UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS LAZARI DA SILVA

Agrupamento de Marcas em Mapas Georreferenciados

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Carlos Alberto Heuser

Porto Alegre
2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Lazari da Silva, Lucas

Agrupamento de Marcas em Mapas Georreferenciados / Lucas Lazari da Silva. – Porto Alegre: CIC da UFRGS, 2014.

44 f.: il.

Trabalho de conclusão (graduação) – Universidade Federal do Rio Grande do Sul. Bacharelado em Ciência da Computação, Porto Alegre, BR–RS, 2014. Orientador: Carlos Alberto Heuser.

I. Heuser, Carlos Alberto. II. Título.

AGRADECIMENTOS

Agradeço a toda a minha família e meus amigos. Principalmente aos meus pais, Ney e Zilá, e aos meus irmãos, Tales e Breno, por todo apoio e suporte durante toda minha trajetória acadêmica.

Gostaria também de agradecer à UFRGS, especialmente aos funcionários e professores do Instituto de Informática por todos os serviços prestados e conhecimentos repassados durante todos estes anos de curso.

Finalmente, gostaria de agradecer minha namorada Mariana por todo apoio motivacional, emocional, afetivo e logístico, principalmente nos momentos mais difíceis do curso. Também gostaria de agradecer à Laila, por me fazer companhia nas longas madrugadas programando na frente do computador. Sem o apoio de vocês, este trabalho não se tornaria realidade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
GEDCOM	Genealogical Data Communication
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MOD	Modifications
PHP	PHP Hypertext Processor
TNG	The Next Generation of Genealogy Sitebuilding©
UI	User Interface
URL	Universal Resource Locator
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Uma das áreas do conhecimento humano de maior interesse em se realizar pesquisas é a genealogia. É uma tendência natural do ser humano de querer buscar informações sobre seus antepassados, especialmente informações sobre locais de origem e de eventos importantes na história da árvore genealógica de sua família.

Com os recursos tecnológicos da atualidade, surgem diversas opções de sistemas para gerenciar esses tipos de informação, entre elas o TNG (The Next Generation of Genealogy Sitebuilding©), sistema para o qual foi desenvolvido este trabalho e que possui uma funcionalidade de disposição de dados georreferenciados utilizando a API do Google Maps.

Visando melhorar a experiência de uso desta funcionalidade, tanto em termos visuais quanto em termos de desempenho, este trabalho apresenta uma série de modificações na solução já existente, como alterações de layout do mapa, melhorias de usabilidade e modificações na arquitetura de troca e processamento de dados entre cliente e servidor, com o objetivo de tornar possível a utilização de bases de dados de grande porte.

Palavras-chave: Genealogia, TNG, agrupamento, Google Maps, PHP, AJAX, JavaScript, JSON.

Marker Clustering in Georeferenced Maps

ABSTRACT

One of the most interesting research fields in human knowledge is geneaolgy. It's natural of human beings to search for information about their ancestors, specially about places where families were formed and remarkable events that happened to members of his genealogical tree.

With all the technology resources nowadays, many options of systems to manage this kind of data are avaliable. Among those options there is a software called TNG (The Next Generation of Genealogy Sitebuilding©), a system that served as the starting point for the development of this paper and has a feature that shows geocoded data on a map built on top of Google Maps API.

Having improvement of usability and performance of this feature as its main goals, this paper presents several changes made on the already existing solution, like map layout changes, improvements in usability items and data processing architecture changes between client and server, aiming for the possibility of usage of large databases with the map.

Keywords: Genealogy, TNG, clustering, Google Maps, PHP, AJAX, JavaScript, JSON.

1 INTRODUÇÃO

O sistema The Next Generation of Genealogy Sitebuilding © (TNG, 2014) é um sistema de gerência de dados genealógicos, desenvolvido utilizando a linguagem de programação PHP[1]. Com ele, podemos armazenar e gerenciar diversos dados referentes à árvores genealógicas familiares, como eventos, fotos e lugares.

Uma das principais vantagens do TNG é o fato dele ser distribuído com código fonte aberto. Isso possibilita que a arquitetura do software possa ser estudada de forma bastante analítica e detalhada. Além disso, o próprio sistema fornece a possibilidade de expansão de suas funcionalidades, através da instalação de MODS desenvolvidos pela comunidade usuária do sistema. A partir destas possibilidades, foi desenvolvido em um trabalho anterior, uma funcionalidade que permite que o usuário visualize lugares onde ocorreram eventos de uma determinada família dentro do TNG, utilizando a API do Google Maps e as informações georreferenciadas que o sistema permite armazenar. Porém, esta funcionalidade apresentava problemas de desempenho quando utilizada em conjunto com uma base de dados muito grande, comprometendo muitas vezes sua utilização.

Este trabalho tem como objetivo apresentar uma série de melhorias desenvolvidas em cima desta funcionalidade realizada em um trabalho anterior (SILVEIRA, 2013), como modificações dos elementos visuais do mapa e na arquitetura da solução, balanceando de forma adequada as rotinas de processamento de dados entre cliente e servidor, sendo a principal dessas rotinas o algoritmo que realiza o agrupamento de marcas de lugares no mapa.

O documento está divido em cinco capítulos. O capítulo dois faz uma breve descrição do sistema TNG, que serve como base para o trabalho desenvolvido. O capítulo três aborda o funcionamento da solução existente, explicando o mapa georreferenciado desenvolvido anteriormente e os principais problemas encontrados em seu uso. O capítulo quatro descreve as melhorias implementadas para atacar os problemas da solução anterior, abordando aspectos da arquitetura do software, usabilidade e a descrição do algoritmo utilizado para realizar o agrupamento de marcas no mapa. O capítulo cinco faz uma conclusão do trabalho e discute possíveis melhorias a serem realizadas em trabalhos futuros.

[1] `http://www.php.net`

2 TNG: THE NEXT GENERATION OF GENEALOGY SITEBUILDING

Neste capítulo é feita uma breve apresentação do sistema TNG e suas principais funcionalidades, demonstrando as principais premissas na qual o desenvolvimento deste trabalho se baseia.

2.1 Visão Geral do Software

O TNG é um software que realiza a gestão de dados de árvores genealógicas, projetado para executar em um servidor web. Idealizado originalmente por Darrin Lythgoe, o sistema foi desenvolvido utilizando a linguagem de programação PHP, armazenando suas informações em um banco de dados MySQL[1]. Sua distribuição se dá através da comercialização de licenças de software on-premise[2]: o fornecedor disponibiliza ao comprador os pacotes necessários para instalação e a responsabilidade de instalação e disponibilidade do sistema é do comprador da licença. No caso do TNG, os pacotes são disponibilizados com os códigos fonte aberto. Isso possibilita um estudo detalhado de como é arquiteturado o sistema, e como essa arquitetura pode ser utilizada para o desenvolvimento de expansões para o sistema.

2.2 Funcionalidades básicas

Por se tratar de um software de genealogia, as principais funcionalidades do TNG estão centradas na gestão de árvores genealógicas, que podem ser múltiplas dentro do sistema.

