UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

RELATÓRIO DE ATIVIDADES DO ESTÁGIO

Lucas Pereira da Silva

Florianópolis, novembro de 2013

ATIVIDADES DO ESTÁGIO DO ALUNO LUCAS PEREIRA DA SILVA

RESUMO

O projeto SincroBUS é um sistema avançado de informações e operações para controle do transporte público. O projeto foi implantado em escala piloto na cidade de Blumenau – SC e entre as atividades está o gerenciamento da linha de ônibus Troncal-10. O aluno Lucas Pereira da Silva participou do projeto com bolsista/estagiário. A atuação do aluno no projeto foi focada no subprojeto Sistema Web que tem como objetivo apresentar informações de linhas de ônibus. O Sistema Web é composto por dois módulos, um cliente e um servidor. O módulo servidor é responsável pela recepção e gerenciamento de informações, enquanto que o módulo cliente tem o objetivo de apresentar as informações gerenciadas pelo servidor em um ambiente Web que seja de fácil utilização. As informações apresentadas pelo sistema são compostas por informações em tempo real que refletem a atual condição de ônibus da linha Troncal-10. O sistema também é responsável por mostrar os horários das próximas saídas de ônibus e pontos de parada pertencentes a linha. O projeto também envolveu atividades de georreferenciamento, mais especificamente a geocodificação e geocodificação reversa de endereços e coordenadas geográficas, respectivamente. Essas atividades se fizeram necessárias para realizar o levantamento do itinerário da linha Troncal-10 e apresentar as informações levantadas em mapas no módulo cliente.

Palavras-chave: SincroBUS, linha de ônibus, Sistema Web, transporte público.

1. INTRODUÇÃO

Esse relatório trata das atividades desenvolvidas pelo aluno Lucas Pereira da Silva no projeto do SincroBUS, mais especificamente no subprojeto Sistema Web, sob responsabilidade do professor Luiz Fernando Bier Melgarejo, coordenador do Laboratório de Software Educacional (Edugraf). O objetivo do subprojeto Sistema Web é disponibilizar em mapas as informações sobre a linha de ônibus Troncal-10 em Blumenau, e informações em tempo real sobre os ônibus que pertencem a esta linha.

