabelecimento;
- O usuário poderá utilizar os dados de geolocalização para traçar rotas através do Google Maps no Android.
- O usuário pode cadastrar itinerários e configurar lembretes, para ser utilizado em outros momentos;
- O cadastro proporciona ao estabelecimento uma visualização em mais de um tipo de pesquisa, pois abrange várias categorias, sendo possível cadastrar mais de um segmento por estabelecimento;

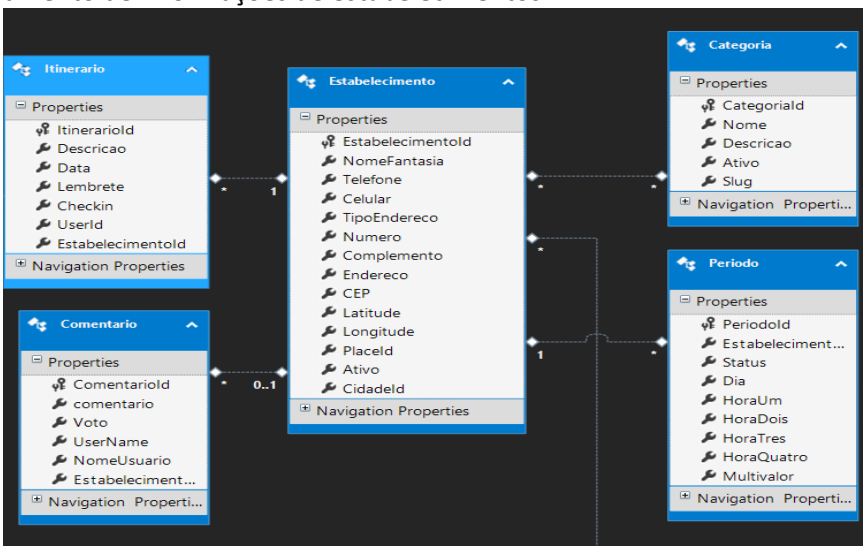
Para demonstrar as funcionalidades do trabalho, primeiramente faz-se necessário adequar o cenário, com as seguintes etapas:

- Pesquisa dos dados: consiste em buscar os dados de estabelecimentos da cidade para formar uma base e cadastra-los no serviço;
- Configuração dos servidores: a devida configuração do SQL Server e da hospedagem com os atributos para funcionamento de aplicações desenvolvidas em .NET;
- Instalação de ambiente: devida instalação dos ambientes de desenvolvimento para ASP.NET e para Android, contento servidores locais (IIS) e máquinas virtuais para emular a aplicação móvel;
- Análise dos dados encontrados: após a devida configuração e instalação dos meios para funcionamento, foi necessário verificar as tabelas que contemplam a lógica do negócio (estabelecimento, periodo, itinerario, identity tables, comentario) para serem criadas e configuradas com os devidos relacionamentos. Na solução proposta, foram criados relacionamentos conforme mostram as Figuras 6, 7 e 8. As integrações descritas nessas figuras permitem o carregamento das entidades através das consultas íntegras da linguagem C# em conjunto com o *Entity Framework*.

Após a instalação do API RESTFul no servidor e a instalação do aplicativo móvel no dispositivo de testes Android e feitas todas as adequações, neste ponto a estrutura proposta já está apta para realizar requisições à API e receber os dados para a disposição em tela.

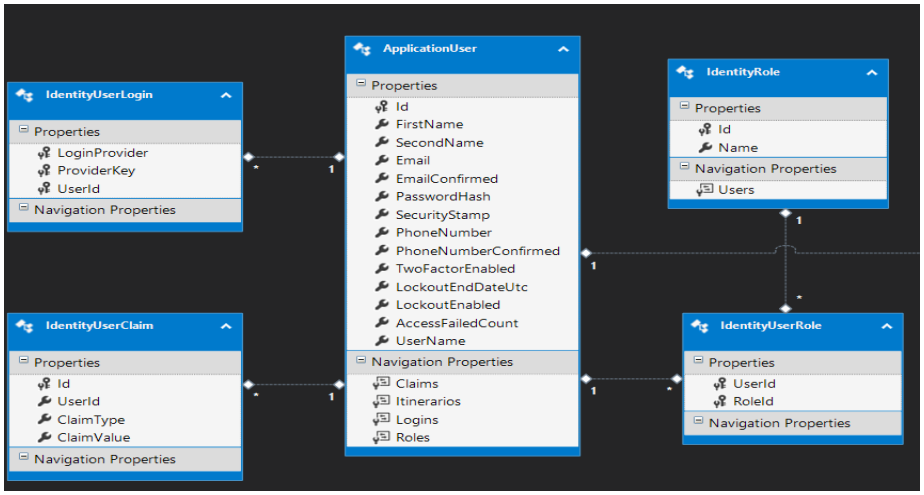
Como forma de coletar indicadores, foram realizadas as seguintes operações no ambiente de testes: Cadastro de um novo usuário através do aplicativo; Autenticação no aplicativo; Consulta de estabelecimentos por categoria; Consulta de estabelecimentos filtrados por segmento e nome; Utilização dos dados obtidos de geolocalização para constatar sua exatidão no GPS; Criar comentários avaliativos; Cadastrar Itinerários; Atualizar Itinerários; e Remover Itinerários;