Uma árvore genealógica consiste em uma hierarquia de pessoas que possuem conexões familiares entre si, como ascendência e descendência. Informações complementares referentes a estas pessoas também fazem parte de uma árvore genealógica: fotos, documentos, datas e localizações de eventos importantes na história da família complementam os dados contidos em uma árvore. A figura 2.1 mostra a representação uma árvore genealógica no TNG, mostrando os ancestrais de uma pessoa.

[1] `http://www.mysql.com`
[2] `http://en.wikipedia.org/wiki/On-premises_software`

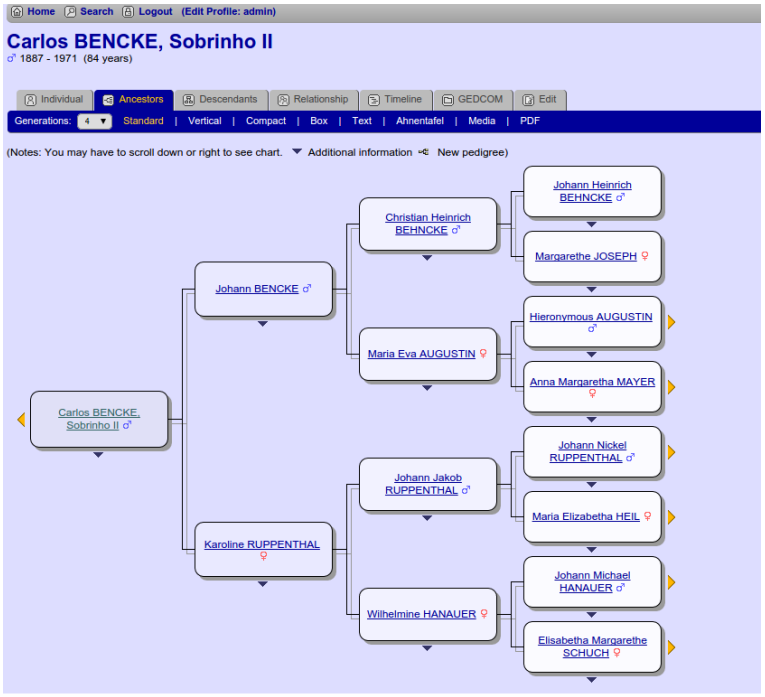

Figura 2.1: Visualização dos ancestrais de uma pessoa no TNG

2.2.1 Entrada dos dados genealógicos no sistema

No TNG, a entrada dos dados de uma pessoa em uma árvore genealógica pode ocorrer de duas formas. A primeira forma é através de um cadastro de pessoas existente dentro do sistema, onde o usuário preenche seus principais dados, como data e local de eventos, anotações e relacionamentos com outras pessoas. Uma pessoa deve obrigatoriamente estar vinculada a uma árvore genealógica previamente cadastrada. A imagem 2.2 mostra a tela de inclusão de pessoas no TNG.

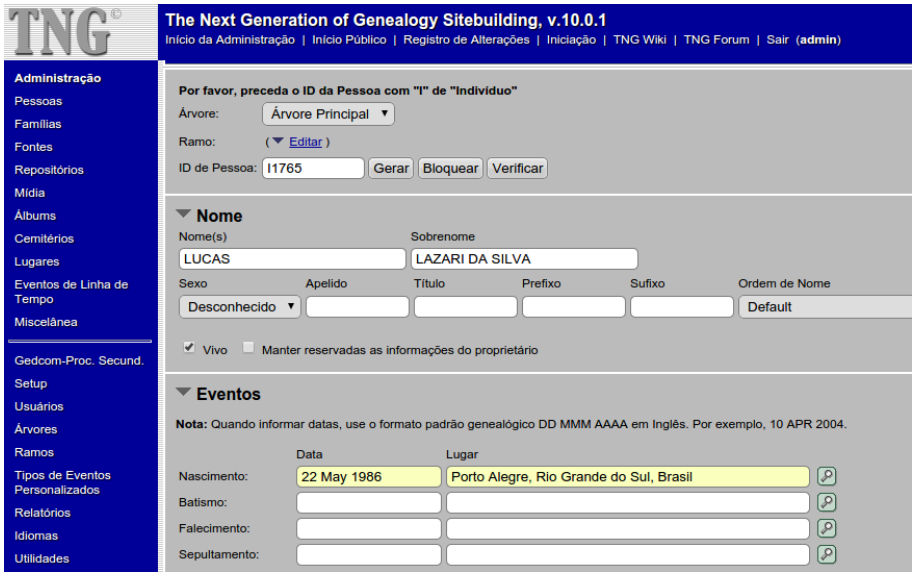

Figura 2.2: Tela de inclusão de uma pessoa à uma árvore genealógica

A segunda forma de entrada de dados no sistema é através da importação de um arquivo no formato GEDCOM[3] (Genealogical Data Communication). Este formato foi criado pelo Departamento de História Familiar da Igreja de Jesus Cristo dos Santos dos Últimos Dias (Igreja Mórmon) com o objetivo de auxiliar na pesquisa de dados genealógicos. Trata-se de um arquivo de texto puro que possui dados genealógicos sobre pessoas e estruturado de forma que essas pessoas possuam relações familiares entre si. Hoje em dia, a maioria dos softwares de genealogia existentes suporta a importação e exportação de dados genealógicos nos formato GEDCOM, inclusive o TNG.

A importação de um arquivo GEDCOM no TNG deve ser feita obrigatoriamente para dentro de uma árvore do sistema, existindo a opção para o usuário criar uma nova árvore no momento da inclusão, se for necessário. Existem diversas configurações para a rotina de importação do arquivo, como atualização de pessoas já existentes em uma árvore, importação de dados multimídia e importação de dados georreferenciados com latitude e longitude.

Tanto nas inserção manual de pessoas como na importação de arquivos GEDCOM, devemos dar destaque para as informações de tipos de eventos de uma pessoa que o sistema permite armazenar. Estes dados servem como base para os filtros do mapa georreferenciado desenvolvido anteriormente e modificado neste trabalho. Para cada tipo de evento de uma pessoa, é permitido cadastrar uma data e localização associado a ele. O TNG adota como padrão, de forma estática, os seguintes eventos: nascimentos, batizados, casamentos, falecimentos e sepultamentos. Também existe a possibilidade de se cadastrar eventos customizados no sistema de forma dinâmica, de acordo com a necessidade do usuário. Isso aumenta o poder de detalhamento que as informações de uma pessoa pode ter em uma árvore genealógica.

2.2.2 Geocodificação de lugares

A partir do momento em que temos a informação sobre o lugar de um determinado evento, é necessário que o sistema obtenha suas informações de geocodificação (latitude e longitude). Desta forma, o TNG possui uma funcionalidade de geocodificação automática. Esta funcionalidade encontra-se na tela de edição de um lugar, e utiliza o serviço de Geocodificação [4] disponibilizado pela Google.