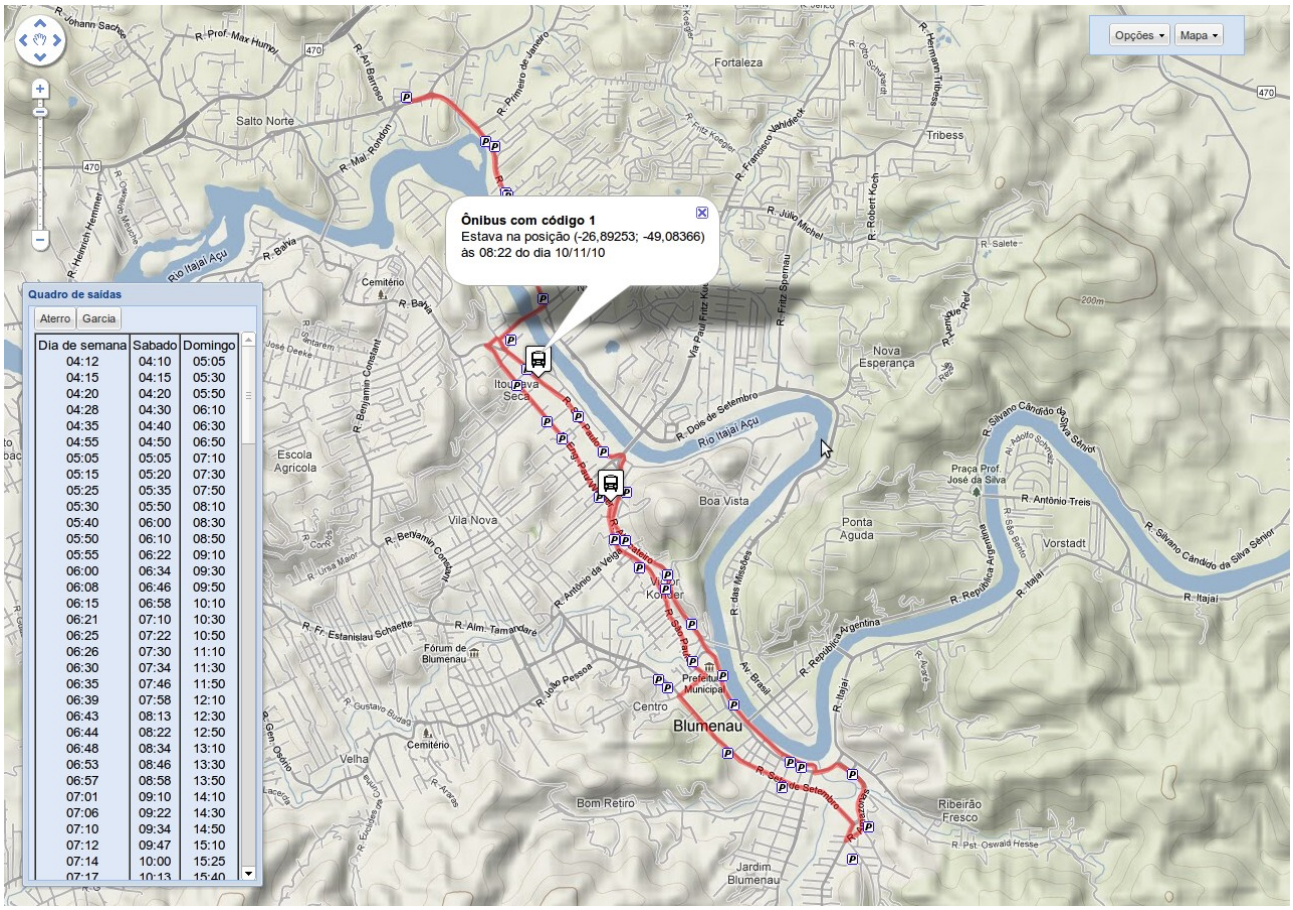

Figura 1: Trajeto da linha de ônibus Troncal-10 visualizado através do módulo cliente do SincroBUS

O subprojeto Sistema Web é composto de dois módulos: um módulo servidor e um módulo cliente. O módulo servidor consiste de um conjunto de serviços Web seguindo o estilo arquitetural REST, responsável tanto pela integração com outros subsistemas do projeto SincroBUS, quanto pela centralização de informações disponibilizadas pelos demais subsistemas do projeto. O módulo cliente é responsável pela apresentação de informações para usuários do transporte público. Os usuários do transporte público podem ter acesso ao módulo cliente através da utilização de um navegador Web.

Todas as atividades desenvolvidas pelo aluno no projeto foram atividades voltadas para o auxílio do desenvolvimento de funcionalidades do subprojeto Sistema Web. Em uma primeira etapa as atividades se focaram no auxílio ao desenvolvimento de recursos Web para o módulo servidor através da utilização do *framework* Restlet. A segunda etapa teve como objetivo realizar alterações no módulo cliente que é responsável por apresentar as informações obtidas em tempo real.

O sistema desenvolvido no escopo do subprojeto pode ser acessado através do endereço: `http://sincrobus.edugraf.ufsc.br`.