Figura 6. Relacionamento de informações de estabelecimentos.



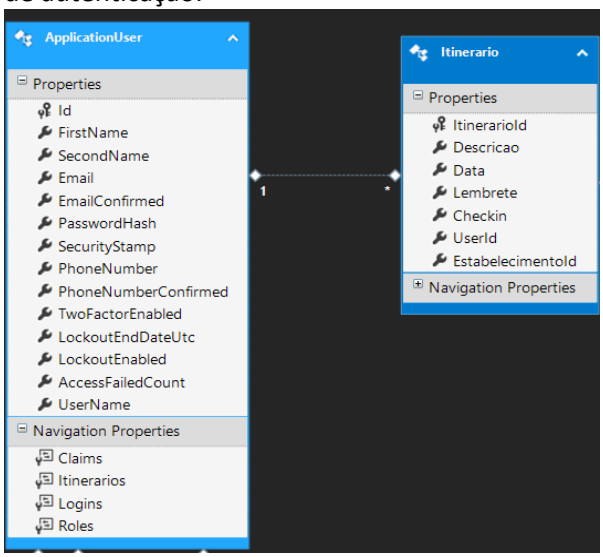
Fonte: Os autores.

Figura 7. Relacionamento entra usuário e itinerários



Fonte: Os autores.

Figura 8. Relacionamento de tabelas do sistema de autenticação.



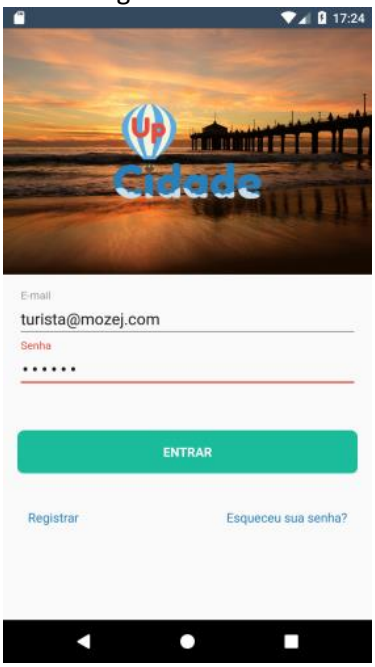
Fonte: Os autores.

Nas Figuras 9, 10 e 11, podem-se observar os resultados das interações do usuário com a interface do aplicativo.

Com a integração de cliente/servidor, as requisições feitas através da interface do aplicativo podem efetuar as requisições HTTP aos recursos disponíveis no serviço RESTFul para que forneçam os dados turísticos disponíveis de interesse do turista.

Para comprovar o resultado do gerenciamento do serviço REST sobre os dados, foram realizadas interações com o aplicativo a fim de verificar o tempo de resposta das ações presentes nas Figuras 10 e 11. As interações com a aplicação não ultrapassaram o limite de 44 segundos para replicar os resultados.

Figura 9. Tela de login.