A rotina automática de geocodificação do TNG faz uma leitura em sua base de dados, procurando a string que descreve o lugar em que ocorreu um determinado evento. Depois, a rotina executa uma requisição HTTP (Hypertext Transfer Protocol) para a URL (Universal Resource Locator) do serviço utilizando a string encontrada como parâmetro. Neste momento, o serviço retorna um arquivo estruturado como resposta, que é processado pelo TNG para buscar os dados de geocodificação. Então, no final do processo, o TNG alimenta sua base de dados com os dados de latitude e longitude do lugar em que ocorreu o evento em questão.

Como a rotina descrita acima depende de um serviço fornecido por terceiros, a geocodificação de lugares não pode depender exclusivamente deste serviço, pois pode ocorrer de o mesmo não disponibilizar de forma correta a informação desejada, contendo erros de precisão ou até mesmo não possuir a informação.

Em alternativa à esta rotina de geocodificação automática, o TNG também desenvolveu uma funcionalidade de geocodificação manual, que se encontra na mesma tela de edição de lugares. Esta funcionalidade utiliza um mapa interativo da API do Google Maps,

[3] `http://www.mormonwiki.com/GEDCOM`

[4] `https://developers.google.com/maps/documentation/geocoding/`

que permite que o usuário posicione um marcador no mapa manualmente, indicando a posição exata do lugar.

A imagem 2.3 mostra esta funcionalidade de geocodificação manual no TNG realizada sobre um ponto específico da cidade de Porto Alegre.

Figura 2.3: Geocodificação manual de um lugar no TNG

Com essas duas funcionalidades de geocodificação, o TNG tem os recursos possíveis para armazenar os dados de latitude e longitude de lugares, que servem como base para o mapa georreferenciado desenvolvido neste trabalho.

2.2.3 Expansibilidade do Sistema

Para o desenvolvimento do mapa georreferenciado, foi explorado um dos principais chamarizes do TNG, que é o seu poder de expansibilidade. Como o software é entregue com o seus arquivos de scripts PHP abertos, o estudo da arquitetura do sistema se torna mais fácil para os desenvolvedores entusiastas do sistema. Pensando no crescimento colaborativo do sistema, o TNG disponibiliza uma funcionalidade de instalação de MODS que podem ser criados pela comunidade que utiliza o sistema. Esta funcionalidade realiza a gestão de todos os MODS instalados no TNG por parte do administrador do sistema, permitindo facilmente que os mesmo sejam instalados, removidos e configurados.

Um MOD consiste em um pacote de instalação que contém uma funcionalidade nova baseada na arquitetura existente do TNG. Este pacote contém os códigos fonte da funcionalidade desenvolvida e um arquivo de configuração que, ao ser lido pelo TNG, realiza as modificações necessárias no código fonte do TNG, com o objetivo de adaptar a nova funcionalidade desenvolvida ao sistema. Um MOD também pode conter parâmetros específicos, que podem ser alterados pelo administrador do sistema para adaptar o MOD de acordo com as suas necessidades.

Neste capítulo, conhecemos um pouco do funcionamento do TNG e as características do sistema que permitiram o desenvolvimento do mapa de lugares. No próximo capítulo, é feita uma descrição da primeira versão do mapa georreferenciado de lugares, suas principais funcionalidades e problemas constatados.

3 MAPA GEORREFERENCIADO DE LUGARES

Neste capítulo, é feita a descrição da solução desenvolvida anteriormente para navegação em mapas georreferenciados dentro do sistema. No final do capítulo, é feito um relato ds problemas encontrados nesta solução, que surgem como motivação para o desenvolvimento deste trabalho.

3.1 Visão Geral da Funcionalidade

Como vimos na seção anterior, o TNG possibilita que sejam registrados dados de geocodificação referentes a eventos de pessoas, tanto de forma manual como através da importação de arquivos GEDCOM. Inicialmente, o sistema utilizava as informações de latitude e longitude somente em um mapa que mostra os eventos de uma única pessoa, carecendo de uma funcionalidade de visão geral de todos os lugares de eventos contidos no banco de dados.

A partir deste contexto foi desenvolvido, em um trabalho anterior, a primeira versão do mapa georreferenciado de lugares de eventos, permitindo ao usuário uma visualização de todos os lugares geocodificados que constam no banco de dados do sistema. Nas próximas subseções, serão listadas as principais funcionalidades deste mapa.

3.1.1 Tela Principal

A tela principal da funcionalidade consiste em um mapa que demonstra marcadores agrupados, representando os lugares de eventos registrados no banco de dados. O usuário tem a possibilidade de aplicar filtros e visualizar estatísticas de acordo com as opções de filtros escolhidas. A medida em que o usuário clica nos filtros, o mapa é atualizado com os marcadores dos lugares que satisfazem os critérios estabelecidos. A imagem 3.1 mostra a tela principal do mapa.

3.1.2 Filtro por árvore genealógica

Existe opção de realizar um filtro pelos lugares referentes aos eventos de uma única árvore genealógica. Este filtro foi criado listando todas as árvores cadastradas no sistema em uma caixa de seleção dropdown.

3.1.3 Filtro por tipos de evento

É possível realizar filtros pelos cinco tipos de evento definidos como padrão pelo TNG. sendo eles: nascimentos, batismos, casamentos, falecimentos e sepultamentos. Adicionalmente, é possível filtrar pelos eventos customizados pelo usuário, que estão disponíveis, agrupados em sua totalidade, na opção “Outros”.

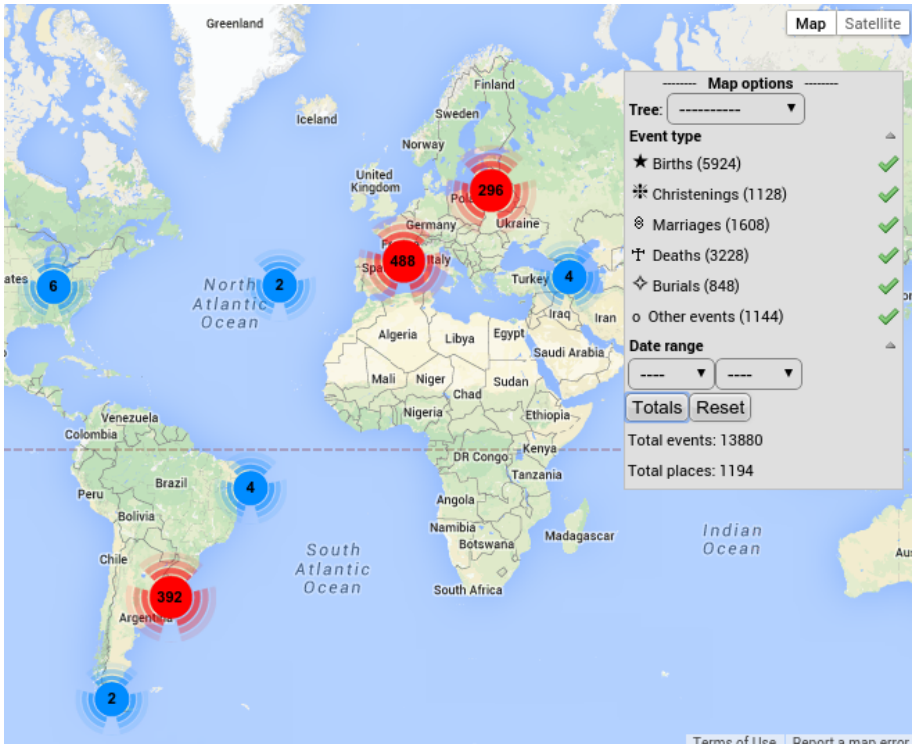

Figura 3.1: Tela principal da versão antiga do mapa de lugares

3.1.4 Filtro por intervalo de datas

A partir das datas registradas para cada evento na entrada de dados do TNG, é possível realizar um filtro determinando um intervalo de anos no qual ocorreram os eventos. Este intervalo é criado a partir de duas caixas de seleção dropdown disponibilizadas nos filtros. Uma caixa representa o ano inicial do intervalo e a outra representa o ano final do intervalo.

