INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

VINICIUS MACIEL

APP WORK - SISTEMA DE GERENCIAMENTO DE HORÁRIO PONTO E GEOLOCALIZAÇÃO.

Josué Toebe
Rafael Marisco Bertei

PASSO FUNDO
2016

VINÍCIUS MACIEL

APP WORK- SISTEMA DE GERENCIAMENTO DE HORÁRIO PONTO E GEOLOCALIZAÇÃO.

Projeto de pesquisa submetido ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, Câmpus Passo Fundo, como requisito parcial para a aprovação na disciplina de Projeto de Conclusão 2 (PC 2).

Orientador: Josué Toebe.
Co-orientador: Rafael Marisco Bertei

PASSO FUNDO
2016

VINICIUS MACIEL

APP WORK- SISTEMA DE GERENCIAMENTO DE HORARIO PONTO E GEOLOCALIZAÇÃO

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Josué Toebe

_____

Rafael Marisco Bertei

_____

José Antônio Oliveira de Figueiredo

_____

Coordenação do Curso

PASSO FUNDO

2016

AGRADECIMENTOS

Primeiramente agradeço a meus familiares em especial a meus pais Noacir Soares Maciel e Rose Mari Maciel pelo incentivo ao estudo desde o inicio de minha caminhada. Também gostaria de estender o agradecimento a minha companheira Morgana Dahmer pelo apoio e paciência durante todo o período de estudos.

Agradeço aos meus orientadores, Josué Toebe e Rafael Marisco Bertei pelo acompanhamento durante este projeto, pelos conhecimentos e experiências compartilhados, sem os quais não seria possível a realização deste projeto.

Agradeço aos colegas pela convivência e companheirismo durante todo este processo de formação. Aos demais professores do IFSul pelos ensinamentos e condições necessárias para o aprendizado.

Por fim agradeço a Deus pela iluminação e força nos momentos de dificuldade.

“What if I say I'm not like the others?
What if I say I'm not just another one of your plays?
You're the pretender
What if I say that I'll never surrender?”
Foo Fighters

RESUMO

Hoje a grande maioria das empresas implementam alguma forma de controle dos horários de seus funcionários, tendo em vista que é previsto e obrigatório segundo a legislação brasileira vigente, representada pela Consolidação das Leis do Trabalho em seu artigo 74, parágrafo segundo, que realizem este controle. Dentre estas opções de registro existem desde as formas mais manuais como controle escrito, até as mais tecnológicas como controles biométricos. Algumas empresas, que possuem funcionários que estão constantemente viajando, enfrentam um problema recorrente: como registrar o horário destes colaboradores e assegurar que eles estejam realmente onde deveriam.

Este trabalho tem o objetivo de realizar o desenvolvimento de uma plataforma completa e integrada entre aplicativo móvel e interface web, possibilitando assim que as empresas consigam gerenciar seus funcionários mesmo quando estão distantes das imediações comuns. Para isso foram utilizadas tecnologias como Java SDK, PHP, HTML, CSS e JavaScrtipt e frameworks como Bootstrap, Codeigniter e Slim Rest framework. Como resultado foi desenvolvido um aplicativo para a plataforma Android que faz comunicação com um banco de dados principal e uma interface web onde é possível emitir relatórios das coletas feitas em campo pelos funcionários.

Palavras-chave: Java, PHP, APP, interface web, controle de ponto.

ABSTRACT

The majority of companies today deploy some form of control of schedules of its employees in view of the fact that it is foreseen and required under the Brazilian legislation in force, represented by the Consolidation of Labor Laws in its article 74 and second paragraph, that they carry out this control. Among the clock-in and clock-out registration options that exist, we find manual control forms, which are written, and technological forms, as biometric mechanisms. Some companies with employees, who are constantly traveling, face a recurring problem: how to register the work time of these employees and how to ensure that they really are where they should be.

This study aims to carry out the development of a complete and integrated platform between mobile application and web interface thus enabling companies to manage their employees even when they are removed from the corporation surroundings. For that matter, we used technologies like Java SDK, PHP, HTML, CSS and JavaScrtipt and frameworks like Bootstrap, CodeIgniter and Slim Rest. As a result, we developed an application for the Android platform that makes communication with a primary database and a web interface, where you can issue reports of the data collected by the staff.

Keywords: Java, PHP, application, web interface, companies, employees.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API: Application Programming Interface.
BSD: Berkeley Software Distribution.
CSS: Cascading Style Sheets.
GPS: Global Positioning System.
HTML: Hypertext Markup Language.
HTTP: Hypertext Transfer Protocol.
JSON: JavaScript Object Notation.
REST: Representational State transfer.
SOAP: Simple Object Access Protocol.
SQL: Structured Query Language.
URI: Uniform Resource Identifier.
URL: Uniform Resource Locator.
XML: eXtensible Markup Language.
CLT: Consolidação das Leis do Trabalho
DDL: Data definition language
PDO: PHP Data Objects

SUMÁRIO

1 INTRODUÇÃO

Atualmente a utilização de dispositivos móveis tem criado um leque de possibilidades com uma infinidade de aplicativos desenvolvidos para facilitar o dia a dia dos usuários de *smartphones*. Do mesmo modo as aplicações web tem extrema importância na vida destes usuários, porém sendo utilizados por uma plataforma diferente seus computadores e navegadores.

Considerando as duas plataformas, web e mobile, este projeto se propõe a desenvolver uma aplicação para dispositivos Android que possa suprir as necessidades de empresas que desejam controlar horário ponto e localização de seus funcionários, visto que na legislação brasileira, está previsto através da Consolidação das Leis do Trabalho (CLT), em seu artigo 74, parágrafo segundo. Nesse sentido, foi necessária a criação de duas áreas, uma interface web administrativa, e um aplicativo móvel onde são gerados os dados, além de um *WebService* que é o meio de comunicação entre as áreas.

Operacionalmente o aplicativo utiliza as funções do celular para armazenar a localização geográfica via GPS. Seu funcionamento consistirá em 3 ações principais: iniciar, fechar o horário e fazer a retransmissão dos dados.

O aplicativo tem como principal objetivo fornecer uma ferramenta às empresas e seus funcionários para que estes possam registrar seus horários de trabalho bem como a sua localização. Com base nas informações geradas pelos funcionários a aplicação web possibilita a geração de relatórios individualizados por cada colaborador da empresa.

1.1 OBJETIVOS

A presente seção tratará dos objetivos abordados neste trabalho de conclusão.

1.1.1 Objetivo geral

O presente trabalho tem como objetivo o desenvolvimento de um conjunto de aplicações formado por um aplicativo para a plataforma Android, uma interface web, e comunicação webservice.O aplicativo tem como principal funcionalidade, fornecer uma ferramenta aos funcionários das empresas para que possam registrar seus horários de trabalho, bem como a sua localização geográfica. Com base nas informações geradas pelos funcionários foi a interface web fornece a gerencia da empresa acesso aos dados os dados vindos do aplicativo.

1.1.2 Objetivos específicos

- Desenvolver uma aplicação na plataforma *Android* capaz de registrar a geolocalização e o horário ponto de funcionários.
- Desenvolver uma interface WEB onde serão disponibilizados cadastros de funcionários, bem como relatórios com dados registrados no aplicativo.
- Por meio de *web services* desenvolver a comunicação entre aplicativo e interface web.

1.2 JUSTIFICATIVA

O principal motivo que levou o desenvolvimento deste projeto é a criação de uma ferramenta que possa resolver um grande problema na gestão de empresas, qual seja, o controle da frequência e atividades de seus funcionários, em especial dos que se deslocam para longe das instalações destas empresas. Desta forma, este projeto busca poder operacionalizar este controle através de tecnologia, mais precisamente da tecnologia de desenvolvimento móvel em conjunto com o desenvolvimento de aplicações web, criando assim uma solução integrada capaz de prover o controle dos funcionários e suas atividades externas.

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentados os embasamentos teóricos necessários para o desenvolvimento deste projeto. Serão abordados temas como a plataforma de

dispositivos móveis Android, bem como a plataforma de desenvolvimento de aplicações, utilizando a linguagem Java para Android. Também serão abordados conhecimentos para o desenvolvimento de aplicações web como a linguagem PHP e o *framework* Codeigniter. Além disso, será tratado do funcionamento do *framework front-end* Bootstrap para desenvolvimento de interfaces WEB. Por fim serão abordados os conceitos de RESTful *webservice* e JSON, utilizados na comunicação entra aplicativo e sistema *WEB*.

2.1 Plataforma de Dispositivos Móveis Android

Atualmente o mercado de celulares e *smartphones* vêm crescendo cada vez mais, estatística esta comprovada em estudos que mostra que mais de 3 bilhões de pessoas possuem um aparelho celular. De encontro a isto cresce também a procura de equipamentos com mais recursos e funcionalidades, para facilitar o seu dia a dia.(LECHETA, 2013).

Em resposta as necessidades do mercado a Google adquiriu em 2005 uma plataforma que foi criada em 2003 por Andy Rubin, Rich Miner,Nick Sears e Chris White. Esta plataforma foi desenvolvida baseando-se no SO Linux, possuindo diversas aplicações instaladas, e ainda disponibilizando um ambiente de desenvolvimento ousado flexível e muito poderoso, o que possibilitou ao Android assumir uma posição muito importante no mercado dos *smarphones*.

2.1.1 Android SDK Java

O Android SDK é um software utilizado para desenvolver aplicações voltadas para o SO. Dispondo de muitas ferramentas, desde a emulação de aparelhos até a comunicação com aparelhos reais, e suporte completo para a linguagem Java, com todas as classes necessárias para o desenvolvimento acopladas, esta plataforma de desenvolvimento tem grande papel na criação de muitos dos aplicativos encontrados no mercado atualmente.

Como sua linguagem é o Java, o SDK precisa ter uma versão da linguagem instalada sendo que as mais indicadas são versões a partir do JDK 6.0 para evitar

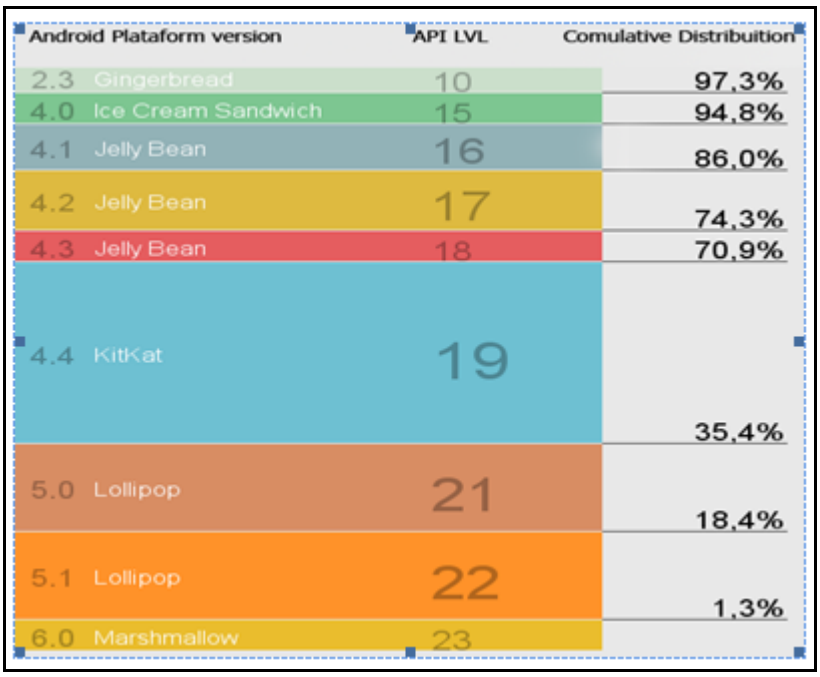
problemas de compatibilidade. Além disso, o SDK é compatível com grande parte dos sistemas operacionais existentes, com, por exemplo, da plataforma Windows (Vista, 7, 8 e 10), Mac OS X10.5.8 e superiores e claro diversas distribuições do Linux. Para o desenvolvimento normalmente são utilizados IDEs como Eclipse e Android Studio que fornecem suporte e integram todas as funcionalidades da plataforma SDK(LECHETA, 2013)

2.1.2 APIs de desenvolvimento

Com o crescente mercado de aparelhos Android e do desenvolvimento de aplicativos, a Google cria muitas plataformas diferentes para o Android e desta o sistema tornou-se compatível com aparelhos com grande desempenho e processamento e também com aparelhos mais simples, o que aumentou ainda mais o sucesso do sistema.

Cada plataforma recebe um código que é identificado pelo API *Level* e *também* um nome como, por exemplo, a API Leve 1.5 *Cupcake* e a API 6.0 Marshmallow. Na Figura 1 podemos visualizar as APIs existentes. Estas APIs são de extrema importância para os desenvolvedores de aplicativos Android, pois toda aplicação ao ser criada, deve especificar em qual API será desenvolvida e ao especificar esta API, o desenvolvedor esta escolhendo qual serão os aparelhos alvos para seu sistema, desta forma o grande dilema dos desenvolvedores Android é apresentado, pois quanto mais alta a API maior é o numero de funcionalidades e facilidades na programação do aplicativo, porém menor é a abrangência que seu produto terá no mercado, até que a API mais recente se popularize entre as empresas que criam os aparelhos (LECHETA, 2013).

Figura 1: Versões e porcentagem de compatibilidade.



Fonte: Retirada da ferramenta Android Studio, 2016.

2.2 DESENVOLVIMENTO DE APLICAÇÕES WEB

2.2.1 PHP

O *Personal* Home Page/*Forms Interpreter* como foi chamado em 1994, ano de sua criação, é uma linguagem de programação desenvolvida por Rasmus Lerdorf. Esta linguagem, inicialmente tinha o intuito de ser utilizada como uma ferramenta de uso pessoal porém, devido a um grande numero de recursos, grande diversidade e uma enorme facilidade de utilização acabou tornando-se uma das linguagens de programação mais utilizadas no mundo (NIEDERAUER, 2008).

De acordo com o site Netcraft (HTTP://www.netcraft.com), em 1999 pouco haviam mais de 50 mil domínios utilizando o PHP como linguagem e em um novo levantamento feito no ano de 2007 este numero ultrapassou 20 milhões de domínios. Nesta época o numero de scripts desenvolvidos na linguagem e disponíveis na internet já ultrapassava o de outras linguagens como Java, ASP Python e Perl (NIEDERAUER, 2008).

Atualmente o PHP encontra-se na versão 5, a qual foi lançada em julho de 2004, contando com o novo motor Zend Engine 2.0, que contém um novo modelo de objeto e dezenas de novas recursos. Desde 2004 foram lançadas várias versões maiores e dentro destas versões, varias *releases* deixando a linguagem cada vez mais robusta e completa.

2.2.2 Codeigniter

O Codeigniter é um framework para desenvolvimento PHP fundado em 28 de fevereiro de 2008 por Rick Ellis fundador da EllisLab. A partir de outubro de 2014, o framework foi vendido para BCIT(British Columbia Institute of Technology) que o mantém até os dias atuais. A versão mais recente encontrada é a 3.0.6 já mantida pela BCIT, porém a mais utilizada seja a 2.X desenvolvida pela EllisLab. Ambas as versões e suas *releases* estão disponíveis no site oficial do *framework* (HTTP://www.codeigniter.com).

O principal objetivo do Codeigniter é garantir às aplicações maior desempenho, capacidade e flexibilidade, utilizando ferramentas como instanciamento dinâmico, junção e ao mesmo tempo singularidade de componentes entre outras. Além de tudo isso, o framework foi desenvolvido baseando-se no padrão MVC, porém de forma muito mais dinâmica, flexibilizando a utilização dos C*ontrollers* sendo possível acessar e chamar diferentes *Views e Models.*

Para auxiliar no desenvolvimento e facilitar a usabilidade do framework, o Codeigniter conta com uma grande coleção de bibliotecas e *helpers* nativos como bibliotecas para calendários, bancos de dados, incluindo *drives* dos bancos mais utilizados atualmente, manipulação de imagens, controle de sessões entre outras (HTTP://www.codeigniter.com).