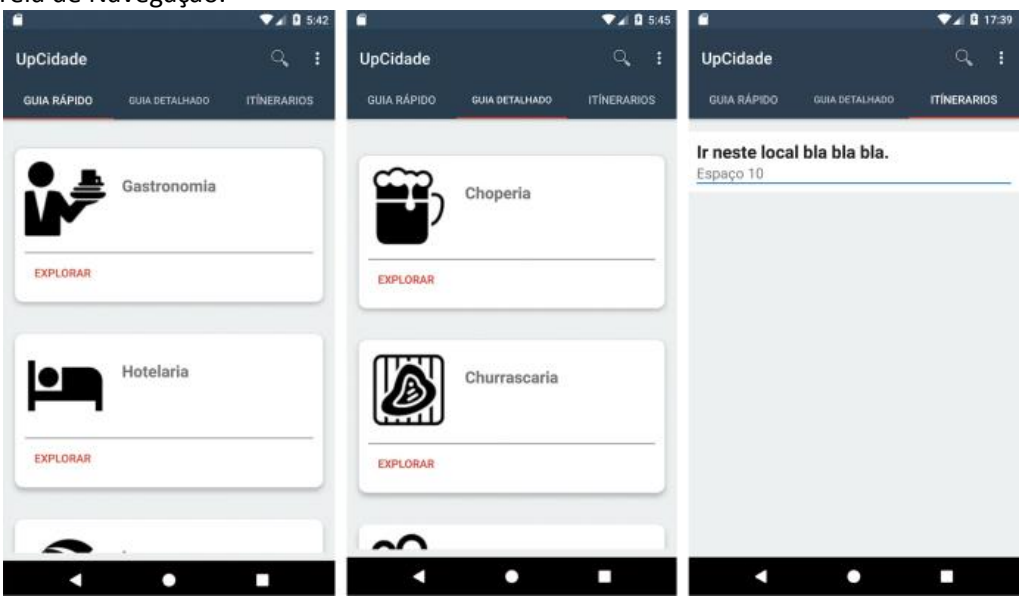


Fonte: Os autores.

As métricas da Tabela 1 foram extraídas do Android Studio, que demonstram o tempo de resposta das interações entre o Aplicativo e o serviço Web. É possível visualizar através das seguintes colunas:

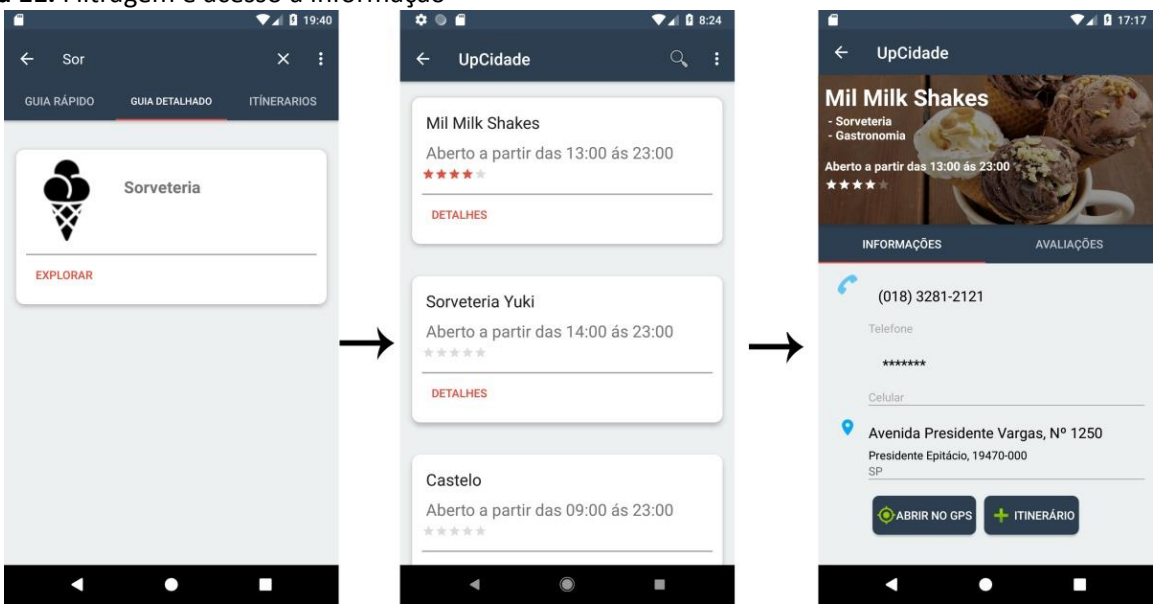
- “Name”: É o recurso acionado pela requisição HTTP, através do dispositivo móvel;
- “Size”: Representa o tamanho do conteúdo retornado da requisição;
- “Type”: É o modelo de dados retornado pela requisição;
- “Status”: Contém o valor do código da situação. O número 200 representa sucesso na solicitação e o 201 representa que um novo recurso está sendo criado;
- “Time”: Possui o valor do tempo de resposta do serviço Web.

Figura 10. Tela de Navegação.



Fonte: Os autores.

Figura 11. Filtragem e acesso a informação



Fonte: Os autores.

Tabela 1. Métricas do Android Studio.