A funcionalidade permite que os valores que são listados nestes dois campos sejam configurados pelo administrador do sistema, através dos parâmetros criados para o MOD. Nas próximas seções, estes parâmetros serão detalhados.

3.1.5 Estatísticas

A medida em que o usuário vai utilizando o mapa e configurando os filtros disponíveis, são demonstradas informações estatísticas sobre os lugares que estão no mapa. Abaixo estão listadas estas informações:

- Total de nascimentos
- Total de batismos
- Total de casamentos
- Total de falecimentos
- Total de sepultamentos
- Total de outros eventos
- Total de eventos
- Total de lugares

Essas estatísticas são contabilizadas considerando o mapa como um todo. Ou seja, mesmo que apenas uma parte do mapa esteja visível, os eventos e lugares que estão de fora do da área visível do mapa são considerados nos cálculos.

3.1.6 Interações com o mapa

Tendo à sua disponibilidade o mapa de lugares e os filtros, o usuário pode interagir com a funcionalidade de diversas formas. À medida em que é alterado o zoom do mapa, os agrupamentos de lugares são atualizados. A imagem 3.2 demonstra a alteração de zoom em dois níveis diferentes, focando no continente europeu.

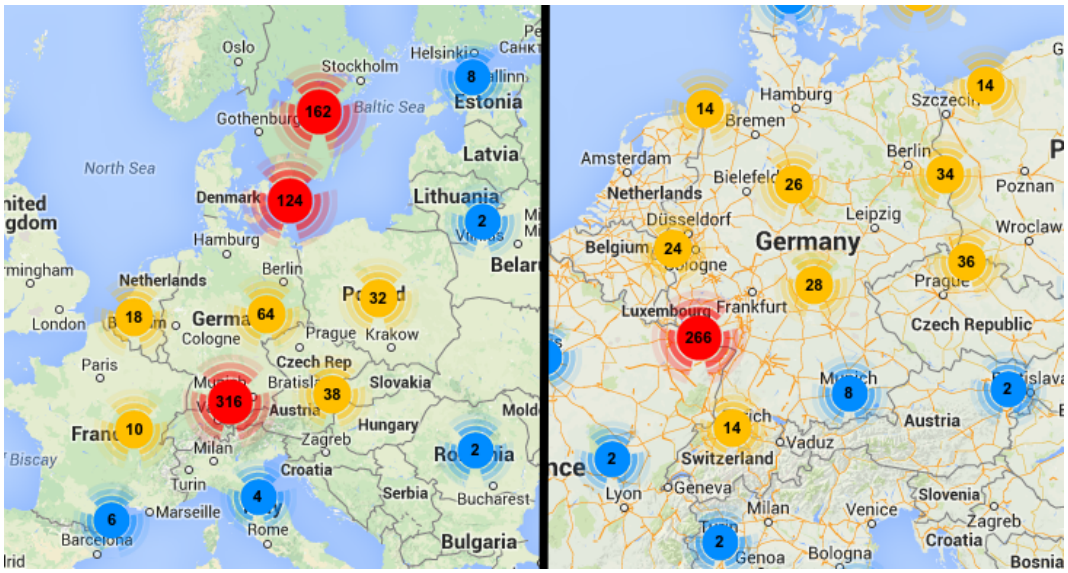

Figura 3.2: Comportamento do agrupamento de marcadores em diferentes níveis de zoom

O usuário também pode navegar pelo mapa clicando em cima dos agrupamentos. Desta forma, o mapa se ajusta automaticamente para que todos os marcadores contidos no agrupamento clicado sejam demonstrados.

A imagem 3.3 mostra a disposição dos agrupamentos antes (à esquerda) e depois (à direita) de um clique em cima do agrupamento localizado nas proximidades da cidade de Capão da Canoa.

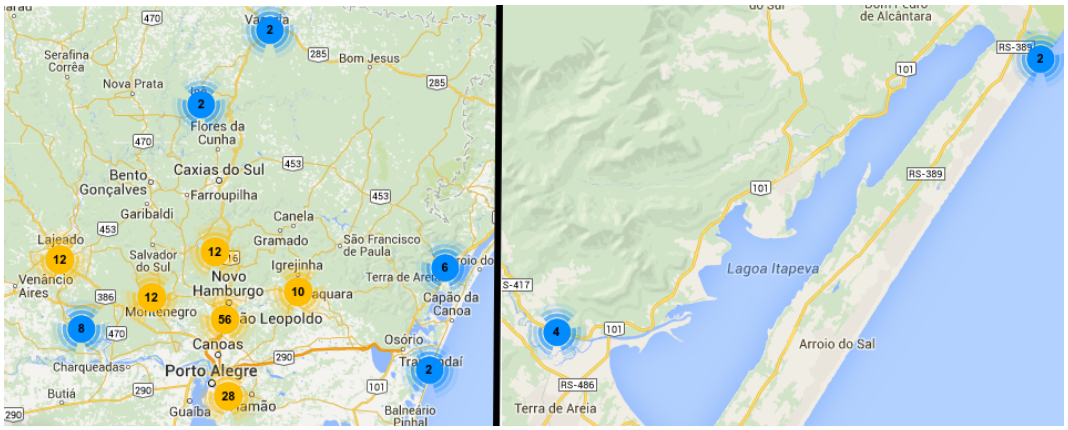

Figura 3.3: Ajuste do mapa após o clique sobre um agrupamento

A medida em que o mapa vai se aproximando do zoom máximo, os agrupamentos vão se desfazendo e os marcadores começam a aparecer individualmente. Se o usuário clicar em cima de um marcador individual, o sistema abre uma nova aba mostrando a tela de lugares nativa do TNG, onde é possível visualizar com mais detalhes os eventos que ocorreram neste lugar.

Porém, pode ocorrer que, ao chegar no zoom máximo permitido pelo mapa, o agrupamento ainda exista, por ainda estarem muito próximos. Isso impossibilita o usuário de clicar sobre o marcador individual do lugar e visualizar seus detalhes. Para resolver este problema, a versão antiga da funcionalidade realiza uma separação "artificial" dos marcadores do agrupamento. Por exemplo, se com o zoom máximo existir um agrupamento com n lugares, este mesmo agrupamento é transformado em n marcadores individuais com posicionamentos distintos. Na prática, isto compromete a veracidade da informação, pois altera a precisão da disposição dos lugares no mapa.