2.2.3 Bootstrap

O Bootstrap é um framework de código aberto voltado ao desenvolvimento *front-end*, sendo que é um dos mais populares e utilizados atualmente. Como qualquer framework o Bootstrap é um conjunto de arquivos HTML, CSS e

JavaScript, utilizados no desenvolvimento de aplicações web de forma fácil, rápida e visando principalmente o design e a responsividade das mesmas.

O framework foi lançado em agosto de 2011 pela equipe do Twitter, em um artigo que foi publicado na própria rede social. Neste artigo, Mark Otto e Jacob Thornton, relataram que no inicio do Twitter cada desenvolvedor utilizava uma biblioteca que tinha mais facilidade ou conhecimento para trabalhar no *front-end* e este procedimento acabou gerando inconsistências e dificultando as manutenções do site. Para resolver os problemas internos de desenvolvimento o Bootstrap foi criado (Silva, 2015).

O Bootstrap contém diversos componentes que podem ser utilizados na montagem de paginas web sendo eles separadamente ou em uma estrutura completa do framework. Para as estruturas mais complexas é aconselhável a utilização dos *templates* HTML do framework, onde estarão presentes as chamadas aos arquivos necessários para a estilização dos componentes como botões, links painéis e demais elementos HTML. Para as funções mais especificas ainda será imprescindível a chamada dos arquivos CSS e JavaScript das mesmas, nativos do framework, por exemplo janelas modais, *sliders* e barras de navegação. Na Figura 2 podemos visualizar o arquivo de estrutura inicial do framework.

Figura 2: Template Bootstrap.

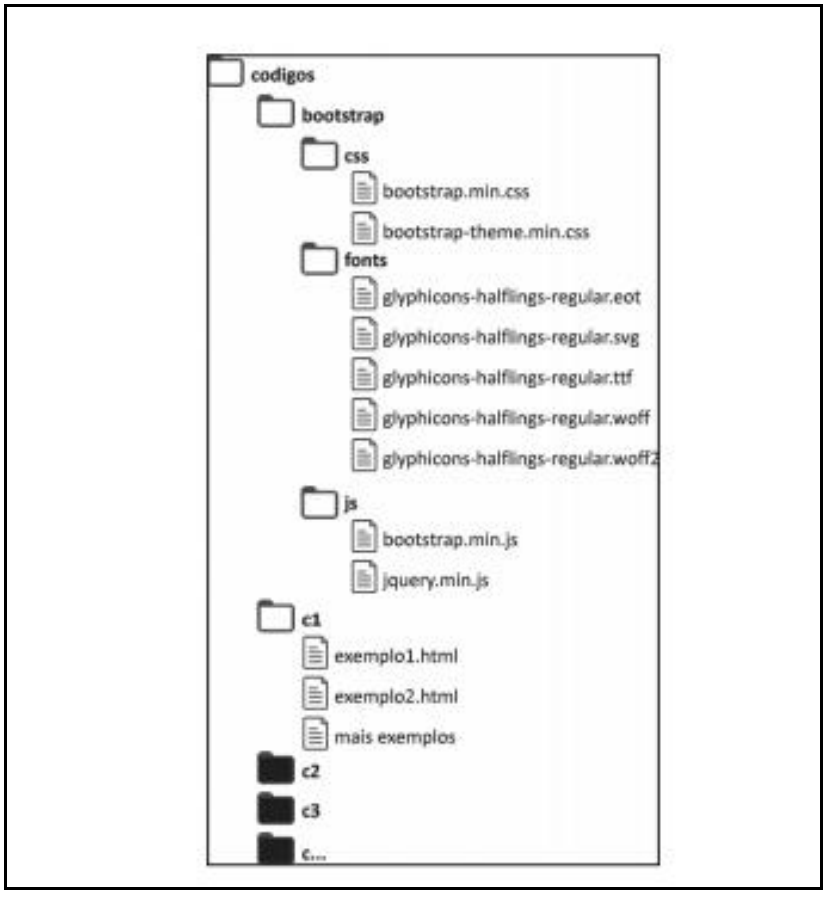
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <!-- Estas 3 meta tags deven obrigatoriamente ser as primeiras na seção head -->
      <!-- As demais devem vir depois delas -->
   <title>Template mínimo para uso do Bootstrap</title>
   <link href="../bootstrap/css/bootstrap.min.css" rel="stylesheet"> <!-- CSS do Bootstrap -->
   <!-- HTML5 shim and Respond.js para suporte dos elementos HTML5 e das media queries ao IE8 -->
   <!-- Atenção: Respond.js não funciona em páginas carregadas com uso do protocolo file:// -->
   <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
   <![endif]-->
</head>
<body>
   <h1>Título da aplicação</h1>
</body>
</html>
```

Fonte: (Bootstrap 3.3.5, SILVA Mauricio Samy).

Como o Bootstrap trabalha fortemente com marcações HTML e estilizações CSS, seu funcionamento depende da inclusão dos arquivos de estilo. Para que seja possível transformar um botão HTML comum em um botão estilizado são utilizados os atributos id e *class*, onde dependente dos valores informados, os elementos apresentarão um comportamento e estilo diferente. Estas definições aplicam-se também a funções JavaScript, como *clicks*, *changes*, entre outras (SILVA, 2015).

Ao baixar o framework o mesmo já vem em uma estrutura pré-configurada, demonstrada na Figura 3, contendo os principais arquivos de estilização, fontes e scripts separados e organizados cada qual em seu devido local. Desta maneira, a adição de novos componentes torna-se simples e rápida, fazendo com que a aplicação tenha grande escalabilidade e fácil manutenção (SILVA, 2015).

Figura 3: Estrutura de pastas Bootstrap.



Fonte: (Bootstrap 3.3.5, SILVA Mauricio Samy)

Atualmente o Bootstrap está na versão v3. 3.6 e contém milhões de sites e aplicações web rodando em sua estrutura, criando assim uma comunidade fortemente ativa o que só aumenta mais a sua popularidade no meio do desenvolvimento responsivo na WEB.

2.3 WEB SERVICES E COMUNICAÇÃO

Serviços web são componentes de softwares que podem ser armazenados em uma maquina e serem acessados por outros de seus componentes ou por outros softwares conectados por uma rede. Estes serviços são conectados utilizando tecnologias como HTTP, JSON e XML através do envio de solicitações entre cliente e servidor. Desta maneira aplicativos ou softwares que estão hospedados em dispositivos cujo processamento não é tão poderoso, podem ser capazes de obter dados resultantes de operações que demandam um grande poder de hardware, os quais são processados e transferidos para serem exibidos posteriormente (DEITEL, 2010).

2.3.1 RESTFULL WEB SERVICE

Com o surgimento da Web 2.0 o desenvolvimento web precisou tornar-se mais pratico e ágil e com o aumento da tecnologia móvel a integração entre aplicativos e plataforma web precisou ser ainda mais rápida e simples. Em vista disso o RESTfull se popularizou cada vez mais (GONCALVES, 2013).

O REST (Representational State Transfer) foi desenvolvido como tese de doutorado de Roy Fieding. Diferente do SOAP que foi desenvolvido para trabalhar com diversos protocolos o REST é uma arquitetura que realiza a manipulação de dados e recursos, explorando do protocolo HTTP e URL podendo dar as respostas em múltiplos formatos e simplicidade de regras (DEITEL, 2010).

Com a utilização do protocolo HTTP e das URLs, o REST disponibiliza acesso aos dados de maneira segura, pois quando um cliente solicita acesso a estes dados recebe uma representação dos mesmos, que pode se dar através de arquivos JSON, XML, HTML entre outros. Estas representações estarão sempre disponíveis

por meio das requisições URI geradas pelo cliente para o servidor. Desta maneira o consumo e a carga no servidor são melhores distribuídos já que os recursos serão requisitados e gravados em um arquivo que poderá ser acessado sem que seja necessário fazer uma nova busca aos dados reais do sistema (GONÇALVES, 2013).

Por trabalhar com as URIs do protocolo HTTP o REST está sujeito aos métodos padrões que o protocolo implementa, desde os mais conhecidos GET e POST, e também métodos mais específicos como PUT, DELETE, HEAD e OPTIONS, cada qual com um funcionamento que pode ser aproveitado na utilização da tecnologia REST.

1. GET: Realiza a recuperação das informações levando em conta uma identificação pela URI sem realizar modificações nas informações do sistema.
2. POST: Através da URI passada, adiciona informações ao sistema, podendo alterar informações já existentes ou criando novos registros.
3. PUT: Da mesma forma que o POST, adiciona ou modifica um recurso da URI, com a diferença que no POST a URI representa o local onde será tratada a informação e no PUT a URI representa o local onde a informação será armazenada.
4. DELETE: Realiza a remoção de um recurso representado pela URI transmitida.
5. HEAD, OPTIONS e TRACE: São responsáveis pela recuperação de metadados da URI, como cabeçalhos e opções que devem ser lidas e interpretadas em um determinado serviço.

Através dos métodos padrões do HTTP sendo transmitido por uma URI, um serviço REST bem feito pode muito bem ser capaz de realizar a interação entre dois sistemas de forma simples e eficiente, o que torna a tecnologia uma grande ferramenta de intercomunicação (TILKOV, 2008).

2.3.1 Slim Framework PHP.

O Slim é um framework baseado na linguagem PHP e é utilizado para o desenvolvimento de APIs e aplicações com comunicação WebService. Sua versão atual é a Slim 3.4.1. Esta ferramenta trabalha muito bem com o protocolo RestFULL

tendo em vista que implementa as funções básicas do HTTP GET, POST, DELETE, PUT entre outras (SlimFramework, 2016).

De uma forma simples e fácil o Slim disponibiliza funções especificas para trabalhar com cada um dos métodos HTTP, para que possam ser acessadas por outras aplicações de forma mais abstrata.

O Slim fornece ao desenvolvedor um nível de abstração muito grande no uso dos métodos HTTP, conforme demonstrado na Figura 4. Para isso é necessário utilizar uma estrutura simples que consiste na declaração do método por meio de um Objeto do tipo SlimApp. Este objeto poderá implementar os métodos GET, POST, DELETE etc. Em sua declaração são passados 2 parâmetros, o primeiro consiste em uma *string* que é a rota de onde ele poderá ser acessado por outras aplicações, sendo que em métodos como GET e DELETE, por exemplo, podem ser adicionados dados entre chaves "{}", que serão atribuídos à variável $args e servirão para especificar as ações que o método deverá realizar. O segundo parâmetro é uma função *CallBack*, que é onde será escrita a programação do método para que o WebService possa realizar alguma ação junto ao banco de dados, por exemplo (SlimFramework, 2016).

Figura 4: Exemplo Implementação métodos GET, PUT e POST via SlimFramework

```php
$app = new \Slim\App();
$app->get('/books/{id}', function ($request, $response, $args) {
    // Show book identified by $args['id']
});
```

```php
$app = new \Slim\App();
$app->post('/books', function ($request, $response, $args) {
    // Create new book
});
```

```php
$app = new \Slim\App();
$app->put('/books/{id}', function ($request, $response, $args) {
    // Update book identified by $args['id']
});
```

Fonte: Slim Framework, 2016.

2.3.2 JSON

JSON (JavaScript Object Notation) é um formato de transmissão de dados entre sistemas onde os dados são representados por conjuntos chave/valor, formando assim uma estrutura organizada de apresentação de informações. O JSON é uma alternativa ao XML, sendo mais simples e de rápida interpretação por serviços como o REST (GONÇALVES, 2013).

Os arquivos JSON podem representar dados em formatos primitivos o que facilita muito a integração com diversas APIs. São estes formatos:

- *Number:* o formato *number* no JSON é muito parecido com o existente em linguagens fortemente tipadas como Java, com a exceção de formatos octais e hexadecimais.
- String: Uma *string* é uma sequência de zero ou mais caracteres envolvidos em aspas duplas
- *Value:* Um *value* em JSON é um formato que está vinculado a outros como uma *string* entre aspas, um número, booleanos *true*, *false* e *null*, em objetos, ou em *arrays.*
- *Array*: São conjuntos ordenados de valores representados por estruturas entre colchetes ([,]) e com seus valores separados por vírgulas (,), também podendo representar objetos como valores.
- Objeto: Com a mesma definição para objetos que a linguagem JAVA os objetos em JSON são conjuntos ordenados ou não de pares chave/valor, onde chaves ({,}) marcam o início e fim de um objeto e sua separação é feita por vírgula (,).

Com estes diversos formatos em JSON é possível realizar a serialização dos dados fazendo uma representação extremamente simples e completa destes dados. Através da representação de variáveis entre chaves, o que representa a criação de um objeto. Com a criação de objetos em JSON é possível definir seus atributos e valores representando assim qualquer estrutura, o que dá a esta tecnologia a

capacidade de ser utilizada para representação de dados, desde os mais simples até dados gerados por softwares e aplicativos (GONÇALVES, 2013).

Na Figura 5 podemos visualizar a estrutura básica de um arquivo JSON.

Figura 5: Exemplo de arquivo JSON