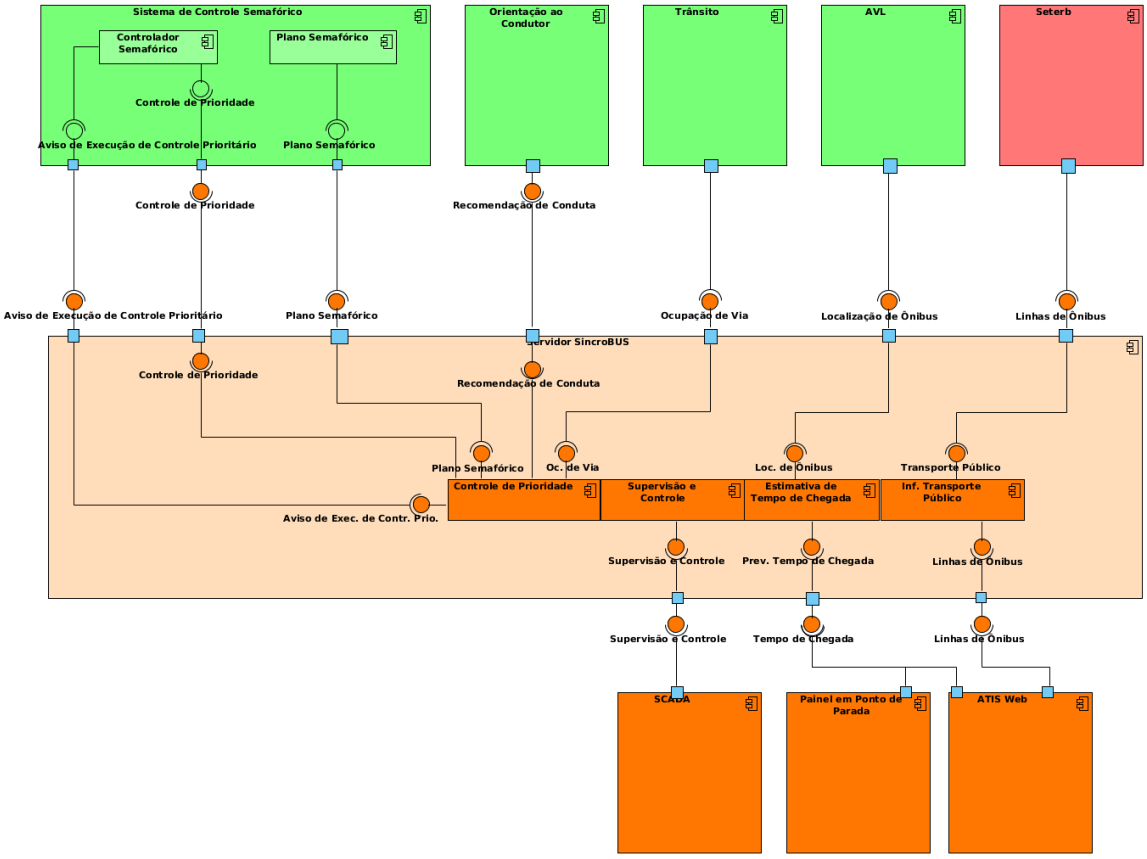

Figura 2: Diagrama da Arquitetura do SincroBUS

2. MATERIAL E MÉTODOS

As atividades desenvolvidas se focaram no estudo das tecnologias voltadas para Web a serem utilizadas no subprojeto. As atividades planejadas e executadas são listadas abaixo:

- Estudo do estilo arquitetural REST;
- Estudo do protocolo HTTP;
- Estudo e aplicação prática do arcabouço Restlet;
- Estudo e aplicação prática do GWT;
- Estudo e aplicação prática do JavaScript;
- Estudo e aplicação prática do JSON;
- Estudo e aplicação prática do Hibernate;
- Estudo sobre georreferenciamento e utilização da API do Google Maps.

Os estudos sobre REST e HTTP são importantes, pois fornecem conceitos basilares para a melhor compreensão das tecnologias utilizadas no projeto. Por isso, as atividades iniciais se focaram nos estudos dessas duas tecnologias

Todas as atividades seguiram o mesmo modelo: estudo de uma tecnologia específica e aplicação prática dessa tecnologia para que se possa ter uma melhor compreensão da mesma.

3. RESULTADOS E DISCUSSÃO

Os resultados alcançados a partir das atividades desenvolvidas foram influenciados fortemente pela pesquisa e utilização de alguma tecnologia. As tecnologias pesquisadas e utilizadas estiveram efetivamente envolvidas no projeto e por isso, a análise dos resultados e discussões foi feita de forma segmentada de acordo com as tecnologias estudadas.

3.1. Estudo do Estilo Arquitetural REST

REST é um estilo arquitetural aplicado a sistemas de hipermídia distribuídos que foi proposto por Roy Fielding (2000) em sua tese de doutorado. Como exemplo de um sistema de hipermídia distribuído que possui sua arquitetura baseada em REST pode-se citar a Web.

Uma arquitetura baseada em REST é a Arquitetura Orientada a Recursos, proposta por Richardson e Ruby (2007). A Arquitetura Orientada a Recursos utiliza tecnologias específicas da Web como HTTP, representações JSON, HTML, XML e outros.

Um conceito fundamental de REST é o conceito de recursos. Um recurso pode ser qualquer informação que possa ser identificada unicamente. Na Arquitetura Orientada a Recursos essa identificação única é chamada de URI. Um exemplo de URI seria `/projetos/sincrobus`. Cada recurso pode ter diversas representações, onde cada representação pode possuir metadados para descreve-la. Por exemplo, pode existir um recurso que é um artigo e esse artigo possuir duas representações, uma representação em texto puro e outra em formato HTML.