3.1.7 Parâmetros do MOD

Como foi mencionado no capítulo anterior, é possível que o desenvolvedor do MOD crie parâmetros para a funcionalidade, de forma que o administrador do sistema ajuste a funcionalidade de acordo com as suas necessidades. A tabela 3.1 lista os parâmetros criados para a versão anterior do MOD, com uma breve descrição de suas funcionalidades.

Tabela 3.1: Tabela descritiva dos parâmetros do MOD

Parâmetro	Descrição
Ano de início para o filtro de datas	Define o ano inicial no qual os valores dos campos de data serão listados.
Ano de fim para o filtro de datas	Define o ano final no qual os valores dos campos de data serão listados.
Intervalo de anos para o filtro de datas	Define o intervalo entre cada valor dos campos de data que serão listados.
Nível de zoom inicial do mapa	Define em qual nível de zoom o mapa será inicializado.
Altura da caixa que contém o mapa	Define a altura com a qual o mapa será renderizado na tela.
Latitude inicial	Define a latitude inicial na qual o mapa será centralizado ao ser inicializado.
Longitude inicial	Define a longitude inicial na qual o mapa será centralizado ao ser inicializado.
Árvore inicial	Define se o mapa deve mostrar os lugares de todas as árvores do sistema ou de uma árvore específica.
Mostrar eventos	Define se o mapa irá mostrar informações de eventos ou não, Caso este parâmetro seja desativado, o mapa mostrará apenas informações referentes a lugares.
Tamanho dos marcadores de lugares	Define o tamanho dos marcadores individuais a serem mostrados no mapa, que podem ser renderizados em seu tamanho normal ou de forma reduzida.

3.2 Arquitetura da Solução

Como mencionado nas seções anteriores, o TNG é um sistema distribuído com o código aberto, utilizando PHP como linguagem de programação e MySQL como mecanismo de banco de dados. Desta forma, o mapa de lugares também foi desenvolvido com estas tecnologias, adicionalmente com bibliotecas auxiliares escritas em JavaScript, como a API do Google Maps e a biblioteca `MarkerClusterer`.

3.2.1 Busca de Dados

No momento em que é iniciada a execução do mapa georreferenciado, o TNG realiza uma única requisição HTTP a um script PHP localizado no servidor. Este script tem a tarefa de buscar todos os dados de eventos e lugares que o mapa deve manipular. Estes dados estão localizados em um banco de dados MySQL. Após realizar esta consulta, o servidor retorna um arquivo HTML (Hypertext Markup Language) ao browser do cliente.

Desta forma, a funcionalidade não necessita realizar nenhuma outra requisição ao servidor para prosseguir com sua execução, realizando toda a manipulação e processamento dos dados no browser do cliente. Este processamento de dados realizado no cliente é feito utilizando a linguagem de programação JavaScript, uma linguagem client-side que é suportada pela maioria dos browsers da atualidade. Todas as rotinas de JavaScript a serem executadas são programadas de forma embutida nos scripts PHP, ou então em arquivos separados de extensão ".js", que são chamados de dentro dos scripts PHP.

3.2.2 API do Google Maps

Para auxiliar no desenvolvimento do MOD, foram utilizadas bibliotecas adicionais escritas em JavaScript, que fornecem o suporte necessário para a criação do mapa desenvolvido no trabalho. Uma destas bibliotecas utilizadas é a API do Google Maps (GOOGLE, 2014), que consiste em um conjunto de rotinas que facilitam a criação de mapas em páginas web.

Utilizando estas rotinas é possível renderizar o mapa mundial e criar marcadores em pontos específicos deste mapa. A imagem 3.4, em conjunto com o código JavaScript, demonstram a criação de um único marcador sobre o continente australiano.

```
var myLatlng = new google.maps.LatLng(-25.363882,131.044922);
var marker = new google.maps.Marker({
  position: myLatlng,
  map: map,
});
```

Figura 3.4: Criação de um marcador no Google Maps

Assim, no momento em que as informações de latitude e longitude dos lugares a serem demonstrados no mapa chegam no browser do cliente, são realizadas diversas chamadas à rotina de criação de marcador, uma para cada lugar georreferenciado encontrado no banco de dados.

3.2.3 Biblioteca MarkerClusterer

No momento em que todos os marcadores individuais já estão criados no mapa, o MOD faz uso da biblioteca MarkerClusterer[1], uma biblioteca JavaScript desenvolvida utilizando a API do Google Maps como base. Esta biblioteca realiza o agrupamento dos marcadores de acordo com a proximidade entre os mesmos. A imagem 3.5 demonstra o resultado do agrupamento de um conjunto de marcadores individuais utilizando a biblioteca `MarkerClusterer`.

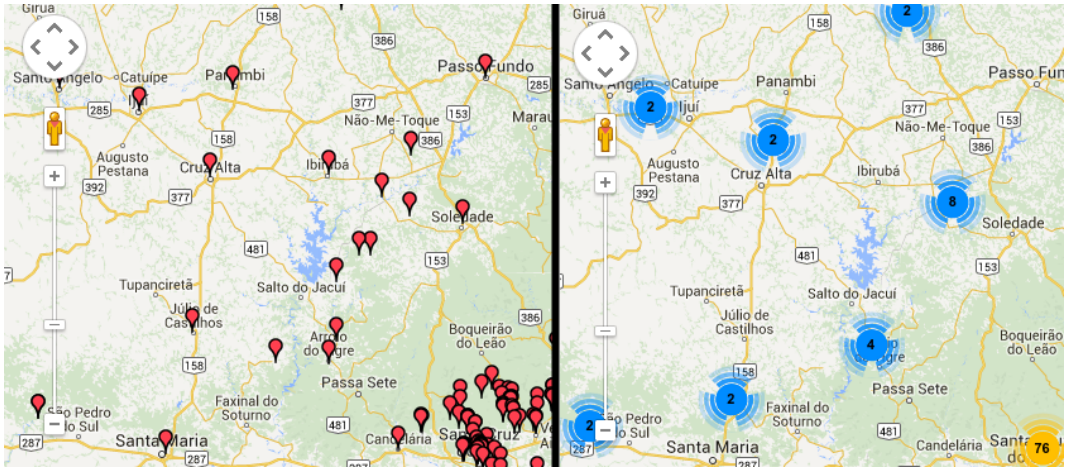

Figura 3.5: Agrupamento de marcadores utilizando a biblioteca MarkerClusterer

Como podemos notar na imagem 3.5, os marcadores são agrupados de acordo com a distância entre eles. Analisando o código fonte da biblioteca `MarkerClusterer`, foi constatado que esses cálculos de distância são realizados utilizando a fórmula de Haversine[2], uma importante equação utilizada na área da navegação que calcula a distância entre dois pontos a partir de suas coordenadas geográficas. Esta fórmula considera a distância em uma superfície esférica, o que pode causar problemas de precisão de acordo com a posição dos marcadores, já que o mapa construído é plano, baseado na projeção de Mercator [3]. Pensando nisso, a API do Google Maps possui rotinas de conversão de coordenadas geográficas em coordenadas cartesianas.