Name	Size	Type	Status	Time
api/Itinerario/all	2B	application/json	200	1s 189ms
api/Estabelecimento/CategoriaFilter/sorveteria	7.73k	application/json	200	1s 719ms
api/Place/20	3.27k	application/json	200	1s 717ms
api/Comentario	112B	application/json	201	3s 926ms
api/Itinerario	0B	application/json	200	1s 361ms

Fonte: Os autores.

Essas métricas demonstram a eficiência do Serviço Web em tratar a solicitação e retornar os objetos que constroem as informações turísticas no aplicativo móvel.

Para comparar o desempenho atual com as outras aplicações Facebook e Google+, foi considerado o tempo de resposta de cada aplicação e a persistência dos dados em papel ou bloco de anotações. As Tabelas 2 e 3 exibem os resultados que contemplam os seguintes processos:

- "Operação": Exibe qual o tipo de operação realizada.
- "Tempo": Esta coluna exibe o tempo de resposta da "operação" realizada.
- "Retorno": Este visa exemplificar o tipo de retorno recebido pelo usuário.
- "Tempo total": Exibe o tempo total com a somatória de todos os processos anteriores.

Nos dados da Tabela 2, são verificados a eficiência da aplicação Google+ em pesquisa, filtragem, seleção e anotação dos dados. Já nos dados da Tabela 3, é possível verificar as mesmas métricas com relação a aplicação Facebook.

Tabela 2. Métricas Google+.

Dados Google+		
Operação	**Tempo**	**Retorno**
Requisitar Categoria Sorveteria	1,07s	11 Locais
Filtrar Local desejado na Lista	15,03s	Perfil do estabelecimento "Mil Milk Shakes"
Anotação dos dados encontrados (Aplicativo Color Notes)	1min e 16s	-
Tempo Total: 1min e 32s		

Fonte: Os autores.

Tabela 3. Métricas Facebook.

Dados Facebook		
Operação	**Tempo**	**Retorno**
Requisitar Categoria Sorveteria	1,07s	(>) 100 Informações
Filtrar Local desejado na Lista	1,03s	Perfil do estabelecimento "Mil Milk Shakes"
Anotação dos dados encontrados (Papel)	1min e 13s	-
Tempo Total: 1min e 16s		

Fonte: Os autores.

A ferramenta Google+ apesar de possuir um "tempo total" superior ao do Facebook, retornou 11 perfis estabelecimentos do segmento "Sorveteria". Contudo a ferramenta Facebook retornou mais de 100 informações de estabelecimentos com variadas categorias, que continham "fã page", locais, marcações de usuários.

O processo de levantamento de dados sobre estabelecimento foi feito manualmente, que demanda um certo tempo para verificar os dados e inclui-los no banco de dados, mas poderia ser melhorado incluindo a funcionalidade comum de permitir que o proprietário do estabelecimento se cadastre, através de uma interface no próprio aplicativo utilizando o recurso disponível no Web Service, e o público que o utiliza fornecesse dados de feedback caso houvesse alguma mudança de endereço ou o mesmo estivesse fechado permanentemente.

Outro ponto que teve que ser adequado para que a integração funcionasse de forma correta foi criar classes no aplicativo móvel com a implementação da biblioteca Volley, através de *threads*[11] para criar rotinas de requisição e incluí-las nas filas para serem disparadas em segundo plano no aplicativo, e também empregar conceitos de *cache* para armazenar dados que podem ser utilizados novamente, e assim otimizar as buscas, que permite a economia de dados móveis do aparelho caso esteja em uso.

Foi observado no experimento que todo o processo de busca, filtragem e cadastro de itinerários retornou um valor de 44 segundos nos

[11] *Thread* é um programa que tem como objetivo realizar tarefas que independem do contexto atual.

processos, com relação as mídias alternativas que foram de até 1,3 minutos e demonstrou ser 2,3 vezes mais rápida, utilizando como fator predominante as funcionalidades presentes e o consumo dos dados no Web Service.

6. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi mostrar que o consumo de um serviço RESTFul proporciona a exibição de conteúdo organizado e consistente a plataformas móveis, reduzindo o tempo gasto e a busca de informações incorretas de dados turísticos. O Web Service proposto demonstrou rapidez e eficiência se comparado ao processo manual, visto que a cidade que foi utilizada como modelo possui uma pequena quantidade de estabelecimentos cadastrados nas mídias sociais e plataformas disponíveis. Sendo assim proporciona que estabelecimentos que não estejam nessas mídias possam ser uma opção ao turista.