```
{
  "order": {
    "id": "1234",
    "date": "05/06/2013",
    "customer": {
      "first_name": "James",
      "last_name": "Rorrison",
      "email": "j.rorri@me.com",
      "phoneNumber": "+44 1234 1234"
    },
    "content": {
      "order_line": [
        {
          "item": "H2G2",
          "quantity": "1",
          "unit_price": "23.5"
        },
        {
          "item": "Harry Potter",
          "quantity": "2",
          "unit_price": "34.99"
        }
      ]
    },
    "credit_card": {
      "number": "1357",
      "expiry_date": "10/13",
      "control_number": "234",
      "type": "Visa"
    }
  }
}
```

Fonte: GONÇALVES 2013

3 METODOLOGIA

Para o alcance dos objetivos este projeto foi desenvolvido em três etapas:

- A primeira etapa consiste no aprofundamento de conhecimentos sobre as tecnologias que foram empregadas. Desta etapa resultará o desenvolvimento do referencial teórico do trabalho.

- A segunda etapa se refere ao estudo de aplicativos semelhantes para a plataforma Android. Nessa etapa os requisitos e diferenciais do aplicativo são identificados.
- A terceira etapa se dá com o desenvolvimento do protótipo. As tecnologias estudadas foram empregadas com para o desenvolvimento do aplicativo.

3.1 ESTUDO DE CASO

O principal objetivo deste trabalho é o desenvolvimento de um conjusnto de aplicativo móvel para a plataforma Android, bem como interface web, capaz de realizar o controle de horário ponto e da geolocalização dos usuários. Nesta seção será apresentado um estudo de caso abordando os motivos que tornam necessária a criação do presente projeto juntamente com uma análise de aplicativos já existentes, que contenham funções similares as que se propõem desenvolver.

3.1.1 Contexto

Nos tempos atuais, a computação móvel esta cadá vez mais presente na vida das pessoas, por meio de *smartphones, tablets* e uma infinidade de outros dispositivos. Através destes dispositivos, tarefas simples como fazer anotações, saíram do papel e passaram ao universo digital, aumentando consideravelmente a praticidade, facilidade e velocidade de acesso as informações.

Com este grande avanço nas tecnologias, abrem-se possibilidades. O projeto se propõe a realizar a automatização de uma atividade muito comum para a grande maioria das empresas e seus funcionários: o registro de horário ponto durante a jornada de trabalho, porém com o diferencial de promover também o controle e registro da localização geográfica dos usuários.

3.1.2 Aplicativos semelhantes

Tendo em vista que o projeto foi desenvolvido visando à plataforma Android, foram realizadas buscas na PlayStore, loja de aplicativos do Google, onde foram

encontrados diversos aplicativos que trabalham com o registro de horário ponto e um deles também armazena a geolocalização.

A. Ponto Fácil (Banco de Horas) – Neto Rebouças

O aplicativo Ponto Fácil (Figura 6) foi desenvolvido por Neto Rebouças, visa o usuário final, possibilitando que os registros dos horários seja feito de maneira independente em relação às empresas. Com mais de 1700 downloads é um aplicativo intuitivo e de fácil operação que apresenta as seguintes funcionalidades:

- Registro de entrada e saída.
- Backups e exportação dos dados.
- Controle de faltas, feriados e férias
- Padrões de horários iniciais e finais
- Relatórios por dia/semana/mês.
- *Widgets* para registro rápido.

Figura 6: Seleção de período para relatórios.

Fonte: Neto Rebouças, Google Play.

Figura 7: Widgets de registro de ponto e saldo de horários.

Fonte: Neto Rebouças - Google Play, 2016.

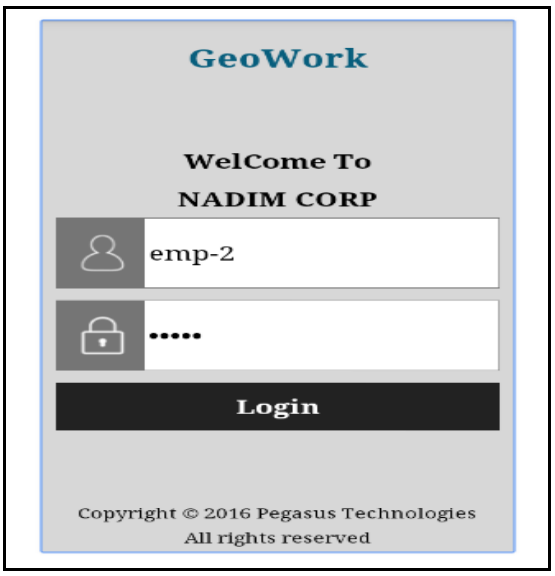
B. GeoWork – Pegasus Tecnologies.

GeoWork é um aplicativo desenvolvido pela Pegasus Tecnologies, que conta com a tecnologia tracker. Sua tecnologia visa a realização do controle das atividades dos usuários, rastreamento dos telefones celulares dos funcionários, fornecendo sua localização precisa, além de contar com um muito baixo consumo de bateria. Para realizar este rastreamento o APP conta com o uso das APIs de GPS, Wi-Fi e sinais de rede. Abaixo seus diferenciais:

- Funciona sem Internet, enviando os resultados de localização quando for detectada rede;
- Rastreia os funcionários;
- Envia notificações aos funcionários;
- Possibilita geração de relatórios no APP;

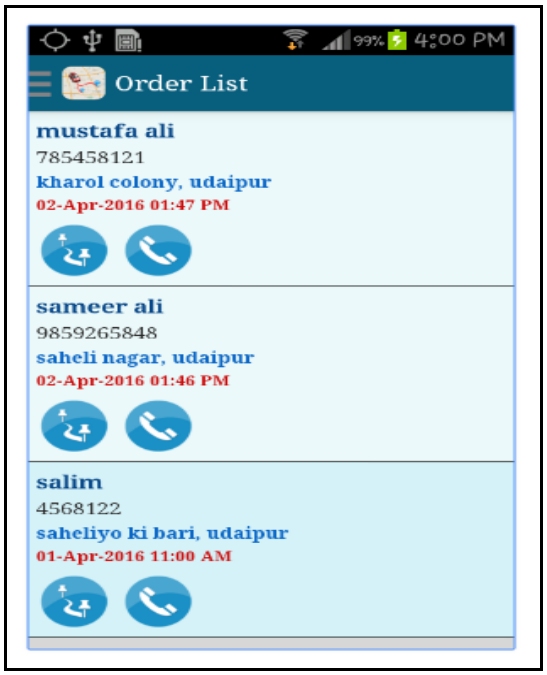
O aplicativo GeoWork se assemelha muito à proposta do presente projeto, principalmente nas funcionalidades básicas. O aplicativo é disponibilizado para funcionários de empresas especificas, conforme demonstrado nas Figuras 8 e 9. Infelizmente, pelo fato de o aplicativo ser disponibilizado por demanda à empresas contratantes do serviço, não foi possível testá-lo, já que não se dispõe de um usuário válido.

Figura 8 : Login do funcionário.



Fonte: Pegasus Technologies Google Play, 2016

Figura 9: Dados do registro de horários e localização.



Fonte: Pegasus Technologies, Google Play.

3.2 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos funcionais são responsáveis pela definição direta das funções que um sistema deve realizar. Já os requisitos não funcionais são aqueles que estão relacionados a utilização da aplicação, seu desempenho, usabilidade e tecnologias envolvidas. Baseando-se nos conceitos acima, a presente seção descreverá estes requisitos, tanto da aplicação web quanto do aplicativo móvel.

3.2.1 Requisitos funcionais – aplicativo

- R1 – Realizar o *login* do funcionário no aplicativo por telefone e senha via web service;
- R2 – Manter horário ponto e localização;
- R3 – Transmitir horários não enviados anteriormente;
- R4 – O usuário só poderá estar logado em um único dispositivo por vez;
- R5 – Exibir visualização do horário iniciado ou encerrado;

3.2.2 Requisitos não funcionais – aplicativo

- RNF 1 – O aplicativo deve salvar os dados em um banco de dados local no dispositivo e sincronizá-los com um banco de dados da aplicação web;
- RNF 2 – A interface do aplicativo deve ser simples e intuitiva;

3.2.3 Requisitos funcionais – aplicação web

- RF1 - Realizar o *login* do usuário da gestão da empresa por e-mail e senha.
- RF2 – Manter funcionários;
- RF3 – Manter usuários;
- RF4 – Manter Cidades;
- RF5 – Manter Estados;
- RF6 – Gerar Relatórios de horário dos funcionários;
- RF7 – Manter Empresas.

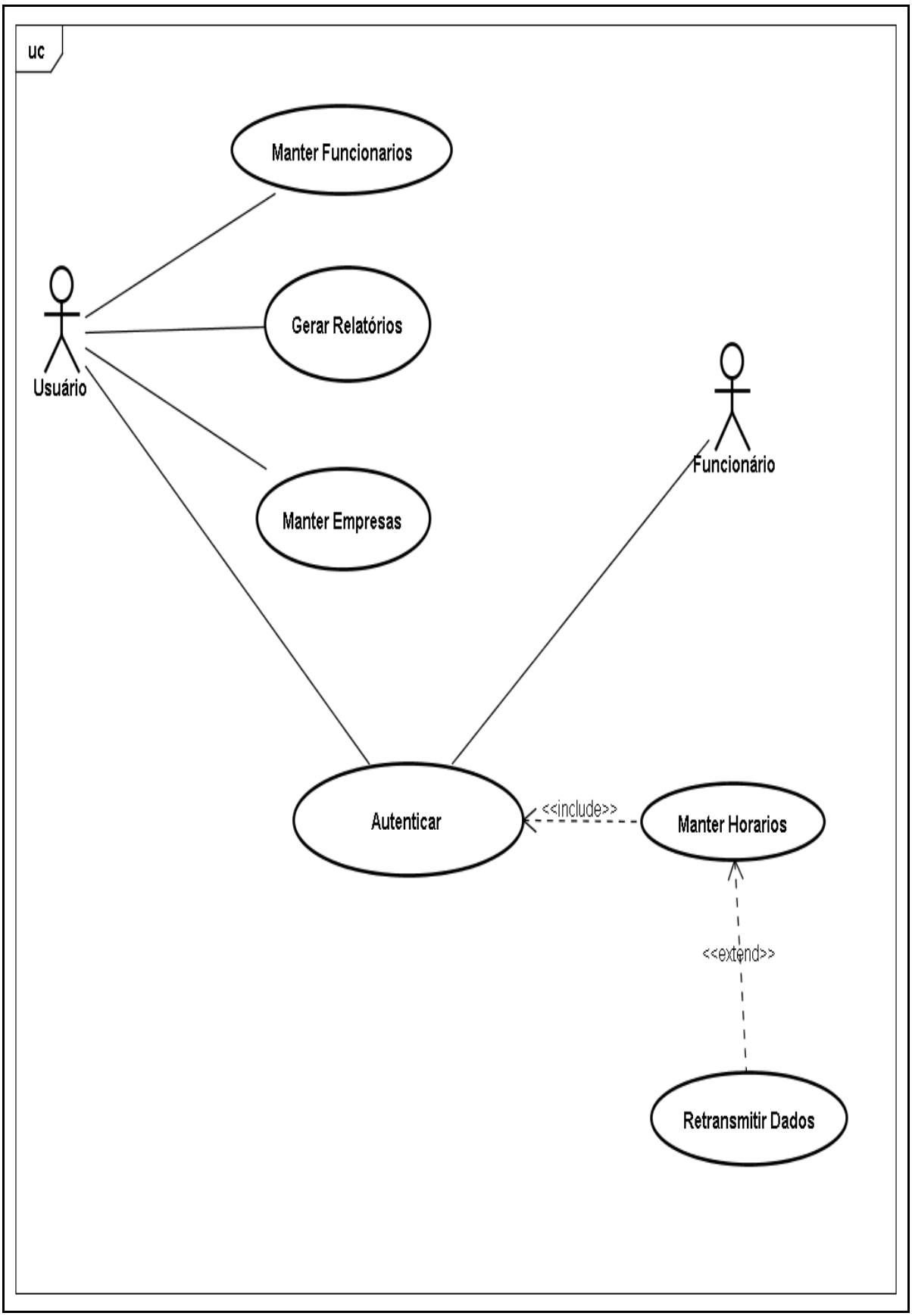
3.2.4 Requisitos não funcionais – aplicação web

- RFN1 – O sistema web deverá ter a interface simples e intuitiva.
- RFN2 – O sistema web deverá ter a interface responsiva.
- RFN3 – O sistema deve ter um banco de dados para armazenar os dados.

3.3 DIAGRAMAS DE CASO DE USO

Os diagramas de caso de uso têm a função de documentar o sistema e as funcionalidades que o usuário terá acesso, descrevendo os principais processos do sistema. Nesta seção serão descritos os diagramas da aplicação web e do aplicativo, iniciando pelo diagrama de casos de uso representado na Figura 10.