REST é baseado no paradigma cliente servidor e nele o servidor é responsável por hospedar os recursos enquanto que o cliente realiza solicitações de operações sobre esses recursos. Uma aplicação REST é composta por recursos que por sua vez é composto por uma ou mais representações. Para que uma aplicação tenha utilidade prática é necessário que se possa realizar operações sobre os recursos. Em REST essas operações são baseadas em uma interface uniforme. A Arquitetura Orientada a Recursos herda essa interface uniforme e a implementa através dos métodos do protocolo HTTP:

- GET: obtém a representação de um recurso;

- PUT: cria um recurso ou modifica sua representação;
- POST: cria um recurso;
- DELETE: exclui um recurso;
- HEAD: obtém os metadados de um recurso;
- OPTIONS: obtém os métodos suportados por um recurso.

3.2. Estudo do Protocolo HTTP

O HTTP é um protocolo do nível de aplicação descrito pela RFC 2616 (HTTP 1.1) e que forma a base da Web. Ele é um protocolo baseado no paradigma de requisição e resposta e no paradigma de cliente e servidor. Nele o cliente fará requisições para o servidor e o servidor deverá responder essas requisições. O HTTP possui ao todo oito métodos, dos quais seis deles são utilizados pela Arquitetura Orientada a Recursos e já foram descritos anteriormente. Os outros dois métodos são o TRACE e o CONNECT.

Tanto uma requisição quanto uma resposta HTTP são compostas por um cabeçalho e pelo corpo da mensagem. Uma requisição básica HTTP contém o método da requisição, a URI do recurso, a versão do HTTP usada, atributos do cabeçalho e o corpo da mensagem (caso exista). Assim, para obter a página Web `index.html` no endereço `sincrobus.edugraf.ufsc.br` o navegador deverá enviar a seguinte requisição HTTP:

```
GET /index.html HTTP/1.1
Host:  sincrobus.edugraf.ufsc.br
Accept: text/html
```

Essa é uma requisição simples, mas poderia ser composta por outros atributos no cabeçalho como o tamanho do conteúdo enviado ao servidor (atributo `Content-Length`), o tipo de conteúdo enviado ao servidor (atributo `Content-Type`), o próprio conteúdo (caso exista), o agente de usuário que está fazendo a requisição (atributo `User-Agent`), entre outros.

Uma resposta para a requisição feita acima será composta pela versão do protocolo, por um código de resposta, por atributos de cabeçalho e pelo corpo da resposta (caso exista):

```
HTTP/1.1 200 OK
```

```
Content-Type: text/html
Content-Length: 512
...
CORPO DA RESPOSTA
```

Assim como no caso da requisição a resposta também poderia ser composta por outros atributos. Um atributo importante em uma resposta HTTP é o código de resposta, que possui três algarismos sendo que o primeiro indica a classe da resposta e os dois últimos representam o significado exato da resposta. Por exemplo, uma resposta com um código 201 significa que a requisição foi realizada com sucesso (primeiro algarismo é o 2) e que um recurso foi criado no servidor (algarismos 01).

- 1xx: classe de respostas que contêm informações sobre o processamento da requisição;
- 2xx: classe de respostas onde a requisição foi bem sucedida;
- 3xx: classe de respostas de redirecionamento;
- 4xx: classe de respostas para requisições mal feitas;
- 5xx: classe de respostas para erros ocorridos no servidor.

3.3. Estudo e Aplicação do Arcabouço Restlet

O arcabouço Restlet é uma ferramenta Java utilizada para o desenvolvimento de aplicações Web utilizando os conceitos do estilo REST. Ele pode ser usado tanto no desenvolvimento do cliente quanto no desenvolvimento do servidor. O estudo do Restlet teve como pré-condição o estudo de REST e do Protocolo HTTP. Esses estudos prévios permitiram uma melhor compreensão dos conceitos envolvendo o Restlet.

O Restlet foi utilizado no desenvolvimento de recursos para o módulo servidor e busca facilitar a criação de uma aplicação em Java que siga as restrições e conceitos do estilo REST. Com ele, o mapeamento de uma especificação REST para a linguagem Java se torna uma tarefa simples.