3.2.4 Problemas no tratamento de grandes volumes de dados

Como foi mencionado nas seções anteriores, a requisição dos dados georreferenciados ao servidor ocorre uma única vez no início da execução do mapa, sendo necessário que todos os lugares sejam enviados para o browser do cliente. Isto compromete bastante o desempenho do MOD caso a base de dados utilizada seja muito grande, já que o cliente deve processar todos os lugares nas etapas de criação e agrupamento dos marcadores. Nos

[1] `http://google-maps-utility-library-v3.googlecode.com/svn/trunk/markerclusterer/docs/reference.html`

[2] `http://en.wikipedia.org/wiki/Haversine_formula`

[3] `http://en.wikipedia.org/wiki/Mercator_projection`

casos em que se faz necessário demonstrar cerca de 5.000 lugares ou mais, o sistema trava e o mapa não é renderizado.

Este comprometimento do desempenho e do uso adequado do MOD serve como principal motivação deste trabalho. No próximo capítulo é apresentado uma reestruturação da arquitetura da versão antiga do mapa, onde são realizadas requisições assíncronas ao servidor, que passa a assumir a tarefa de agrupar os marcadores de acordo com as interações do usuário com o mapa.

Juntamente com esta mudança de arquitetura, também são realizadas diversas melhorias visuais e funcionais do MOD, que resolvem parte dos problemas citados durante este capítulo.

4 ALTERAÇÕES IMPLEMENTADAS NO MAPA

Neste capítulo é apresentada a versão final do mapa, demonstrando as melhorias implementadas a partir da solução anterior. Estas melhorias consistem em alterações realizadas nos filtros, estatísticas e uma reestruturação da arquitetura de busca e processamento dos dados demonstrados no mapa.

4.1 Alterações de Layout

Com o objetivo de melhorar a experiência de uso do mapa de forma geral, foram realizadas mudanças na disposição dos filtros customizados do mapa. Observando a imagem (número da imagem da tela principal antiga), podemos ver que os filtros sobrepõem boa parte do mapa, o que dificulta a visualização como um todo pelo usuário.

4.1.1 Tela principal

A imagem 4.1 mostra o resultado visual da nova disposição dos filtros no mapa.

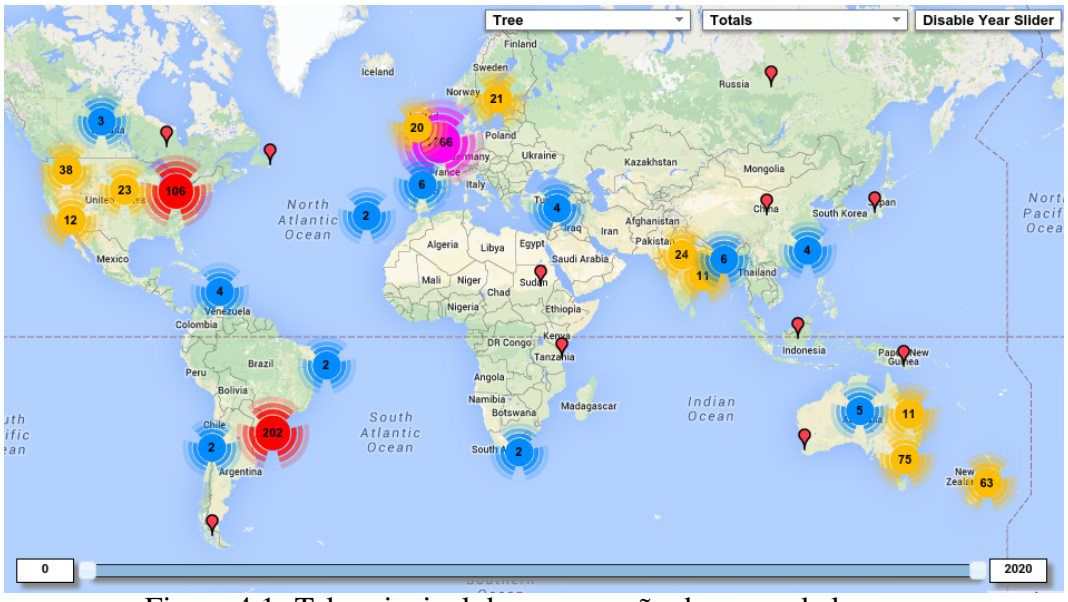

Figura 4.1: Tela principal da nova versão do mapa de lugares

Realizando a modificação das posições dos filtros para as partes superiores e inferiores, foi possível deixar a parte central com mais espaço para visualização do seu conteúdo, facilitando a interação do usuário com o mapa.

Os filtros da parte superior do mapa foram modificados utilizando uma biblioteca

chamada gdropdown.js[1], que permite que sejam criados controles customizados para o Google Maps mantendo a aparência destes controles semelhante a dos controles nativos do mapa. Esta biblioteca foi desenvolvida em JavaScript e utiliza em conjunto um arquivo CSS (Cascading Style Sheets) para estilizar os menus. Nas próximas seções são demonstradas as alterações realizadas em cada filtro.

4.1.2 Filtro por múltiplas árvores

Na versão antiga do mapa, este filtro permitia a seleção de uma única árvore genealógica. A nova versão possibilita que o usuário escolha múltiplas árvores , flexibilizando o uso e a visualização dos lugares no mapa.

Desta forma, foi utilizado a rotina de criação de menus dropdown da classe gdropdown.js, permitindo opções de múltipla escolha. Se o usuário clicar em cima deste campo, todas as árvores existentes no TNG são listadas. A imagem 4.2 demonstra o novo filtro de árvores e como o usuário interage com ele.

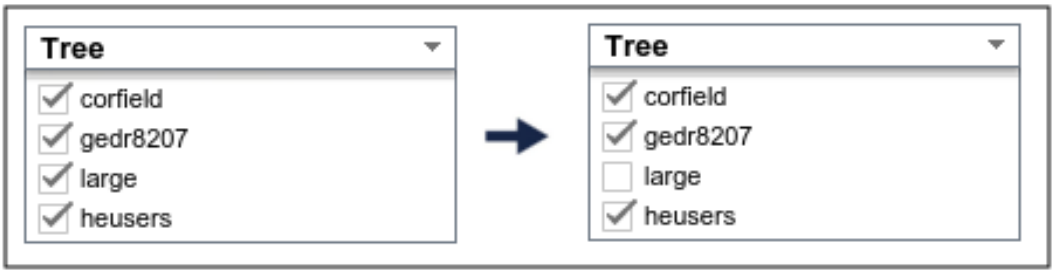

Figura 4.2: Seleção de múltiplas árvores no filtro

Por padrão, o mapa é inicializado com todas as árvores selecionadas. Com isso, o parâmetro do MOD que define a árvore inicial a ser demonstrada no mapa foi removido, pois perdeu sua utilidade após esta melhoria.