Figura 10: Diagrama de Casos de uso



Fonte: do Autor.

3.4 DESCRIÇÃO DOS CASOS DE USO

3.4.1 Autenticar Funcionário

Nome do Caso de Uso	**Autenticar Funcionário**
Ator Principal	Funcionário
Atores Secundários	
Resumo	Este caso de uso descreve o processo de um funcionário. O funcionário informa seus dados, que são consultados via WebService no banco de dados principal.
Pré-Condições	Conexão com a Internet
Pós-Condições	Autenticação correta ou não.
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Informar número do aparelho e senha	
	2. Consultar no banco de dados principal por meio do Webservice
	3. Se os dados informados corresponderem a um funcionário cadastrado no banco principal, o acesso ao aplicativo será concedido. Caso contrario o sistema deve informar que dados estão incorretos
Restrições e validações	1. Funcionário precisa estar cadastrado
Estrutura de Dados	
1. Número do Aparelho	
2. Senha do usuário	

3.4.2 Manter Funcionário

Nome do Caso de Uso	**Manter Funcionário**
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve o processo de criação e manutenção dos funcionários que utilizarão o aplicativo móvel de horário ponto.
Pré-Condições	Usuário deve estar logado. Existir empresa cadastrada.
Pós Condições	Os números de aparelhos não devem se repetir
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Informar dados do funcionário: Nome, Endereço, Cidade, Estado, telefone, e-mail, status. Logo após deve informar os dados de seu aparelho que serão utilizados na autenticação do aplicativo: Numero telefone, senha, operadora, status, data de cadastro.	
	2. Gravar dados do funcionário no banco de dados.
	3. Direcionar para a lista de funcionários.
Restrições e validações	1. Funcionário não pode ter e-mail repetidos. 2. O número do aparelho não pode ser igual a outros já cadastrados.
Estrutura de Dados	

Funcionário 1 Nome 2 Endereço 3 Telefone 4 E-mail 5 Status.	Aparelho do funcionário 1 Número telefone 2 Senha 3 Operadora 4 Status 5 Data de cadastro.

3.4.3 Manter Relatórios

Nome do Caso de Uso	**Manter Relatório**
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve o processo de geração dos relatórios, que serão alimentados pelos dados obtidos pela comunicação do WebService com o aplicativo.
Pré-Condições	1 Usuário deve estar conectado ao sistema. 2 Os dados devem ter sido transmitidos pelos funcionários no aplicativos.
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1 Informar os filtros de busca dos funcionários que se deseja.	
	2. Acessar banco de dados gerado pelo aplicativo com os dados de horário ponto e coordenadas dos funcionários
	3. Consultar através da API do Google a cidade, bairro e endereço, utilizando as coordenadas de latitude e longitude

Restrições e validações	1. Realizar o filtro de busca. 2. Caso o usuário não informe o filtro, trazer por padrão o filtro do mês atual para todos os funcionários.
Estrutura de Dados	
1. Dados dos filtros	
2. Dados da tabela GPS	
3. Dados WebService API do Google.	

3.4.4 Manter Horários

Nome do Caso de Uso	**Manter Horários**
Ator Principal	Funcionário
Atores Secundários	
Resumo	Este caso de uso descreve o processo em que os funcionários registram os horários de entrada e saída bem como a sua localização.
Pré condições	1)Estar logado no sistema. 2)Conexão de rede.
Pós condições	Transmitir para o banco de dados principal
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Acionar registro início de horário	2. Consultar horário do celular. 3. Consultar localização via API de GPS, WIFI, ou 3G 4. Gravar no banco local 5. Enviar dados via WebService para banco de dados web.

6. Acionar registro de fechamento de horário.	2. Consultar horário do celular. 3. Consultar localização via API de GPS, WIFI, ou 3G 4. Consultar ponto aberto no banco local 5. Enviar via dados WebService para banco de dados web.
Estrutura de dados.	
1. Botão de inicio e fim de horário. 3. Botão Retransmitir.	2. Horário ponto não finalizado.

3.4.5 Manter Empresas

Nome do Caso de Uso	**Manter Empresas**
Ator Principal	Usuário
Atores Secundários	
Resumo	Este caso de uso descreve o processo manutenção dos dados da empresa, feito pelos usuários da aplicação
Pré-condições	1. Estar logado no sistema.
Pós-condições	Validação de CNPJ
1. Informar dados da empresa: Nome, Razão social, CNPJ, Endereço, Cidade, Estado, telefone, e-mail, status e contato responsável.	2. Gravar dados da empresa no banco de dados. 3. Direcionar para cadastro de funcionários.
Restrições e validações	

1. CNPJ deve ser valido. 2. Cadastrar Funcionários
Estrutura de dados.
1. Dados da empresa: Nome, Razão social, CNPJ, Endereço, Cidade, Estado, telefone, e-mail, status e contato responsável. 2. Botões salvar e Cancelar.

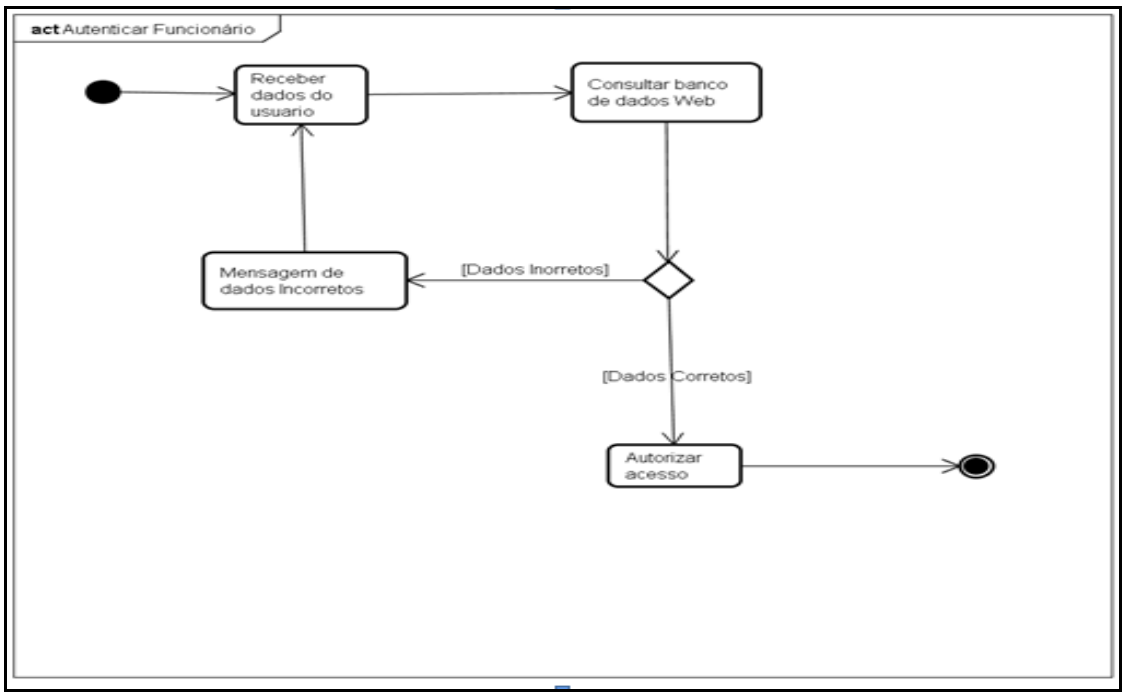
3.5 DIAGRAMA DE ATIVIDADES

Os diagramas de atividades são elementos da UML semelhantes aos fluxogramas e são utilizados para descrever o fluxo de atividades em um sistema desde o início até o fim da mesma.

Nesta seção serão apresentados os diagramas de algumas das principais atividades que a aplicação a ser desenvolvida se propõe a fazer, demonstrados através das Figuras 11, 12 e 13.

3.5.1 Diagrama de Atividade – Autenticar Funcionário

Figura 11: Diagrama de Atividades - Autenticar Funcionário.



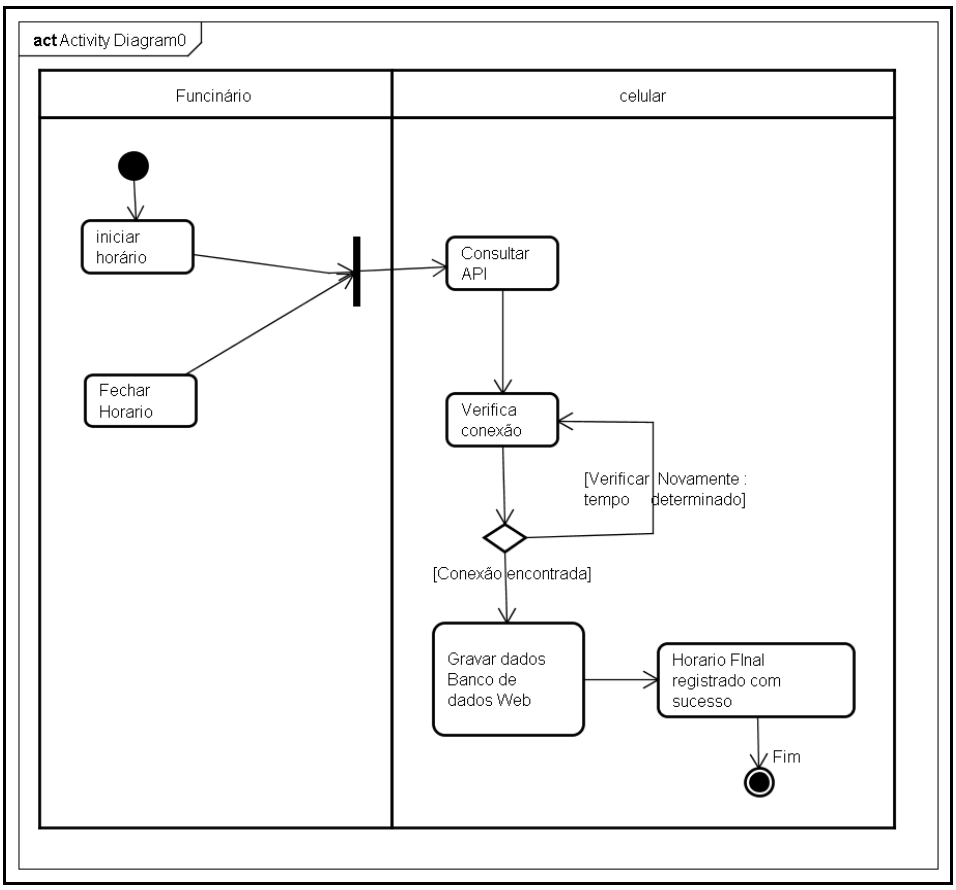
Fonte: do Autor.

O aplicativo que foi desenvolvido deverá armazenar o horário e localização dos funcionários. Para isso será necessário que o funcionário informe o número do telefone e uma senha que serão consultados no banco de dados principais através de um *Web Service*. Após a confirmação dos dados o funcionário obterá acesso à aplicação completa para poder registrar seus horários.

Caso os dados informados pelo funcionário não sejam compatíveis com os dados encontrados no banco de dados web, o aplicativo deverá retornar ao formulário de login informando que dados estão incorretos.

3.5.2 Diagrama de Atividade – Manter Horários.

Figura 12: Diagrama de Atividades - Manter Horários.



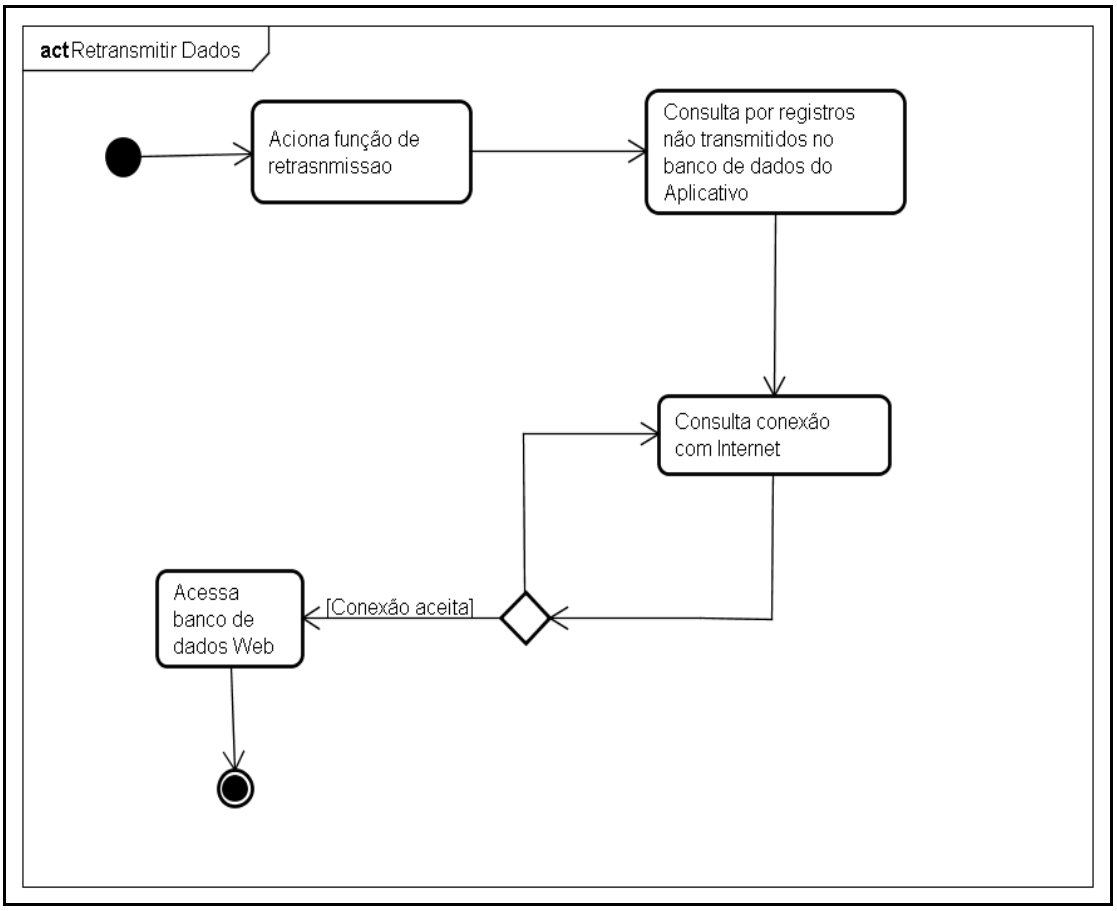
Fonte: do autor.

Da mesma forma que na autenticação, os funcionários utilizarão o aplicativo para registrar o horário em que iniciam e finalizam suas atividades, salvando também sua localização. Para isso, será acessado a API de GPS do aparelho celular do funcionário, ao buscar a localização a mesma será armazenada no banco de dados do celular juntamente com o horário. Coletadas as informações o aplicativo realizará uma conexão no banco de dados web através de um W*ebService*. Caso não haja conexão com a internet o aplicativo deve manter o registro em aberto para que possa ser transmitido em outro momento.