O ponto de partida do Restlet é a classe `Application` que deve ser estendida por uma classe da aplicação que está sendo desenvolvida. A classe da aplicação conterá entre outras coisas, o mapeamento das URIs dos recursos para classes Java que

representarão esses recursos. Esse mapeamento é realizado através da classe `Router`. Por exemplo, para mapear a URI `/projetos/{nomeDoProjeto}` para a classe Projeto, utilizamos o método `attach` de `Router`. Para o Restlet, `{nomeDoProjeto}` representa um valor variável que poderá ser diferente para cada recurso que tiver a URI iniciada com `/projetos/`. Dessa forma, é possível criar o mapeamento de URIs (genéricas ou não) para classes Java que representarão os recursos da aplicação REST.

Após a realização dos mapeamentos de URIs para classes que representam os recursos, é necessário especificar as operações sobre os recursos. As classes que representam recursos devem estender a classe `Resource` que faz parte do Restlet. Após isso, utiliza-se anotações Java para definir a operação sobre o recurso, o tipo de mídia da representação e tipo de mídia do conteúdo consumido pelo recurso. Por exemplo, no caso de um método que é responsável por fornecer a representação HTML de um recurso, então teríamos a seguinte anotação Java: `@Get("text/html")` e para o caso de um método onde é cadastrado um dado no formato JSON e é retornado uma página HTML, teríamos a anotação: `@Post("application/json:text/html")`.

Nos exemplos descritos anteriormente, `@Post` e `@Get` representam os métodos HTTP pelo qual o recurso está disponível e o valor entre os parênteses é o tipo de mídia disponível para o recurso. O tipo de mídia descreve o formato de uma representação especifica do recurso. Para o último exemplo descrito acima, será recebido uma representação no formato JSON enviada através do método HTTP POST e será retornada uma resposta de uma representação no formato HTML.

3.4. Estudo e Aplicação do GWT

O GWT (Google Web Toolkit) é uma ferramenta para desenvolvimento Web. Dentro do contexto do projeto SincroBUS o GWT foi utilizado no módulo cliente para gerar a interface gráfica Web. A utilização do GWT no projeto trouxe várias vantagens, pois ele é um *framework* para o desenvolvimento de aplicações que rodam em navegadores através da geração de código JavaScript. Ou seja, um conjunto de ferramentas e bibliotecas que permitem escrever um programa em Java, usando um subconjunto dessa linguagem, e transformá-lo em código JavaScript otimizado e compatível com grande

parte dos navegadores.

Para a criação de interfaces gráficas Web pode-se utilizar a linguagem de marcação de páginas (HTML) ou utilizar a linguagem JavaScript que permite a manipulação do DOM (*document object model*) da página Web. Através da manipulação do DOM é possível criar os elementos que formarão a interface gráfica.

O GWT utiliza o segundo modo para gerar as interfaces gráficas e tem a vantagem de assegurar compatibilidade com os principais navegadores. O que a ferramenta faz é, a partir da API Java que ela disponibiliza, gerar um código JavaScript que será interpretado pelos navegadores. Com isso, a interface gráfica pode ser totalmente construída através da linguagem Java utilizando a API fornecida pelo GWT. Por exemplo, para criar o botão “Opções” no módulo cliente do SincroBUS, basta utilizar o seguinte código Java:

```
Button botãoOpções = new Button("Opções");
```

A partir do cliente criado através do GWT foi possível acessar os recursos Web disponibilizados pelo módulo servidor e mostrar as informações na interface gráfica. No módulo cliente é possível visualizar o itinerário da linha de ônibus, localizar um ônibus em seu trajeto pela linha Troncal-10, visualizar informações de um ônibus e consultar os horários das saídas de ônibus.

3.5. Estudo e Aplicação do JavaScript

O JavaScript é uma linguagem de programação voltada principalmente para navegadores Web que possui como características principais a orientação a objetos baseada em protótipos e tipagem fraca e dinâmica. Os estudos do JavaScript foram importantes para compreender o conceito de orientação a objetos baseada em protótipos que difere da orientação a objetos do Java que por sua vez é baseada em classes.