4.1.3 Filtro por tipos de eventos e estatísticas

O filtro por tipos de evento e as estatísticas foram agrupados no mesmo menu, que também foi criado como um campo dropdown da classe gdropdown.js. Na versão antiga do mapa, estes elementos já eram demonstrados de forma próxima, pois o foco dos dados estatísticos do mapa é justamente passar informações quantitativas a respeito dos tipos de evento possíveis.

Adicionalmente, foi realizada uma modificação na atualização dos valores quantitativos das estatísticas, de forma que sejam contabilizados somente os eventos de lugares que estão visíveis no mapa no momento. Portanto, qualquer alteração no posicionamento do mapa ou no nível de zoom irá atualizar as estatísticas. A imagem 4.3 mostra as estatísticas antes e depois do aumento do zoom do mapa sobre o sudoeste asiático.

[1] `http://vislab-ccom.unh.edu/~briana/examples/gdropdown/gdropdown.js`

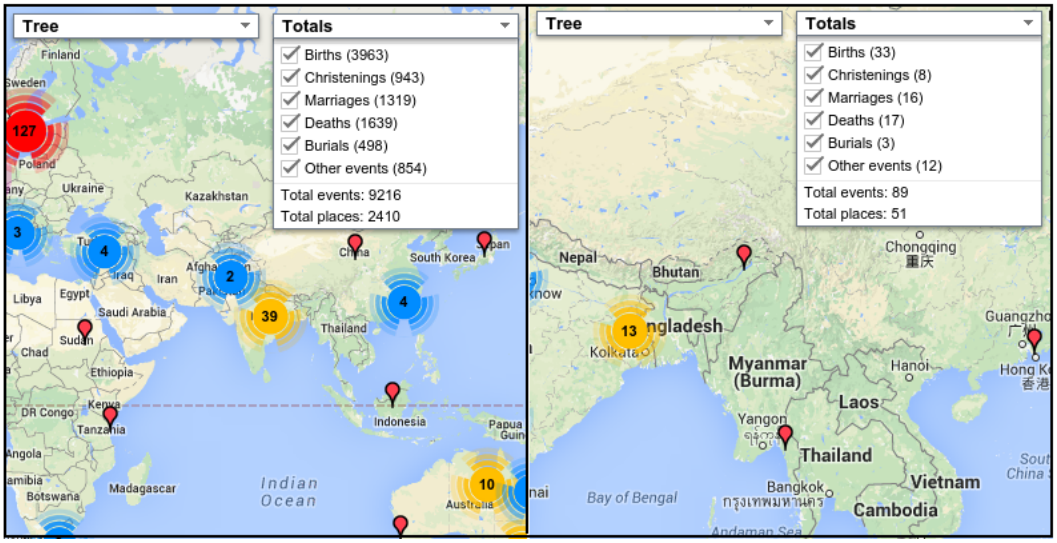

Figura 4.3: Atualização das estatísticas de eventos de acordo com os marcadores visíveis no mapa

O cálculo das estatísticas por tipo de evento só é realizada caso o parâmetro do MOD "Mostrar Eventos" esteja habilitado, conforme descrito na tabela 3.1. Caso contrário, estes valores e o totalizador de eventos aparecem zerados.

4.1.4 Filtro por datas

Como se pode observar na imagem 4.3, o filtro por data foi modificado para uma barra de rolagem localizada na parte inferior do mapa. Com esta barra de rolagem, o usuário tem a possibilidade de definir um período de eventos específicos, utilizando o mouse para arrastar as barras de ano inicial e ano final.

Esta barra de rolagem foi criada utilizando a biblioteca jQuery UI[2], composta por diversos plugins escritos em JavaScript e voltados para a criação de elementos visuais em páginas web. Como o TNG já faz uso da jQuery UI, não foi necessária a inclusão da biblioteca no MOD para utilizá-la.

O plugin utilizado para a criação da barra de rolagem é o Slider[3], que tem seus valores de limite superior e inferior definidos pelos parâmetros já existentes do MOD, que são os parâmetros de "Ano de início" e "Ano de fim" para o filtro de datas.

A imagem 4.4 mostra a interação do usuário com a nova barra de rolagem de datas, modificando o período em que ocorreram os eventos a serem demonstrados no mapa.

Figura 4.4: Barra de rolagem utilizada no filtro de datas

Na parte superior da imagem, a barra é demonstrada em seu estado inicial, com a data inicial no ano 0 e a data final no ano 2020. Estes valores são os valores extremos que o filtro por datas pode assumir, e são configurados nos parâmetros do MOD. Na parte inferior da imagem, a barra é demonstrada após a interação do usuário, com a data inicial no ano 1050 e a data final no ano de 1604.

[2] http://jqueryui.com/
[3] http://jqueryui.com/slider/

Adicionalmente, foi criado um botão para habilitar ou desabilitar a barra de rolagem de datas, com o objetivo de mostrar no mapa os lugares de eventos que não possuem a informação de data. A imagem 4.5 demonstra o botão criado, localizado na parte superior do mapa.

Figura 4.5: Botão que desabilita a barra de rolagem de datas

4.1.5 Caixa de diálogo de lugares

Conforme relatado no capítulo anterior, o mapa possuía o problema da existência não desejada de agrupamentos quando se atingia seu nível de zoom máximo. Na versão antiga, este problema foi contornado realizando um afastamento artificial entre os marcadores dos agrupamentos em questão.

Para manter a fidelidade do posicionamento dos marcadores no mapa, foi implementada uma funcionalidade que, se o usuário estiver utilizando o mapa em seu zoom máximo e clicar sobre um agrupamento de marcadores, uma caixa de diálogo é aberta listando todos os lugares que estão dentro do agrupamento, com seus respectivos links para a página nativa do TNG que detalha o lugar. A imagem 4.6 demonstra um exemplo desta caixa de diálogo após o clique sobre um agrupamento situado em Santa Cruz do Sul.

Figura 4.6: Caixa de diálogo listando os lugares de um agrupamento

Desta forma, um clique em cima de um agrupamento pode disparar dois eventos distintos. Se o mapa estiver em seu nível de zoom máximo, a caixa de diálogo de lugares é aberta. Caso contrário, o mapa se reajusta para mostrar os marcadores contidos no agrupamento clicado. Para implementar este controle do nível de zoom, foi necessário analisar os tipos de mapa permitidos pela API do Google Maps, pois cada um destes mapas possui um nível máximo de zoom distinto. A tabela 4.1 lista todos os tipos de mapa disponíveis e seus respectivos limites de zoom.

Tabela 4.1: Nível máximo de zoom por tipos de mapa do Google Maps

Tipo de mapa	Nível máximo de zoom
Estradas	21
Terreno	15
Satélite	19
Híbrido	19

Da mesma forma que a barra de rolagem criada para o filtro de datas, a caixa de diálogo de lugares foi criada utilizando a biblioteca jQuery UI, mais especificamente o

plugin Dialog[4], que utiliza elementos HTML de uma página para criar uma caixa de diálogo.