Após a realização do inicio do horário, o aplicativo deverá habilitar o botão de finalizar horário e este deve executar o mesmo processo do iniciar, possibilitando assim o fechamento das atividades do funcionário.

3.5.3 Diagrama de Atividade – Retransmitir Dados

Figura 13: Diagrama de Atividades - Retransmitir Horários.



Fonte: do autor

Este diagrama demonstra o fluxo de atividade do processo de retransmissão dos dados. Devido a se tratar de um aplicativo móvel é possível que em algum momento o funcionário esteja sem acesso à internet. Para resolver este problema, o aplicativo deverá disponibilizar a funcionalidade de retransmitir dados, desta forma não há riscos de que dados sejam perdidos. O aplicativo deve consultar o banco de dados local encontrando os registros que não foram enviados via WebService para o banco de dados web.

Caso ocorra algum problema de conexão ou na transmissão, o sistema deverá informar para que seja possível retransmitir novamente em outro momento.

3.6 Modelagem Banco de dados web

Com base nos requisitos para aplicações web e nas necessidades que foram levantadas para o desenvolvimento da mesma, foi projetado um diagrama de entidade e relacionamento, representado na Figura 14, onde buscou-se entender estes requisitos e necessidades e documentar as tabelas e campos necessários para que o banco de dados da aplicação web possa dar condições para o seu desenvolvimento.

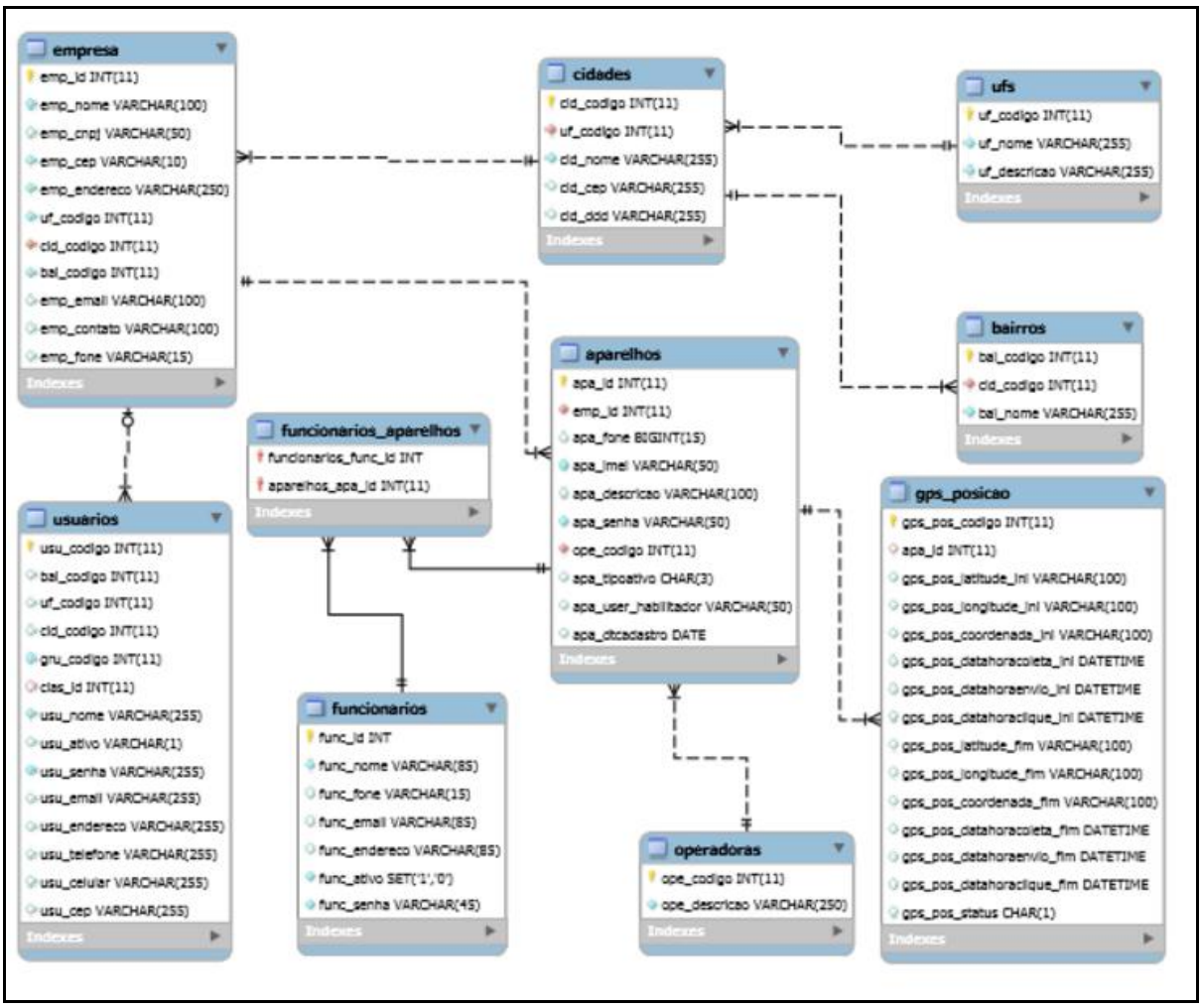
Foram modeladas todas as tabelas, desde tabelas auxiliares como “cidade”, “estado” e “operadoras”, até as tabelas que darão forma ao sistema, como as tabelas de “empresa”, “funcionário” e “aparelho”. Dentre todas as tabelas, a mais importante na aplicação será a tabela “gps_posicao”, pois é nesta entidade que serão armazenados todos os dados capturados pelos funcionários em campo, como horários localização, além de alguns campos gerenciais para controle dos registros abertos e fechados.

Outras entidades importantes são as de “aparelhos” e “funcionário” as quais formam um relacionamento que será consultado via WebService para autenticar o funcionário no aplicativo celular, para que o mesmo possa gerar os dados para a tabela “gps_posicao”.

Por fim, através das entidades “empresa” e “usuário” serão criados os dados das empresas que utilizarão o sistema como um todo.

Através dos usuários cadastrados na aplicação web será possível realizar manutenções nos cadastros de funcionários e empresas, bem como gerar relatórios a partir dos dados enviados pelos funcionários que utilizam o aplicativo móvel.

Figura 14: Diagrama Entidade Relacionamento – Bando de Dados Web



Fonte: do autor.

Com base nesta estrutura de dados, pretende-se realizar todas as ações necessárias para a plena utilização do sistema, tanto pelos envolvidos diretos com a aplicação web, usuários e empresas, como para os funcionários, que por meio da comunicação WebService, alimentam o banco de dados.

3.7 TESTES

Após o estudo das tecnologias e o desenvolvimento dos componentes do projeto, aplicativo, comunicação e interface web, os mesmos foram testados seguindo o um roteiro de testes.

- Aplicativo móvel

    - Registro de horários;
    - Acesso a APIS de localização;
    - Coordenadas geográficas;
- *WebServices*
    - Autenticação de funcionários;
    - Transmissão de dados no ato do registro;
    - Transmissão de dados automatizada;
    - Transmissão de dados forçada;
- Interface Web
    - Cadastros e manutenções do sistema;
    - Geração de relatórios;
    - Consulta a API de decodificação de coordenadas (Google);

4 DESENVOLVIMENTO

Nesta seção serão retratadas algumas informações sobre o desenvolvimento do projeto como, por exemplo, sua estrutura de arquivos, ambiente, onde e como será disponibilizado. Também serão retratados alguns pontos específicos do funcionamento do sistema como um todo, tanto aplicativo quanto interface web, e sua comunicação.

4.1 PREPARAÇÃO DO AMBIENTE.

Conforme descrito na seção de metodologia, foram desenvolvidos duas áreas distintas que farão a composição do AppWork. Primeiramente uma plataforma web que servirá como interface aos usuários da gestão e também como repositório principal dos dados.

Além desta plataforma web, foi desenvolvido um aplicativo que fará a alimentação dos dados principais do banco principal. Esta alimentação se dará através de uma comunicação WebService desenvolvida em paralelo as duas áreas principais.

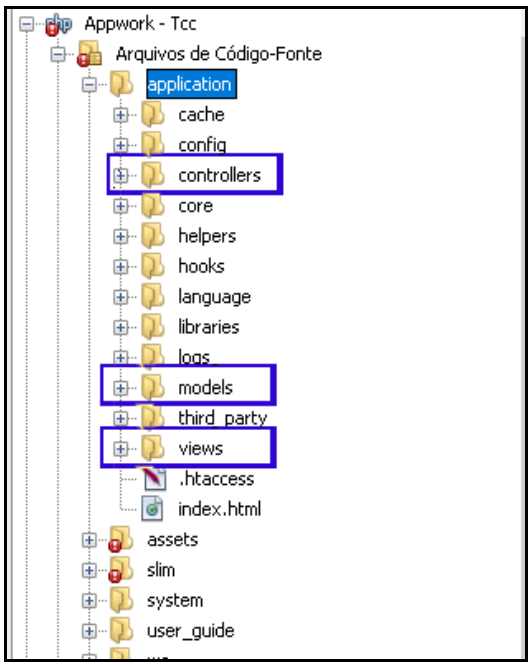
4.2 APLICAÇÃO WEB

4.2.1 Arquitetura e Configurações.

O desenvolvimento do projeto iniciou-se pela interface web, a qual tem a função de realizar os cadastros dos usuários, funcionários e aparelhos, informações da empresa, o relatório de horário e localização dos funcionários. Para esta etapa foi necessária a utilização do servidor local XAMPP, onde foi criada toda a estrutura da aplicação contendo o banco de dados, desenvolvido com o MySql e parte da codificação onde foi utilizado a Linguagem PHP integrada com HTML, CSS e JavaScript.

Foi utilizado o framework Codeigniter, que é uma ótima ferramenta para criar aplicações baseadas no Modelo MVC, separando em camadas o HTML(view), operações lógicas do sistema (controller) e operações com banco de dados (model). Este framework utiliza a linguagem PHP e é uma ótima opção para criar projetos bem organizados e facilmente gerenciados. Conforme exemplificado na Figura 15 pode-se entender o nível organizacional de um projeto que mantenha os padrões MVC.

Figura 15: Estrutura do projeto web.

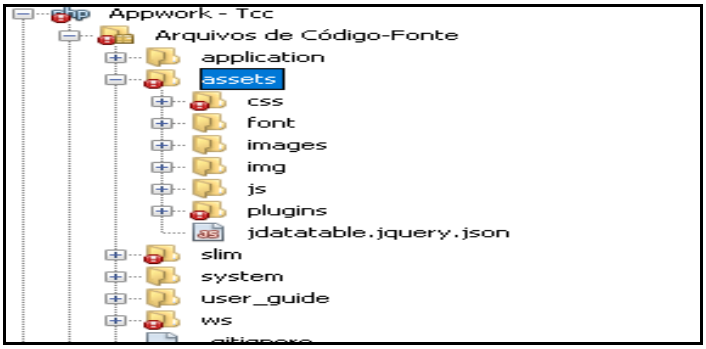


Fonte: Do autor

Conforme citado no referencial teórico, foram utilizadas as bibliotecas de HTML, CSS e Javascript do Bootstrap, a fim de proporcionar ao usuário uma experiência mais agradável na utilização da aplicação.

A estrutura da utilização desta tecnologia esta exemplificada na Figura 16 que demonstra a organização das pastas com os arquivos necessários para a utilização da biblioteca. Além disso, faz-se necessária, também a chamada destes componentes dentro das views onde estes serão renderizados e utilizados.

Figura 16: arquitetura de bibliotecas CSS, JS



Fonte: do autor.

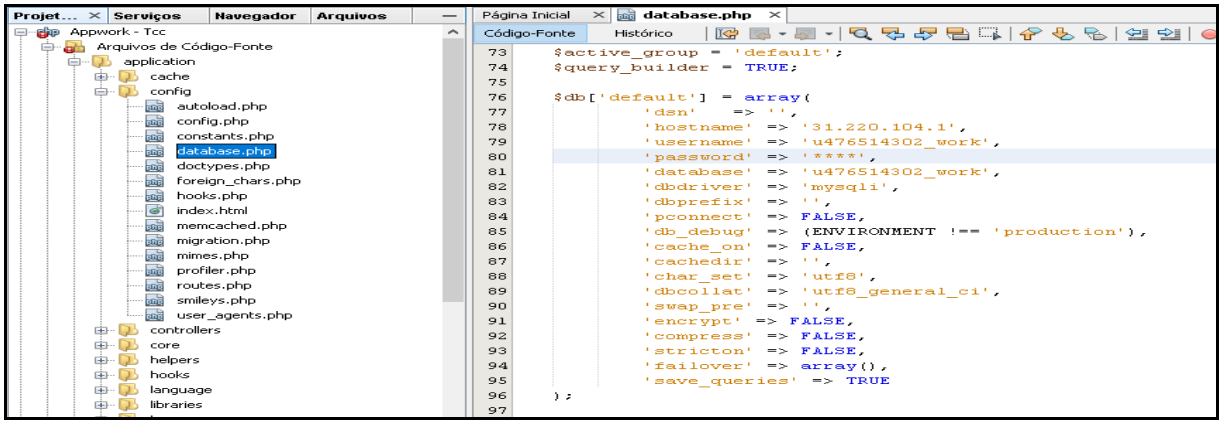
O banco de dados para a aplicação seguiu a modelagem ER (especificada previamente) e foi desenvolvido no MySql. Neste banco de dados estão presentes as tabelas de todas as manutenções do sistema, bem como a tabela de "gps_posicao" que contém os dados captados pelo aplicativo, conforme demonstrado na Figura 17.

Figura 17: Banco de dados principal

Fonte: do autor