No subprojeto Sistema Web é utilizado o GWT que é uma ferramenta que através da linguagem Java gera um código JavaScript para a criação de aplicações com a tecnologia AJAX. Essa tecnologia é composta por JavaScript e XML que por sua vez é um formato extensível para a marcação de documentos.

O JavaScript permite que sejam realizadas requisições HTTP através do objeto

`XMLHttpRequest`. Essas requisições são baseadas em XML. Através do JavaScript é possível que se crie um cliente Web que será executado em um navegador e fará requisições para recursos Web disponíveis no servidor.

Apesar do JavaScript não ser usado diretamente no projeto, o estudo dele justifica-se pelo fato de que o GWT (tecnologia utilizada diretamente no projeto) gera um código JavaScript. Por isso, o estudo do JavaScript bem como o do AJAX foi importante para facilitar o futuro estudo o GWT.

3.6. Estudo e Aplicação do JSON

O JSON é um formato de intercâmbio de dados que faz parte da notação de objeto do JavaScript, mas que não necessariamente é utilizado com JavaScript. O JSON foi o formato de dados utilizado para representar os recursos que foram desenvolvidos no módulo servidor do subprojeto Sistema Web. A vantagem do JSON é que ele é um formato leve e de fácil conversão para outros formatos.

O JSON possui dois elementos fundamentais: a coleção de pares chave e valor e a lista ordenada. A coleção é representada por chaves (`{}`) e dentro dela existem pares de chave e valor separados por vírgula. A chave deve ser um texto, enquanto que um valor pode ser outra coleção, uma lista, um número, um texto, `true`, `false` ou `null`. Já a lista é uma estrutura englobada por colchetes (`[]`) que possui valores separados por vírgula. Esses valores podem ser os mesmos que os valores das coleções.

Exemplo de dado JSON:

```
{"metaDados": {"nome": "Linha troncal-10", "campos": [{"somenteLeitura": false}, {"nomeDeExibicao": "Caminho"}]}
```

3.7. Estudo e Aplicação do Hibernate

O Hibernate é uma ferramenta Java que auxilia na tarefa de realizar a persistência de dados. Entre outras coisas, ele fornece maneiras para acessar, alterar ou excluir dados em um banco de dados SQL.

O Hibernate trabalha com anotações Java. As anotações são o ponto chave que fazem com que realizar a persistência de dados se torne uma tarefa simplificada. Essas anotações servem para dizer ao Hibernate o que deve ser persistido. Com isso, a ferramenta permite a realização do mapeamento objeto-relacional para que um objeto Java possa ser persistido em um bando de dados SQL.

Com a utilização do *framework* não é necessário se preocupar com o SGBD (Sistema de Gerenciamento de Banco de Dados) que está sendo utilizado, pois para se comunicar com o SGBD o Hibernate utiliza um *driver* e em geral este já está implementado para boa parte dos SGBDs atuais. Caso exista a necessidade de trocar de SGBD, então basta alterar o *driver* utilizado pelo Hibernate. Isso faz como que se tenha uma gama de opções bastante variada na escolha do SGBD e que a alteração do SGBD utilizado possa ser realizada com facilidade.

No exemplo abaixo temos a classe `Onibus`. Para que seja possível persistir uma instância dessa classe é necessário realizar o mapeamento objeto-relacional. No Hibernate esse mapeamento é feito através das anotações que são precedidas pelo símbolo `@` (arroba).

```
@Entity
public class Onibus {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long codigo;
  @Column(length = 50, nullable = false)
  private String usuario;
  @Column(length = 50, nullable = false)
  private String senha;
  …
}
```

A primeira anotação (`@Entity`) indica que uma instância da classe `Onibus` é uma entidade que pode ser persistida e será mapeada em uma tabela no banco de dados. Após isso, é necessário definir o mapeamento dos atributos que resultarão nas colunas da tabela. A anotação `@Id`, antes do atributo `codigo` indica que este atributo é uma chave primária. Já a anotação `@GeneratedValue(strategy =`

`GenerationType.AUTO)` determina que a chave primária será gerada automaticamente. Existe também duas anotações `@Column(length = 50, nullable = false)` que são precedidas dos atributos `usuario` e `senha`. Essas anotações declaram que estes dois atributos serão uma coluna da tabela, terão tamanho 50 e não poderão ser nulos.