Com a utilização da solução proposta, foi possível manter informações atualizadas e consistentes relacionadas ao turismo da cidade à medida que evidenciasse os estabelecimentos de forma clara e usual, utilizando as tecnologias disponíveis e os conceitos de aplicações provedoras de conteúdo, a fim de atender a demanda do mercado.

Desta forma, conclui-se que o objetivo foi alcançado, pois o processo manual executado leva a cerca de um minuto e trinta e dois segundos, foi reduzido para em média quarenta e quatro segundos, além do que, quando dados disponíveis nas mídias podem ocasionar inconsistência por estarem incorretos, e a persistência de endereços e papel podem ocasionar perda de dados, que então possibilita erros.

Entretanto, apesar dos serviços demonstrarem uma confiabilidade e agilidade nos processos, poderá ser abordado em trabalhos futuros uma análise de usabilidade e nível de satisfação do produto como um todo.

REFERÊNCIAS

BRASIL. Ministério do Turismo. **Novo Mapa do Turismo Brasileiro tem Recorde em Número de Regiões. Brasília,** 2017a. Disponível em: http://www.turismo.gov.br/últimas-notícias/8135-novo-mapa-do-turismo-brasileiro-tem-recorde-em-número-de-regiões.html. Acesso em: 30 out. 2017.

BRASIL. Ministério do Turismo. **Estatísticas e Indicadores.** Brasília, 2017b Disponível em: http://www.dadosefatos.turismo.gov.br/estatísticas-e-indicadores/desembarques-domesticos.html. Acesso em: 27 abr. 2018.

EGGEA, R. F. **Aplicação Android utilizando sistema de localização geográfica para determinação de pontos turísticos na cidade de Curitiba**. (2013). Disponível em: http://repositorio.utfpr.edu.br:8080/jspui/handle/1/19886. Acesso em: 02 jan. 2018

CORTEZ, L. **Sistema De Controle De Pedidos Integração Utilizando Web Service**. 2014. 74 f. Monografia (Graduação em Tecnologia e Análise e Desenvolvimento de Sistemas) – Fundação Educacional do Munícipio de Assis, Assis, 2014. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/1211320197.pdf. Acesso em: 02 jan. 2018.

FIELDING, T. R. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. 90 f. Dissertação (Doutorado em Ciência da Computação) – Universidade da California, Irvine, 2000.

JSON. **ECMA-404 The JSON Data Interchange Standard**. 2017. Disponível em: http://www.json.org/. Acesso em: 05 mar. 2017.

KEARN, M. **Introduction to REST and .net Web API**. (2016) Disponível em: https://manojtechnicalblog.blogspot.com/2016/10/introduction-to-rest-and-net-web-api.html. Acesso em 27 fev. 2017.

MICROSOFT. **Introdução ao .NET Framework.** 2022. Disponível em: https://docs.microsoft.com/pt-br/dotnet/framework/get-started/. Acesso em: 17 de mar. 2022.

MICROSOFT. **Create Foreign Key Relationships**. 2016. Disponível em: https://msdn.microsoft.com/pt-br/library/ms189049(v=sql.120).aspx. Acesso em: 17 dez. 2017.

MICROSOFT. **NET Microservices: Architecture for Containerized .NET Applications**. 2021. Disponível em: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-

microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern. Acesso em: 30 out. 2017.

OASIS. **OASIS UDDI Specification TC**. 2017. Disponível em: https://www.oasis-open.org/committees/uddi-spec/faq.php. Acesso em: 14 fev. 2017.

ROTH, D. Introducing the ASP.NET 5 Preview. Your Next Creat ASP.NET App Starts Here, **MSDN Magazine Issues**, Los Angeles, v. 29, n. 12A, Dez. 2014. Disponível em: https://msdn.microsoft.com/pt-br/magazine/Dn879354.aspx. Acesso em: 02 maio 2018.

SANT'ANNA, M. **Soap e Webservices.** 2015. Disponível em: http://www.linhadecodigo.com.br/artigo/38/soap-e-webservices.aspx#ixzz3rrgXSAyA. Acesso em: 15 nov. 2017.

W3C. **Web Services Architecture**. 2004. Disponível em: https://www.w3.org/TR/ws-arch. Acesso em: 11 fev. 2017.

W3C. **Web Services Description Language (WSDL) Version 2.0 Part 1:** Core Language. 2007. Disponível em: https://www.w3.org/TR/wsdl. Acesso em 13 fev. 2017.

W3C. **XML Essentials**. 2016. Disponível em: https://www.w3.org/standards/xml/core. Acesso em: 15 fev. 2017.