Inicialmente o projeto foi desenvolvido todo em ambiente local, utilizando o XAMPP como servidor de arquivos e de banco de dados, porém posteriormente surgiu a necessidade de implantar o banco de dados em um servidor web, a fim de que o aplicativo pudesse acessá-lo e gerar os enviar os dados captados. Para tal, foi necessário mudar as conexões de banco de dados. Este procedimento é realizado dentro de um dos arquivos de configuração do framework codeigniter chamado Database, conforme demonstrado na Figura 18.

Figura 18: Configuração do acesso ao Banco de dados.



Fonte: do autor

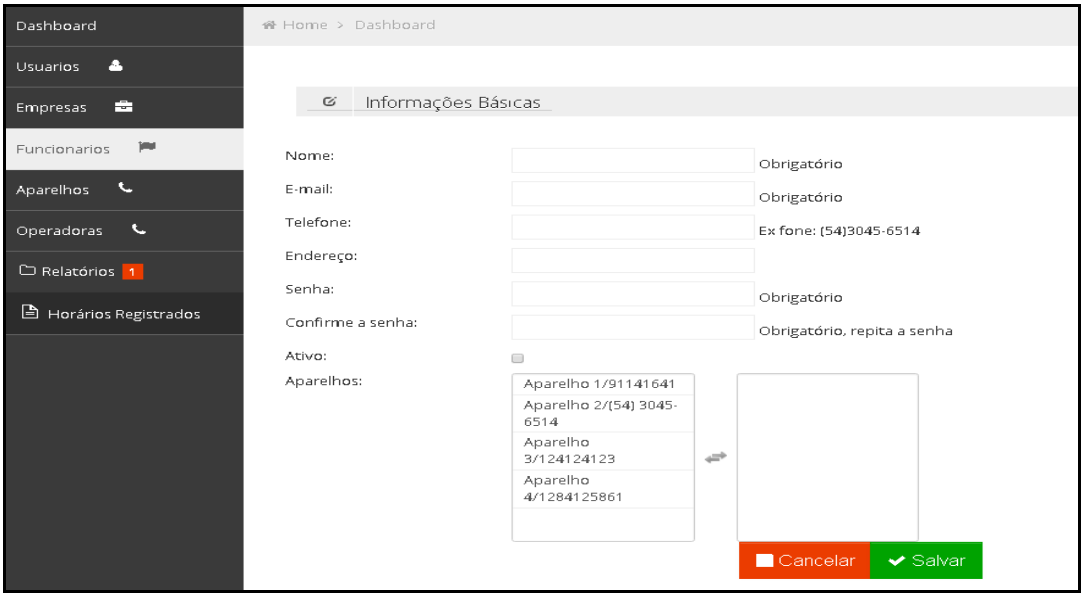
4.2.2 Funcionalidades

Para o pleno funcionamento da aplicação web foram desenvolvidos módulos divididos em: funcionários, aparelhos, empresas, relatórios entre outros. Alguns deles são imprescindíveis para a utilização da aplicação como, por exemplo, aparelhos e funcionários, onde são cadastrados os aparelhos que estão autenticados a utilizar o aplicativo móvel, e também os funcionários que utilizarão estes aparelhos.

4.2.3 Módulo de funcionários

No módulo de funcionários foi desenvolvida a lista onde é possível fazer buscas, excluir funcionários, acessar os formulários de alteração e inserção de novos funcionários. Destacam-se campos como e-mail e senha, onde posteriormente pode-se desenvolver uma área na aplicação para o acesso dos funcionários as suas informações. A funcionalidade mais importante neste cadastro trata-se do vinculo do funcionário aos aparelhos, já que é desta forma que o funcionário terá acesso ao aplicativo podendo gerar as informações de horário e localização. Neste campo só é permitido adicionar aparelhos que ainda não estejam vinculados a outros funcionários. Demonstração da tela de cadastro de usuários na Figura 19.

Figura 19: Cadastro de Funcionários

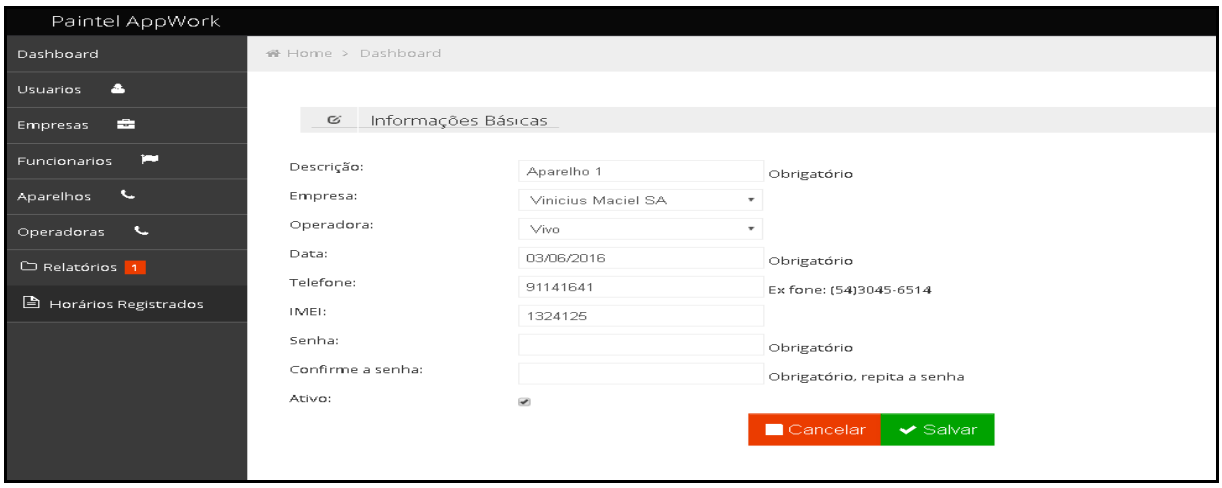


Fonte: do autor

4.2.4 Módulo de aparelhos

Neste módulo serão cadastradas as informações dos celulares que geram os dados para o banco principal. É composto por dados como o número do aparelho e senha, que concedem o acesso a área interna do aplicativo móvel. Demais dados como os da operadora, da empresa responsável e numero EMEI, servirão como informações complementares para este módulo. Tela de cadastro na Figura 20.

Figura 20: Cadastro de Aparelhos

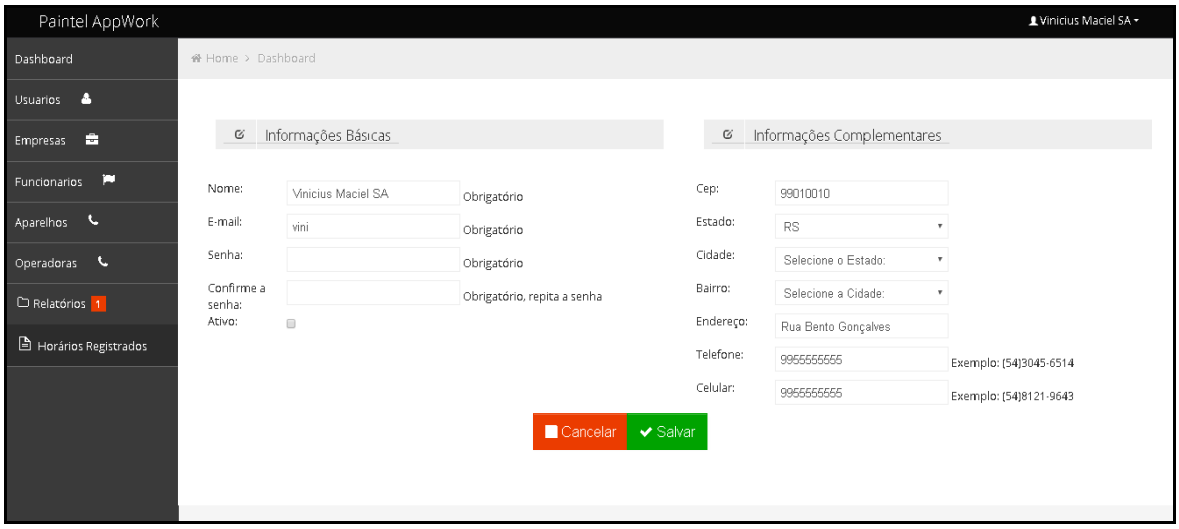


Fonte: do autor

4.2.5 Módulo de usuários

O módulo de usuários será responsável por manter todos os usuários da aplicação web, que realizam os cadastros em todos os demais módulos do sistema. Seguindo a mesma lógica dos funcionários, os usuários são vinculados às empresas, sendo este, juntamente com os campos e-mail e senha, os mais importantes do módulo, pois autenticam o usuário como um registro valido e concedem acesso a área administrativa do sistema. Tela de cadastro na Figura 21.

Figura 21: Cadastro de usuários



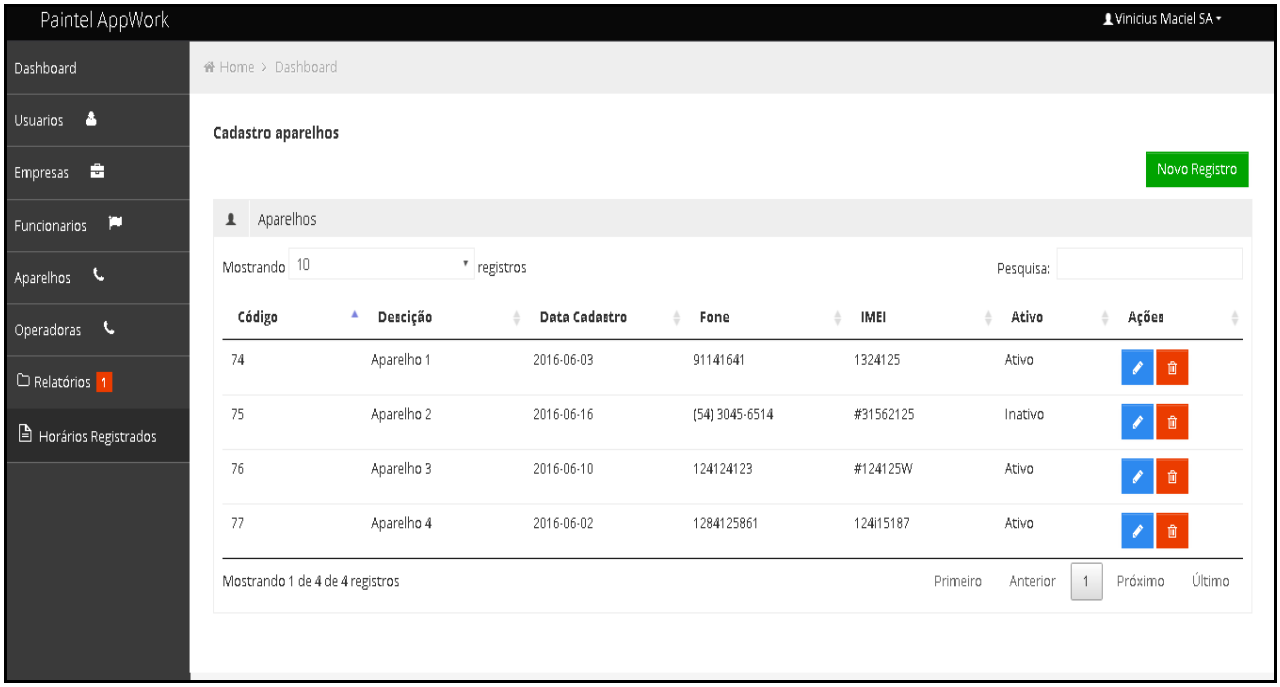
Fonte: do autor

4.2.6 Listagens

Em todos os módulos existe uma lista onde é possível fazer buscas, alterar registros por página, acionar paginação e fazer buscas, além de acessar os dados do registro e realizar eventuais exclusões quando necessário. Há também a possibilidade de alterar a ordenação, levando em consideração os cabeçalhos da tabela, tanto na ordem ascendente quando decrescente.

Este recurso é possível graças ao *plugin* de dataTable que possibilita estas e muitas outras funções de forma pratica e fácil auxiliando no desenvolvimento de interfaces administrativas. Tela de listagem na Figura 22.

Figura 22: Listagem de conteúdo



Fonte: do autor.

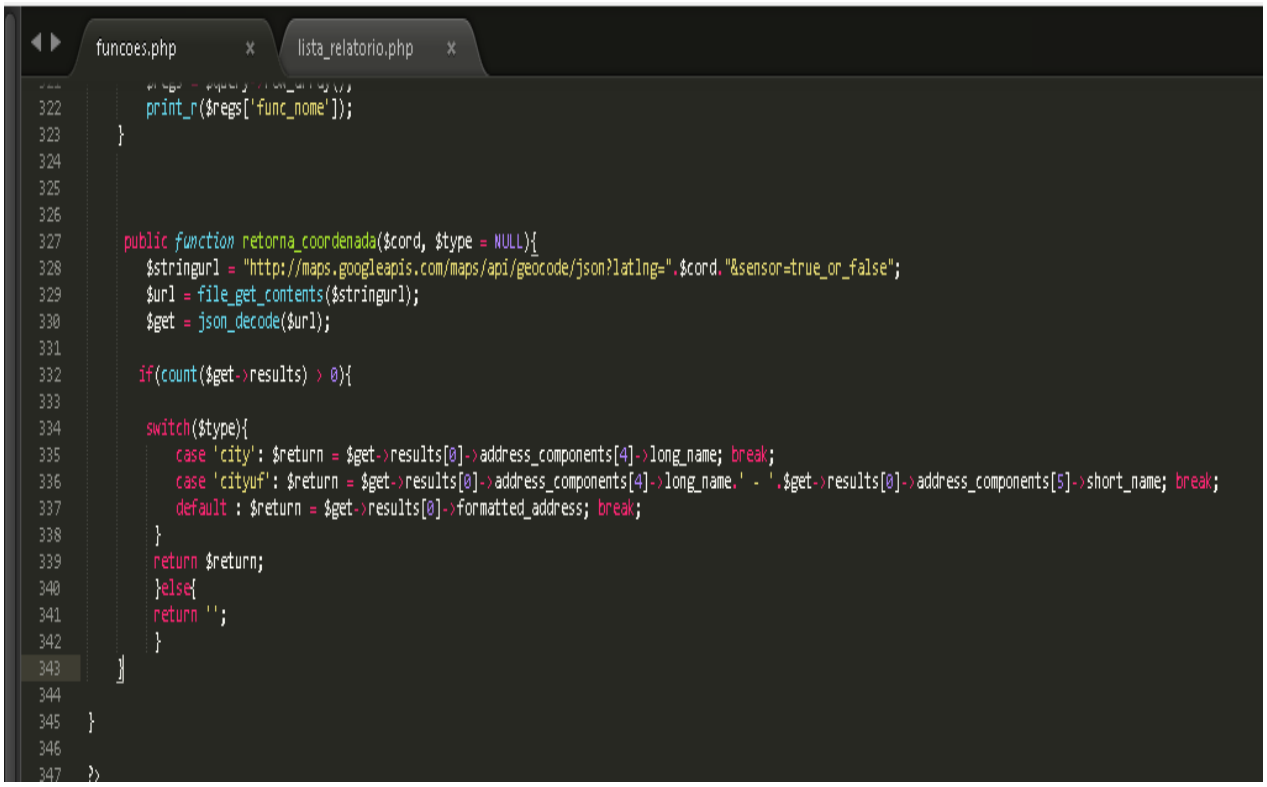
4.2.7 Módulo de relatórios

No módulo de relatórios está a visão principal de todos os funcionários, onde os gestores poderão fazer pesquisas através de filtros para auxiliar em seus levantamentos.

Para desenvolver esta área foi utilizado uma API de conversão de coordenadas do Google. Esta API passa por parâmetros as coordenadas, retornando o endereço aproximado, composto de rua, bairro, cidade e estado.

O código para este processo pode ser visualizado na Figura 23.

Figura 23: Função de conversão das coordenadas.



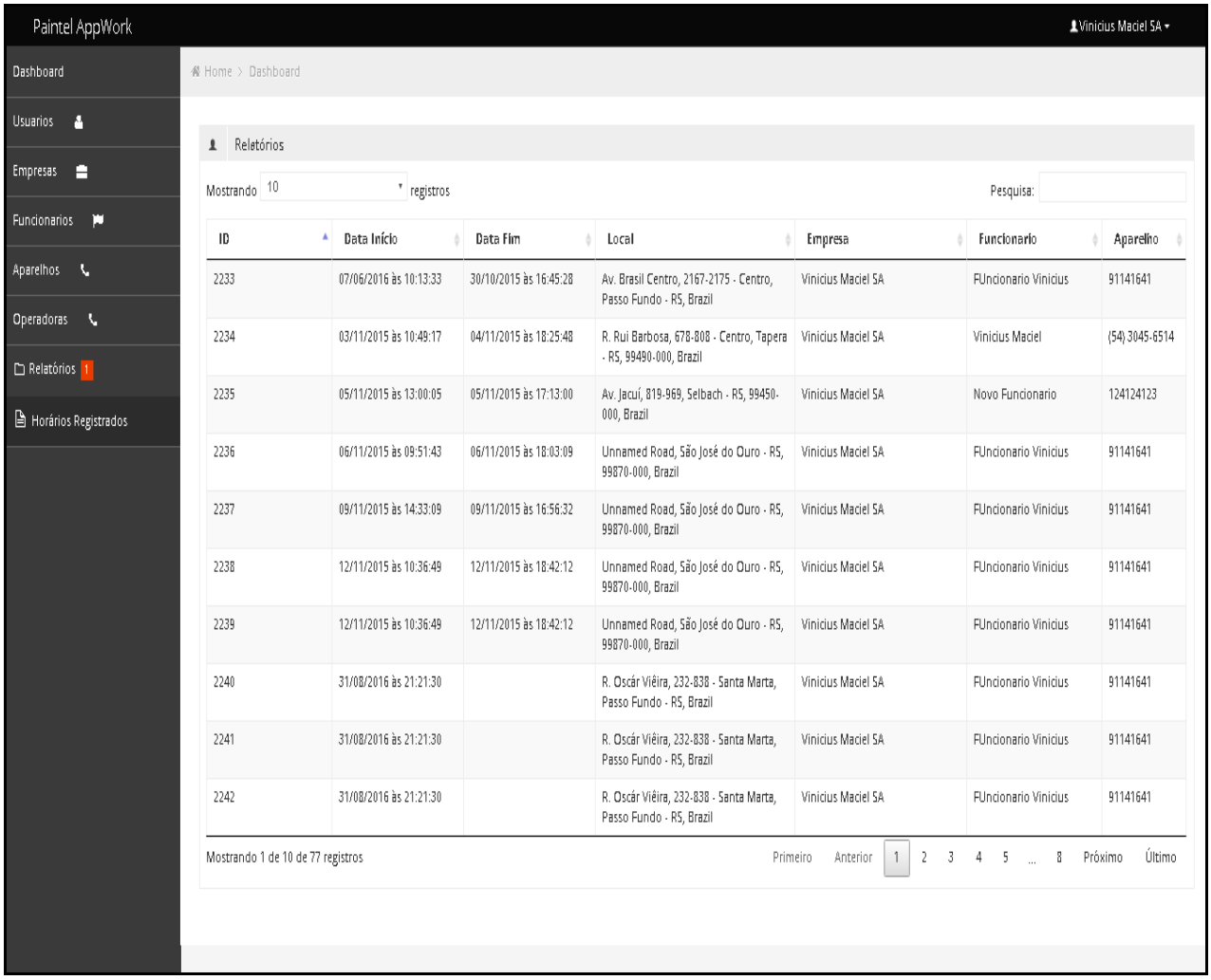
Fonte: do autor

Como resultado da aplicação os usuários da interface web podem acessar um relatório completo, com informações do funcionário e seus horários de trabalho.

Este relatório foi desenvolvido em uma *dataTable* que possibilita que os usuários realizem pesquisas, que buscam por qualquer termo dentro das colunas da tabela, ID, Data Início, Data Fim, Local, Empresa, Funcionário e Aparelho, além da funcionalidade de conversão de coordenadas.

Estes resultados são demonstrados através da Figura 24.

Figura 24: relatório de horário e localização.



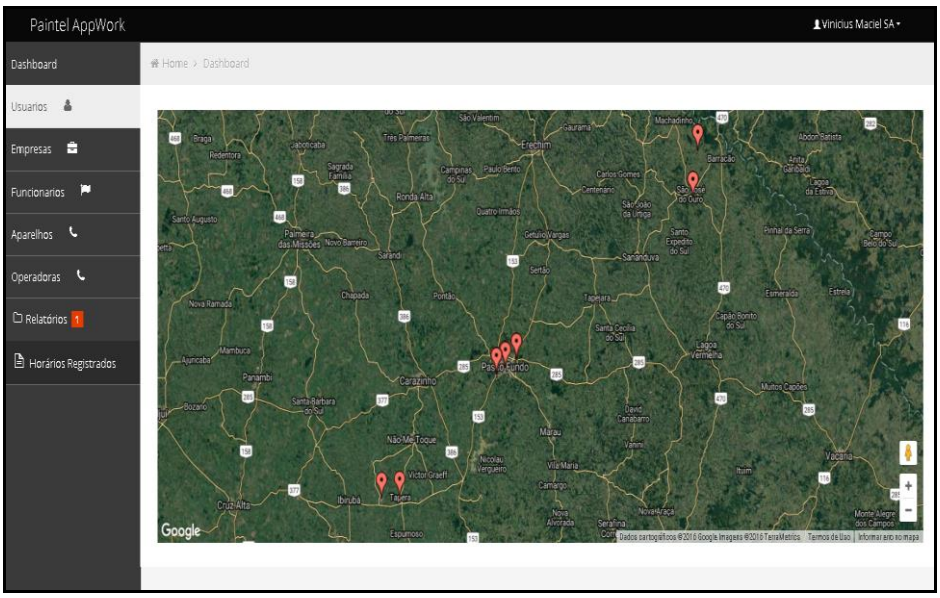
Fonte: do autor

4.2.8 Dashboard interativa

Foi desenvolvida uma *dashboard* interativa onde o usuário da aplicação web pode ver em que locais do mapa existem funcionários trabalhando, bem como a hora exata que iniciaram e finalizaram o experiente. Esta área verifica os pontos iniciados na data atual e exibe na tela.

Através de um *plugin* de mapas do Google, é possível converter as coordenadas buscadas no banco de dados principal e convertê-las em pontos de localização. Este procedimento é demonstrado através da Figura 25.

Figura 25: Dashboard inicial



Fonte: do autor

O procedimento de conversão é simples, depois de fazer a inclusão dos *scripts* do *plugin* é feito uma consulta a função da API passando por parâmetro os dados do banco de dados principal, através de um laço de repetição é percorrido todos os dados do banco e realizado a consulta montando os dados em um *array utilizado no* carregamento no mapa. Este procedimento é desenvolvido com o código da Figura 26.

Figura 26: Codificação da Dashboard Inicial.

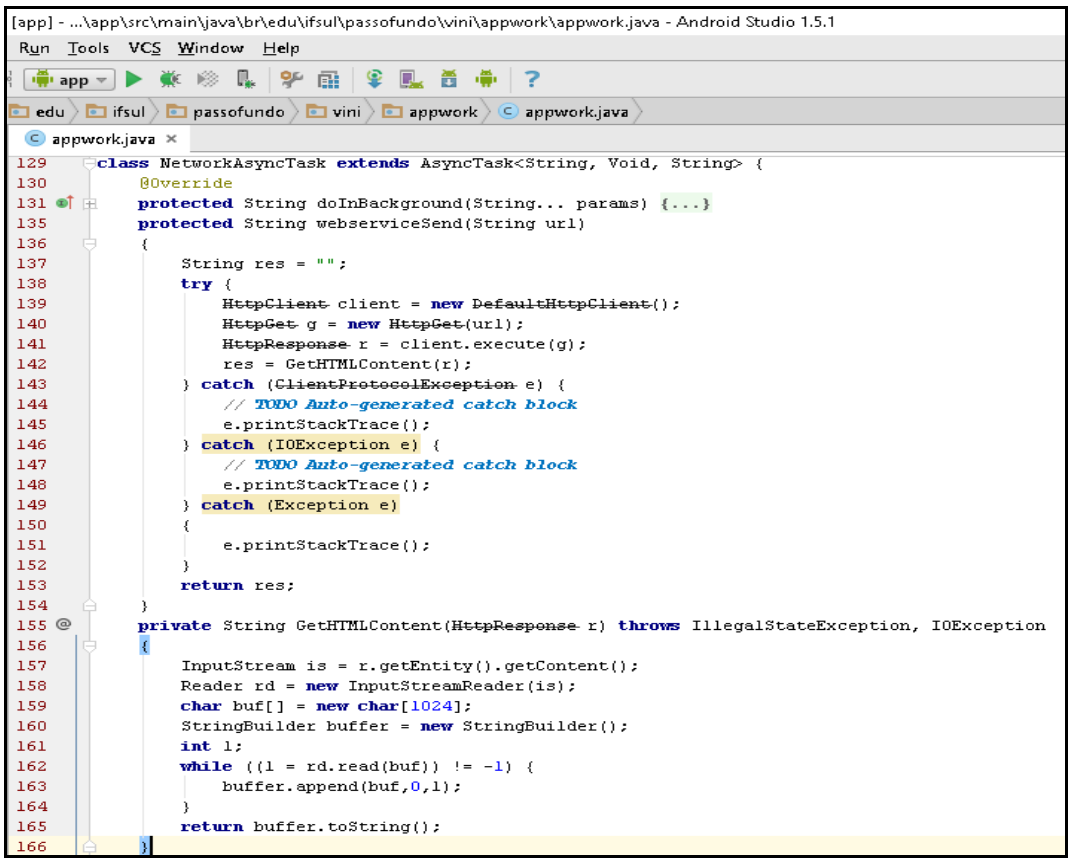
Fonte: do autor

4.3 APLICATIVO MÓVEL

O aplicativo móvel é a principal parte deste projeto, pois através desta interface são gerados os dados para a área administrativa na web. Todo o aplicativo foi desenvolvido na linguagem Java para Android e utilizando a IDE Android Studio, que auxilia na gerência dos pacotes necessários para a utilização das APIs da plataforma Android.

Como descrito no módulo de funcionários e aparelhos da interface web, é necessário uma autenticação no banco principal para que o usuário possa registrar seus horários e enviar para a área web. Para realizar a conexão do aplicativo com o WebService responsável pela consulta dos funcionários é necessária a utilização de uma classe especifica do Java, a HttpClient. Sua utilização no projeto se da através de dois métodos que se encontram na classe NetworkAsyncTask, sendo o método principal o webserviceSend que recebe por parâmetro a uma String para a consulta dos funcionários. Na Figura 27 estão apresentados os métodos de requisição HTTP.