O Hibernate ainda possui um arquivo de configuração (`hibernate.cfg.xml`). Através desse arquivo será estipulado quais classes representarão objetos que serão mapeados para o banco de dados, o *driver* do SGBD utilizado, e outras configurações como as credenciais (usuário e senha) para acesso ao SGBD.

3.8. Georreferenciamento

O projeto do SincroBUS possuiu algumas atividades relacionadas a georreferenciamento. Entre essas atividades está o levantamento da localização de pontos de parada pertencentes ao itinerário linha de ônibus Troncal-10.

O processo no qual a partir de um endereço é obtida uma coordenada geográfica (latitude e longitude), é chamado de geocodificação. Já o processo contrário, onde a partir de um coordenada geográfica se obtém um endereço é chamado de geocodificação reversa. A geocodificação reversa facilita a utilização de uma mapa por pessoas, pois a partir de uma coordenada geográfica, fornecida por um dispositivo de GPS por exemplo, é possível obter um endereço. A vantagem da utilização do processo de geocodificação reversa está no fato de que endereços são mais compreensível por pessoas do que coordenadas geográficas.

No projeto do SincroBUS foi utilizada a API (*Application Programming Interface*) do Google Maps para realizar o processo de geocodificação. O acesso a esta API é feita através da comunicação pelo protocolo HTTP que a partir de requisições feitas às URIs fornecidas pelo Google Maps, pode acessar os serviços disponibilizados. Entre os serviços disponíveis estão a geocodificação, geocodificação reversa, cálculo de distância entre duas coordenadas geográficas e outros.

No contexto do projeto SincroBUS foi necessário realizar o georreferenciamento do itinerário da linha Troncal-10. Como oficialmente não existem as

coordenadas de cada ponto de parada dessa linha, foi necessário realizar uma análise de endereços e fazer a coleta dos pontos.

A tarefa de obter a geolocalização dos pontos de parada do itinerário da linha Troncal-10 foi dividida em três etapas: a primeira consistiu em realizar a geocodificação de um conjunto de endereços de pontos de parada. Na etapa seguinte, com a utilização das coordenadas geográficas obtidas no passo anterior, foi realizado o traçado dos pontos de parada e o resultado foi exibido em um mapa através do Google Maps. A última etapa consistiu em analisar o mapa gerado na etapa anterior e filtrar alguns dos pontos de parada, pois alguns deles não faziam parte do itinerário da linha Troncal-10.

A geolocalização dos pontos de parada da linha Troncal-10 permitiu traçar em uma mapa o itinerário da linha, bem como apresentar os pontos de parada no mesmo mapa. A ***Figura 1*** apresenta uma visualização do sistema cliente do SincroBUS. Nela é possível perceber o resultado do processo de geocodificação sendo possível visualizar os pontos de parada coletados (representados pela letra P) e o itinerário da linha (traçado em vermelho).

3.8.1. API de Geocodificação do Google Maps

O Google Maps fornece uma API para geocodificação. O acesso a esta API pode ser feito através de requisições HTTP. A URI base para acesso a API é: `http://maps.googleapis.com/maps/api/geocode/`**`output`**`?`**`parameters`**. Onde **`output`** indica o formato dos dados que serão recebidos como resposta, podendo ser JSON ou XML e **`parameters`** deve ser substituído por um ou mais parâmetros que serão explicados a seguir.

Os parâmetros devem ser separados pelo caractere `&` (caso exista mais de um). Os dois parâmetros principais são `address` e `latlng`, sendo um desses dois deve ser obrigatoriamente fornecido. Caso o parâmetro `address` seja fornecido, então o serviço realizado será a geocodificação de um endereço de tal forma que seja retornada uma ou mais (no caso de existirem endereços ambíguos) coordenadas geográficas. No caso do parâmetro `latlng` ser fornecido, então será realizada a geocodificação reversa, sendo retornado o endereço da coordenada geográfica fornecida (caso exista).

Além de ser obrigatório a passagem ou do parâmetro `address` ou do parâmetro `latlng`, outro parâmetro de passagem obrigatória é o `sensor`. O `sensor` pode receber um valor `true` ou `false` e ele indica se a solicitação vem ou não de um dispositivo com sensor de localização. Existem ainda outros parâmetros opcionais que podem ser usados, mas estes não foram necessários no escopo do projeto.