Figura 27: Métodos de conexão ao Webservice.



Fonte: do autor

A consulta inicial é composta da URL do servidor WebService, telefone do funcionário e da senha de acesso, que retorna o código do funcionário o mesmo tenha informado os dados corretos, ou um retorno negativo, que dispara uma mensagem de dados inválidos ao usuário. O resultado deste processo é retratado na Figura 28 que demonstra a tela de login do aplicativo móvel.

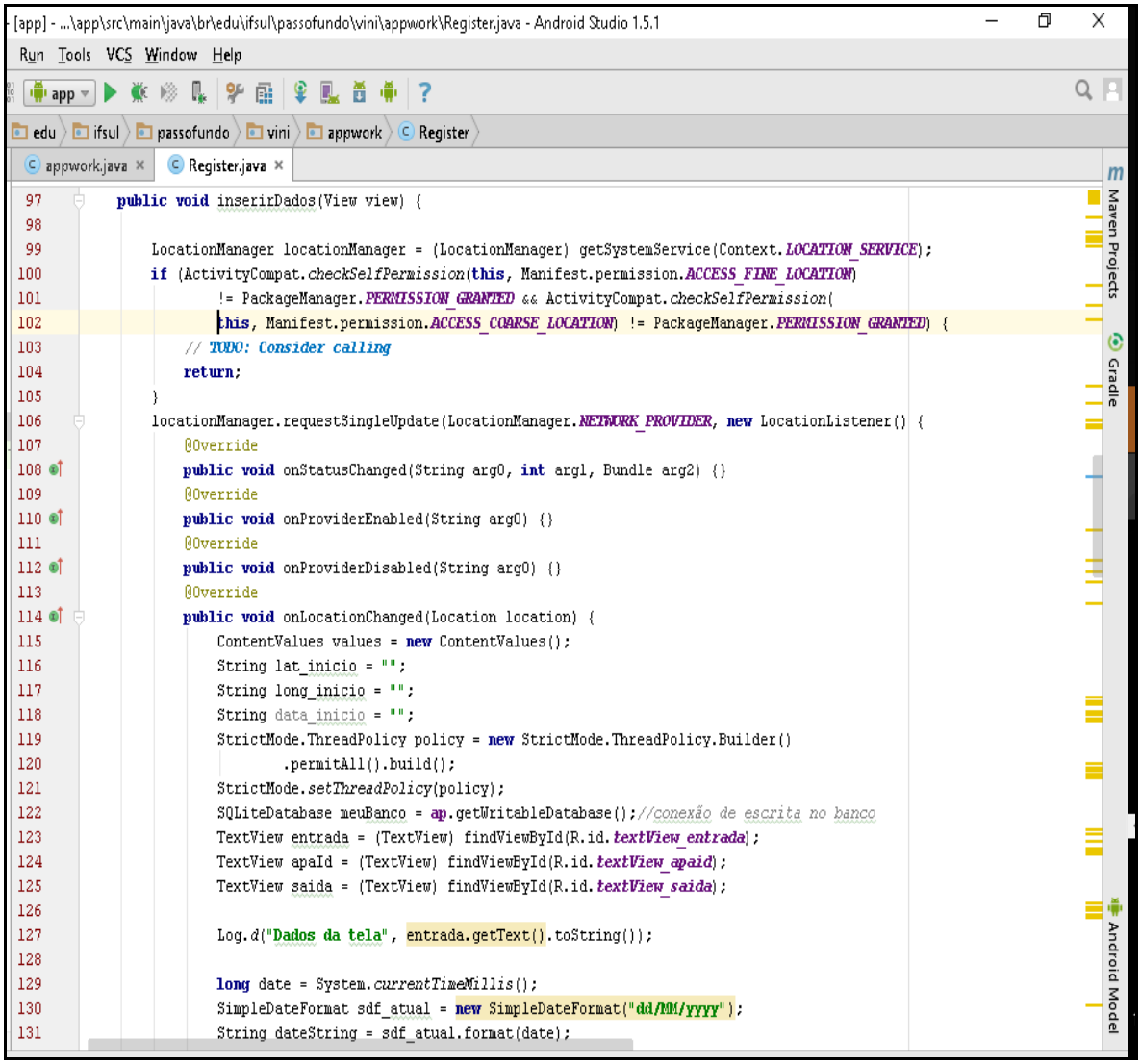
Figura 28: Login do aplicativo.

Fonte: do autor

Após o registro o usuário é direcionado para a tela inicial do aplicativo onde pode iniciar seus horários e os comunicar com o banco de dados principal. Para este procedimento foi necessária a utilização da API de geolocalização. Esta API trata-se da LocationManager, uma classe do Java para a plataforma Android. Ao implementar um objeto desta classe a própria IDE informa os métodos que são obrigatórios para a utilização da mesma.

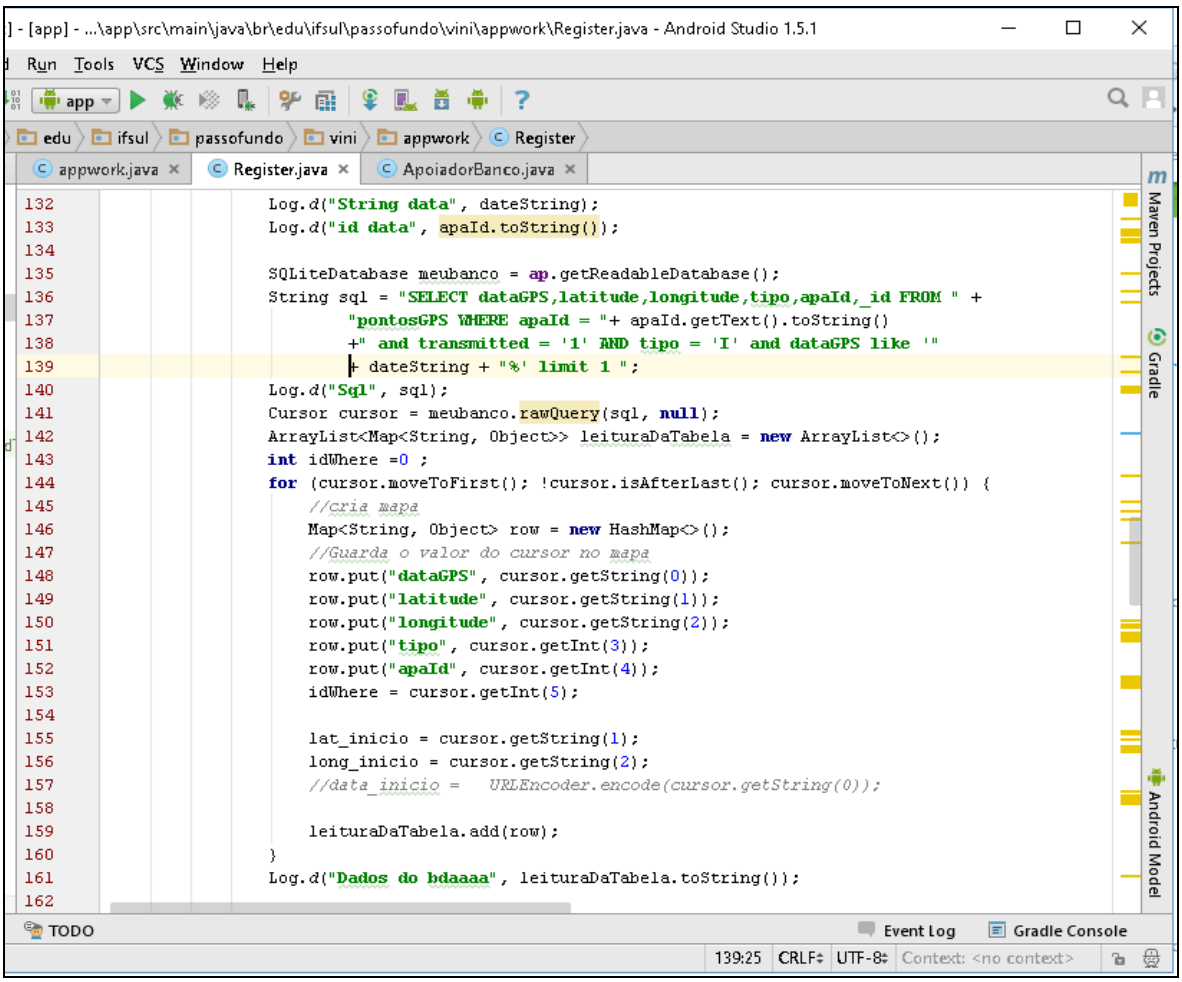
A implementação desta API está no método Inserir dados, ao criar o objeto da LocatioManager se faz necessário que se informem alguns parâmetros na aplicação, os quais serão utilizados para a recuperação das coordenadas geográficas. As opções para este parâmetro são: *GPS_PRIVIDER, NETWORK_PROVIDER* entre outros. Na presente aplicação foi utilizada a opção *Network_provider* que se mostrou mais eficiente e rápida para a aquisição dos dados. Nas Figuras 29 e 30 fica demonstrada a forma de iniciar as consultas de geolocalização.

Figura 29: Método de gravação dos horários.



Fonte: do autor.

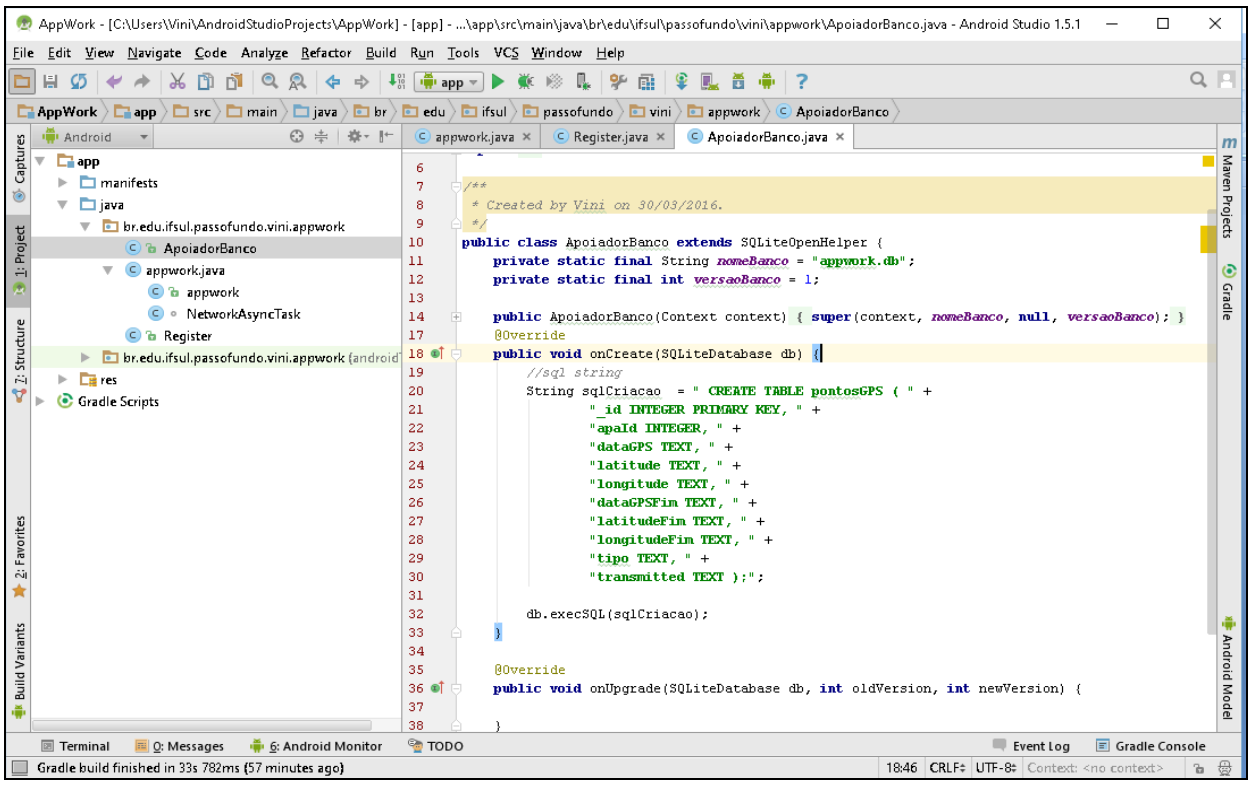
Figura 30: Método de gravação de horários.



Fonte: do autor.

Ao iniciar um registro o aplicativo faz uma nova consulta *WebService* e salva as coordenadas, data, hora e o código do aparelho logado, da mesma forma com que foi feita a consulta do login, exemplificada na sessão anterior. Além da consulta WebService o aplicativo também contém um banco de dados local onde são armazenados os registros para validação. Ao inserir um registro o aplicativo armazena no banco local os dados da hora de início e marca o status como iniciado. Assim, se o funcionário encerrar o aplicativo poderá abri-lo em outro momento e o sistema verifica se existem pontos em aberto no dia atual, informando que devem ser encerrados. Para tal se fez necessária a criação de uma classe que faz a administração do banco de dados local que é feito através do SQLite banco de dados para aplicações Android. Esta classe é apresentada na Figura 31.

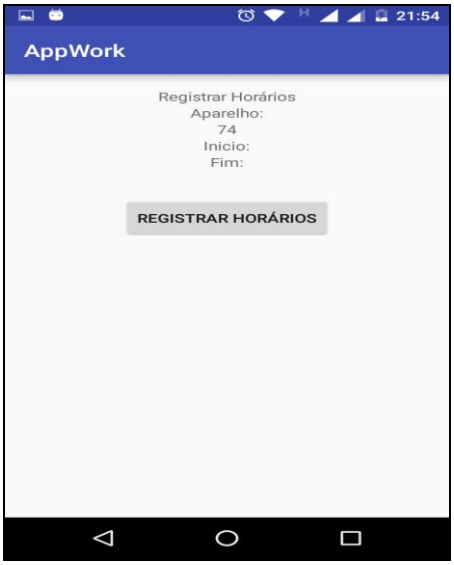
Figura 31: Configuração do banco de dados local.



Fonte: do autor.

Na Figura 32 é apresentada a tela de inicio de horários, que responde aos controles criados e é carregada após a verificação dos registros do banco de dados local.

Figura 32: Tela de Inicio do horário.



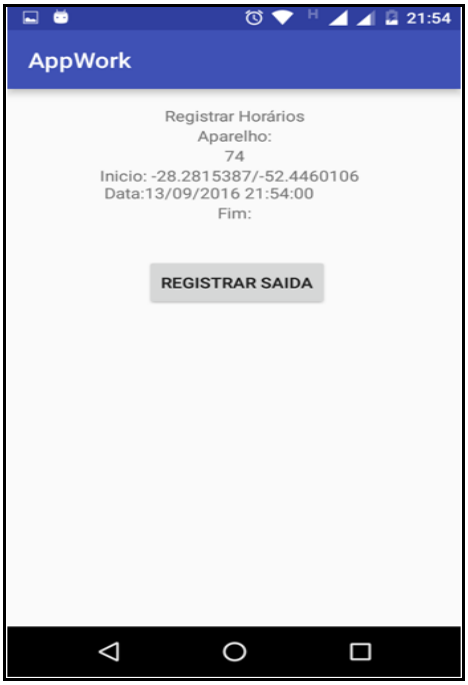
Fonte: do autor.

Após o registro inicial o aplicativo exibe a hora da gravação e as coordenadas, possibilitando o encerramento do registro atual e enviando a atualização dos horários para o banco de dados principal.

Conforme citado acima, o aplicativo controla os registros locais e informa ao usuário se há pontos a serem encerrados. Através de uma consulta no banco de dados local o aplicativo redireciona o usuário para uma tela em branco, onde o mesmo pode iniciar o horário, ou para uma tela com o registro em aberto. De qualquer uma destas telas o funcionário pode efetuar os registros e enviar ao banco de dados principal os dados de seu expediente atual.

Como este procedimento utiliza o SQL para gravar os dados localmente, o aplicativo pode ser fechado ou até mesmo o aparelho desligado que, na próxima execução do mesmo, o sistema irá consultar o banco de dados e se encontrado um registro em aberto será exibida uma tela aos moldes da Figura 33.

Figura 33: Tela de encerramento de horário.



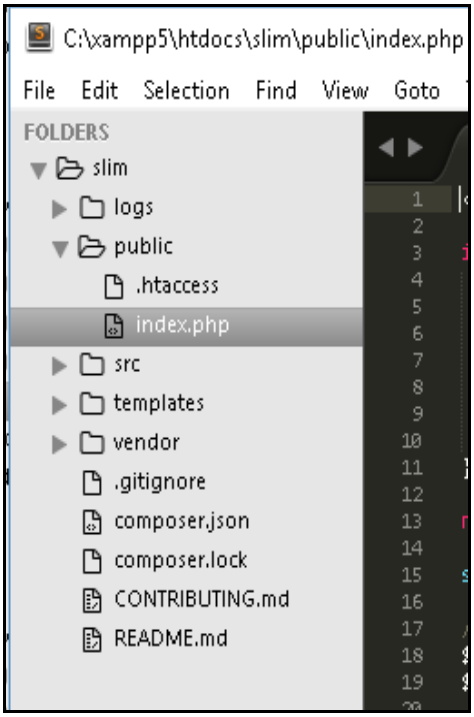
Fonte: do autor.

4.4 COMUNICAÇÃO WEBSERVICE

Conforme citado durante toda a seção do aplicativo móvel, o mesmo fica constante comunicação com o banco de dados principal, uma vez que este funciona como base para a interface web. Esta comunicação se dá através de *Webservices* desenvolvidos a partir da estrutura REST, mais especificamente o SlimFramework para desenvolvimento de *WebServices* em PHP, o qual foi citado no referencial teórico do presente trabalho, na seção 2.3.1. Esta estrutura é conhecida pela praticidade e facilidade de implementação, tendo em vista que utiliza os métodos do protocolo HTTP que são facilmente manipuláveis.

Primeiramente se faz necessário explicar a estrutura utilizada no servidor de *WebService* para que fosse possível desenvolver este projeto. O SlimFramework disponibiliza uma estrutura previamente configurada, que contem as bibliotecas necessárias para a implementação de um servidor REST, a qual é demonstrada através da Figura 34.

Figura 34: Estrutura SlimFramework.

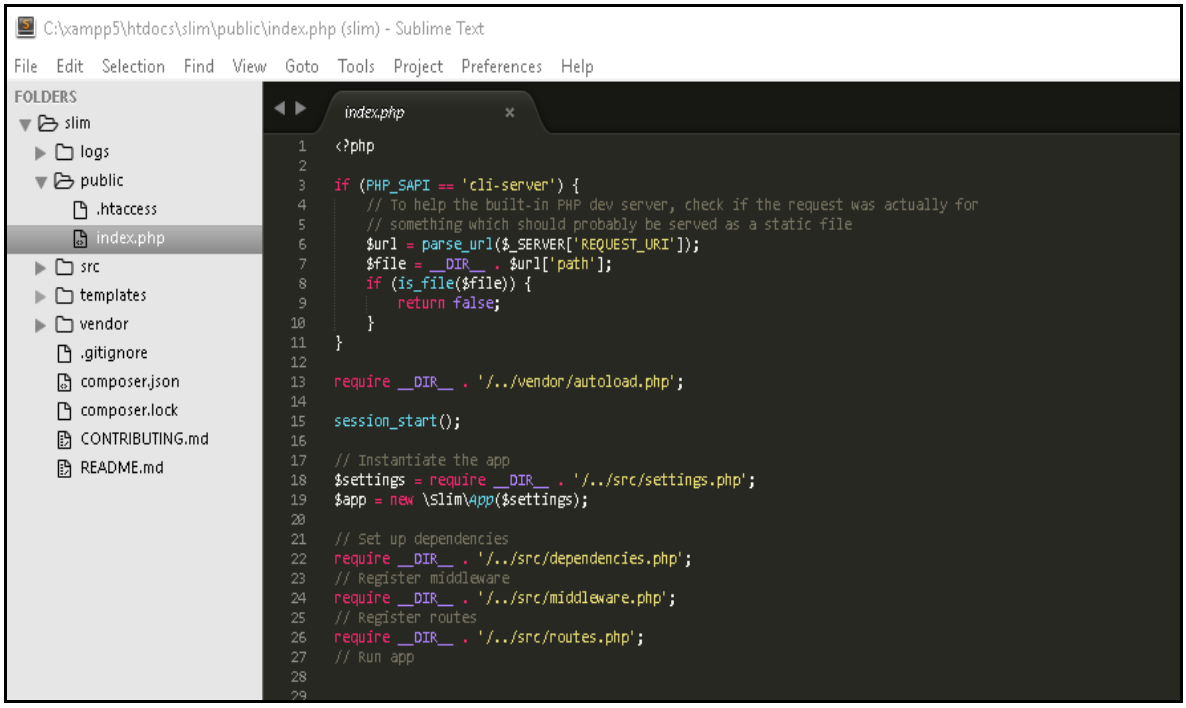


Fonte: do autor.

Após a instalação do framework foi criado o servidor. Para isso foi necessário que se instanciasse algumas variáveis, tais como $url que é do caminho base do servidor, a variável de configurações que é um objeto da classe *Settings* do

framework, e também a $app que instancia um objeto da classe Slim\App, sendo esta a mais importante, visto que será responsável por criar os métodos do *WebService*. Estas configurações são demonstradas na Figura 35.

Figura 35: Configuração inicial do servidor REST.



Fonte: do autor.

O passo seguinte necessário para o desenvolvimento dos serviços foi fornecer uma conexão entre o servidor e o banco de dados principal. Para realizar a conexão com o MySql foi criado um método que faz a consulta com os parâmetros do banco de dados e retorna um objeto do tipo PDO, através do qual é possível realizar operações de DDL no banco principal. Exemplo da configuração do banco na Figura 36.

Figura 36: Conexão ao banco principal.

Fonte: do autor.

Após todas as configurações foram desenvolvidos os métodos específicos responsáveis por comunicar as ações do aplicativo com o banco principal. Por padrão foi adotado o método GET em todas as consultas, o que facilitou na comunicação entre as interfaces.

O primeiro método foi o de consulta aos funcionários, que trata-se de um método relativamente simples que recebe dois parâmetros, sendo estes número do aparelho e senha de acesso. Estes parâmetros são verificados em uma consulta ao banco de dados e validados, retornando para o aplicativo o número do aparelho ou uma variável *false*, assim, realizando as validações necessárias. Este procedimento é demonstrado pela Figura 37.

Figura 37: Consulta de funcionários.