Por exemplo, para obter a geocodificação do endereço **R. São Paulo, - Itoupava Seca, Blumenau - SC, Brasil**, então faríamos uma requisição HTTP GET à seguinte URI:

`http://maps.googleapis.com/maps/api/geocode/`**`json`**`?`**`address`**`=R.%20São%20Paulo%20-%20Itoupava%20Seca,%20Blumenau%20-%20SC,%2089030-000,%20Brasil&`**`sensor`**`=false`.

Outro exemplo, é obter a geocodificação reversa da coordenada **(-26.8996592, -49.0863542)**. Para isto, realizaríamos um requisição HTTP GET ao seguinte endereço:

`http://maps.googleapis.com/maps/api/geocode/`**`json`**`?`**`latlng`**`=-26.8996592,-49.0863542&`**`sensor`**`=false`.

O resultado retornado em ambos exemplos estaria no formato JSON.

4. CONCLUSÃO

Os resultados alcançados foram satisfatórios e ocorreram dentro do previsto. O aprendizado inicial dos conceitos mais importantes facilitou o estudo das tecnologias estudadas posteriormente e a utilização prática dessas tecnologias facilitou a fixação dos conceitos aprendidos. Outro fator importante para o aprendizado e que melhorou os resultados alcançados foi o apoio de outros bolsistas do projeto que auxiliaram no esclarecimento de dúvidas em relação as tecnologias estudadas.

O estudo e aplicação das tecnologias envolvidas no projeto contribuíram bastante para o aprendizado e foram importantes para esclarecer vários pontos sobre a utilização da Web como plataforma para o desenvolvimento de aplicações. Foi possível perceber o grande potencial da Web, evidenciando suas vantagens como baixo acoplamento e alto nível de escalonabilidade.

Além das tecnologias fortemente consolidadas da Web foi possível estudar e utilizar na prática diversas ferramentas atuais que dão suporte para o desenvolvimento de *software*. Ferramentas como o Restlet, Hibernate e GWT foram estudas e através da utilização dessas ferramentas foi possível adquirir um leque de conhecimentos que futuramente poderão ser expandidos.

As atividades desenvolvidas pelo aluno trouxeram outros benefícios que vão além da aquisição de novos conhecimentos na área de desenvolvimento de sistemas Web. Isso se deve ao fato de que apesar das atividades desenvolvidas serem restritas ao subprojeto sistema Web, foi possível adquirir conhecimentos referentes a área de controle da automação, já que o projeto SincroBUS é englobado nesta área. Com isso, foram vistos conceitos que permitiram agremiar conhecimentos que estão fora da área de sistemas Web, conhecimentos como controle de semáforos, SCADA, estimativa de tempo de chegada, AVL e outros.

REFERÊNCIAS BIBLIOGRÁFICAS

CROCKFORD, D. *The application/json Media Type for JavaScript Object Notation (JSON).* RFC 4627, IETF, 2006.

FIELDING, R. Architectural Styles and the Design of Network-based Software Architectures. 100 p., University of California, 2000.

FIELDING, R. et al., *Hypertext Transfer Protocol - HTTP/1.1*, RFC 2616, IETF, 1999.

GOOGLE DEVELOPERS. *Google Web Toolkit developers guide*. Disponível em: <https://developers.google.com/web-toolkit/doc/2.4/DevGuide>.

GOOGLE DEVELOPERS. *The Google Geocoding API*. Disponível em: <http://developers.google.com/maps/documentation/geocoding>.

HIBERNATE TEAM. *Hibernate Developer Guide*. Red Hat, Inc, 2011. Disponível em <https://docs.jboss.org/hibernate/orm/4.1/devguide/en-US/html_single/>.

NOELIOS CONSULTING. *Restlet 2.0 - Tutorial*. 2010. Disponível em: <http://www.restlet.org/documentation/2.0/tutorial>.

POLÔNIA, P. V. *Proposta de arquitetura orientada a recursos para SCADA na Web,* Universidade Federal de Santa Catarina, 2011.

RICHARDSON, L.; RUBY, S. RESTful Web Services. 1. ed, Sebastopol: O'Reilly Media, Inc, 2007.

TOBALDINI, R. G. *Aplicação do Estilo Arquitetural REST a um Sistema de Congressos,* Universidade Federal de Santa Catarina, 2008.