```
$app->get('/funcionario/{fone}/{senha}', function ($request, $response, $args) {

    $sql = "SELECT
                *
            FROM
                aparelhos a
            INNER JOIN
                funcionarios_aparelhos ON a.apa_id = funcionarios_aparelhos.aparelhos_apa_id
            INNER JOIN
                funcionarios on funcionarios.func_id = funcionarios_aparelhos.funcionarios_func_id
            WHERE
               a.apa_fone = ". $args['fone']."
            AND a.apa_senha = md5('". $args['senha']."')
            AND a.apa_tipoativo = 1";


    $result = getConn()->query($sql);
    $funcionario = $result->fetchAll(PDO::FETCH_OBJ);

    if($funcionario != '' && sizeof($funcionario) >  0){
    echo  json_encode($funcionario, JSON_PRETTY_PRINT) ;
    }else {
        echo json_encode("[{'apa_id'= '0'}]");
    }


    //return $response->write("Funcionario " . $funcionario);
});
```

Fonte: do autor.

Outros dois métodos foram implementados no servidor: o “gps_inicio” e “gps_fim”, que são responsáveis, respectivamente, pela inserção dos registros iniciais de horário e localização dos funcionários e pelo fechamento dos registros iniciados. Para o desenvolvimento destes métodos foram passados por parâmetro as coordenadas de localização, horários de início e encerramento dos pontos, o código do aparelho, o qual havia sido enviado ao aplicativo anteriormente, e o status que identifica o início ou encerramento de um horário. Ambos os métodos foram implementados com o padrão GET e podem ser visualizados através das Figuras 38 e 39.

Figura 38: Método GPS_INICIO.

Fonte: do autor.

Figura 39: Método GPS_FIM.

```
$app->get('/gps_fim/{ws_fone}/{coordenada_ini}/{datahoracoleta_ini}/{
latitude_fim}/{longitude_fim}/{coordenada_fim}/{datahoracoleta_fim}/{
datahoraenvio_fim}/{datahoraclique_fim}/{status}', function ($request
    , $response, $args) {

    $apa_id = $this->aparelho_id($ws_fone);
    $latitude_fim = $args['latitude_fim'];
    $longitude_fim = $args['longitude_fim'];
    $coordenada_fim = $args['coordenada_fim'];
    $datahoracoleta_fim = $args['datahoracoleta_fim'];
    $datahoraenvio_fim = $args['datahoraenvio_fim'];
    $datahoraclique_fim = $args['datahoraclique_fim'];
    $status = $args['status'];
    $apa_id = $args['apa_id'];
    $coordenada_ini = $args['coordenada_ini'];
    $datahoracoleta_ini = $args['datahoracoleta_ini'];


    $sql =  " UPDATE gps_posicao SET gps_pos_latitude_fim = '$
        latitude_fim',
                gps_pos_longitude_fim= '$longitude_fim',
                gps_pos_coordenada_fim= '$coordenada_fim',
                gps_pos_datahoracoleta_fim= '$datahoracoleta_fim',
                gps_pos_datahoraenvio_fim= '$datahoraenvio_fim',
                gps_pos_datahoraclique_fim= '$datahoraclique_fim',
                gps_pos_status = '$status'
            WHERE apa_id = $apa_id and
            gps_pos_coordenada_ini = '$coordenada_ini'  and
            gps_pos_datahoracoleta_ini = '$datahoracoleta_ini'";
//     echo $sql;
    $result = getConn()->query($sql);
    $funcionario = $result->fetchAll(PDO::FETCH_OBJ);

    echo "{funcionario:" . json_encode($funcionario,
        JSON_PRETTY_PRINT) . "}";

});
```

Fonte: do autor.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo realizar o desenvolvimento de um projeto composto por um aplicativo móvel e uma interface web que fossem capazes de, em conjunto, disponibilizar uma ferramenta de geração de horários ponto e localização, bem gerência destes horários. Visto que atualmente a maioria das empresas tem alguma forma de registro de horário ponto, este projeto se propôs a auxiliar aquelas que possuem funcionários que trabalham em ambiente externo, onde o controle dos horários das atividades se torna mais complexo e de difícil comprovação.

Para que este projeto atingisse seus objetivos, grande parte do esforço foi empregado na modelagem do sistema, incluindo o levantamento de requisitos e a construção de diagramas. Desse modo pode-se ter uma visão dos principais processos que deveriam ser executados e no que estes processos acarretariam.

Após a etapa de modelagem e análise do projeto foram estudadas as tecnologias e linguagens utilizadas. Este projeto poderia ser desenvolvido de diversas outras maneiras e com tecnologias diferentes, porém por uma questão de atualidade, praticidade e conhecimento, foi escolhida as tecnologia Java para

Android. A linguagem PHP e o framework *Codeigniter* foram utilizados na área web. Como ponte de comunicação entre estes, o *RestFull,* com a arquitetura do *SlimFramework*.

Uma das principais dificuldades do projeto foi a utilização do Java para Android no desenvolvimento do aplicativo, principalmente nos momentos em que foi necessário acessar as APIs, como a de geolocalização, e também na comunicação com o servidor externo para gravação e consulta de dados. Devido a estas dificuldades, a funcionalidade de verificação da rede de dados e o processo de retransmissão automática de dados não pôde ser desenvolvido, pois, por sua complexidade e o curto espaço de tempo para desenvolvimento acarretariam na inviabilização do projeto como um todo. Estas melhorias serão desenvolvidas para trabalhos futuros.

Visando estas dificuldades de desenvolvimento nativo para o Android, uma das soluções levantadas para uma versão futura, seria o desenvolvimento da área do aplicativo em um *framework* chamado Cordova. Esta arquitetura permite desenvolver aplicações hibridas, que rodam HTML, CSS e Javascript, porém se comportam como um aplicativo instalado no smartphone. Desta maneira por se tratar de uma linguagem mais maleável, o Javascript, o desenvolvimento seria mais simples. Além disso, este framework possibilita a compilação do mesmo código fonte para mais de uma plataforma, evitando a necessidade de reescrever o sistema caso houvesse a necessidade de desenvolver para o IOS, por exemplo.

Durante o decorrer do projeto outras dificuldades surgiram, mas foram contornadas, como por exemplo, a necessidade da compra e criação de um servidor web onde foi hospedado o banco de dados e os arquivos de WebService para que o aplicativo pudesse ser testado em situações reais. Outra dificuldade foi a de como fornecer as coordenadas geográficas na interface web de uma forma que os usuários administrativos pudessem interpretá-las, este obstáculo foi vencido com o auxilio de APIs do Google, que fazem a conversão destes dados para os endereços aproximados.

Apesar dos problemas encontrados durante o desenvolvimento do projeto foi possível implementar praticamente todas as funcionalidades modeladas na etapa de análise. Tanto a interface administrativa, quanto o aplicativo executaram corretamente as funcionalidades desenvolvidas e desta forma pode-se afirmar que

através de um aplicativo para dispositivos móveis e de sua comunicação com uma interface na web é possível solucionar o problema das empresas que precisam manter o controle de seus funcionários externos coletando seus horários e suas coordenadas geográficas.

REFERÊNCIAS

Bruno Luiz. Artigo Java Magazine 56 - Web services REST uma abordagem prática Disponível em:< http://www.devmedia.com.br/artigo-java-magazine-56-web-services-rest-uma-abordagem-pratica/8447 > Acesso em 10 de abril de 2016.

British Columbia Institute of Technology. Codeigniter Documentation Disponivel em: < https://www.codeigniter.com/docs> Acesso em 10 de abril de 2016.

DEITEL, Paul; DEITEL, Harvey. Java: como programar. 8. ed. São Paulo: Pearson, 2010.

EllisLab, Inc. A Brief History of CodeIgniter. Disponivel em: <https://ellislab.com/codeigniter >. Acesso em 10 de abril de 2016.

GONCALVES, Antonio. Beginning Java EE 7. Apress, 2013.

GUEDES, Gilleanes T. A. UML 2: uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.

LECHETA, Ricardo R. Google Android , Aprenda a criar aplicações para dispositivos moveis com SDK, Novatec Editora, 2013.

NIEDERAUER, Juliano. PHP pra quem sabe PHP , 2008.

PHP Hypertext Preprocessor. Disponivel em: < http://php.net/>. Acesso em 10 de abril de 2016.

SILVA, Mauricio Samy. Bootstrap 3.3.5 Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos, 2015.

Slim a microframework for PHP.Disponivel em < http://slimframework.com >. Acesso em 21 de maio de 2016

TILKOV,Stefan. Uma rápida Definição de Restfull, 2008 Disponivel em: <https://www.infoq.com/br/articles/rest-introduction> Acesso em 10 de abril de 2016.