4.2 Alterações de Arquitetura

Para resolver o problema de trava do mapa no momento de sua inicialização, a arquitetura geral do MOD foi alterada de forma que o servidor não enviasse todas as informações de lugares e eventos para o browser do cliente imediatamente. A nova arquitetura foi estruturada de forma que o servidor enviasse apenas os dados necessários para a demonstração dos marcadores e agrupamentos, a medida em que o usuário interage com o mapa.

Esta mudança de arquitetura possibilitou uma melhor divisão de processamento entre cliente e servidor. O cliente agora é responsável pelo processamento dos elementos visuais da página e pela realização de requisições por dados ao servidor, enquanto que o servidor é responsável pelas tarefas mais pesadas de processamento e retornar os dados solicitados pelo cliente. A próxima seção do trabalho detalha as técnicas e conceitos que tornaram possível a implementação da nova arquitetura.

4.2.1 Requisições assíncronas ao servidor

O mapa de lugares foi programado de forma que, após ser totalmente carregado no browser do cliente, realize requisições HTTP assíncronas a medida em que o usuário interage com o mapa. Esta abordagem tornou-se possível através da utilização de AJAX[5] (Asynchronous JavaScript and XML), que consiste em uma técnica de utilização de diversas tecnologias, como JavaScript e XML[6] (Extensible Markup Language), com o objetivo de executar chamadas HTTP ao servidor sem que a visualização e o comportamento da página web sejam afetados (NIEDERAUER, 2007).

O intercâmbio de informações entre cliente e servidor se dá através do objeto XMLHttpRequest, que é implementado pela maioria dos browsers da atualidade e serve como ponto central da técnica AJAX. A imagem 4.7 demonstra a arquitetura geral de uma aplicação que implementa AJAX, bem como o fluxo de informações que passam pelos elementos envolvidos no processo, desde o browser do cliente até a base de dados que alimenta a aplicação hospedada no servidor web.

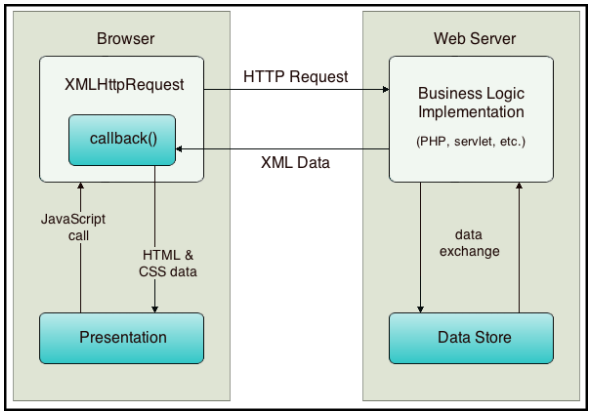

Figura 4.7: Arquitetura geral de aplicações AJAX

[4] http://jqueryui.com/dialog/
[5] http://www.w3schools.com/ajax/
[6] http://www.w3.org/XML/

Observando a imagem 4.7, podemos facilmente identificar que o nó de entrada do servidor web é aonde está hospedado o TNG e os scripts PHP que realizam as tarefas necessárias para implementação da nova arquitetura. A linguagem JavaScript desempenha um papel imprescindível em aplicações AJAX, visto que é ela que instancia o objeto XMLHttpRequest e faz o intermédio do resultado das requisições ao servidor com a camada de apresentação do browser.

No contexto deste trabalho, para facilitar a implementação da técnica AJAX, foi utilizada a biblioteca jQuery[7], que consiste em um framework JavaScript de fácil compreensão e utilização. Esta biblioteca possui uma rotina ajax que implementa o fluxo demonstrado na imagem 4.7 de forma fácil e intuitiva. O trecho de código abaixo exemplifica uma chamada AJAX utilizando jQuery.

```
$.ajax({
        type : 'post',
        data : 'funcao=buscarLugares',
        url  : 'index.php',
        success: function(object) {
          // code...
        }
);
```

Figura 4.8: Chamada AJAX utilizando jQuery

O atributo `type` define qual o comando HTTP a ser realizado pelo cliente (GET, POST, PUT ou DELETE). Este comando irá enviar os dados definidos no atributo `data` para a página definida no atributo `url`, página esta que está hospedada no servidor.

Quando o servidor recebe a requisição do cliente, o arquivo `index.php` é executado e faz uma leitura dos dados enviados, que no exemplo acima consiste na definição da variável `funcao` assumindo o valor `buscarLugares`. A partir dessas informações, o script PHP executa as regras de negócio implementadas, de acordo com os dados que recebeu, e envia uma resposta desta execução de volta ao cliente.

4.2.2 Arquivo JSON

No momento em que o browser recebe a resposta da chamada AJAX, a função definida no atributo `success` é executada, trazendo os dados solicitados pelo cliente dentro do parâmetro object. Apesar da nomenclatura da técnica de AJAX sugerir que o parâmetro deva estar no formato XML, seu uso não é obrigatório. Neste trabalho, foi utilizado o formato JSON[8] (JavaScript Object Notation), formato de intercâmbio de dados de fácil compreensão e de processamento leve.

Como a própria sigla sugere, o formato JSON foi escolhido pois ele possui notações sintáticas muito semelhantes à linguagem JavaScript. A imagem 4.9 mostra a implementação do analisador sintático das principais estruturas de um arquivo JSON.

[7] `http://jquery.com/`
[8] `http://www.json.org/`

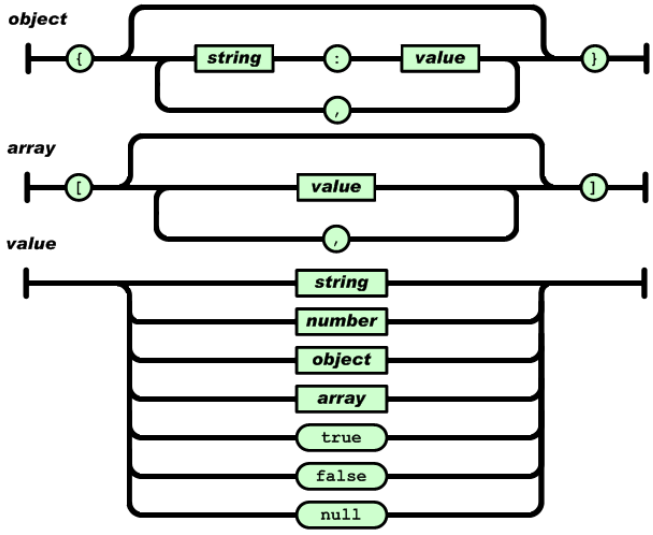

Figura 4.9: Analisador sintático de um arquivo JSON

Podemos ver através da imagem que as principais estruturas de um arquivo JSON são os os objects e os arrays, principalmente porque estas estruturas podem ser aninhadas dentro delas mesmas.

4.2.3 Rotina AJAX implementada

Para modificar o mapa de lugares de forma a utilizar as técnicas de AJAX, foi implementada, em JavaScript, a rotina `getClustersFromServer`, que realiza a requisição AJAX cada vez que o usuário interage com o mapa. Esta rotina faz uma leitura do estado atual da aplicação, envia essas informações para o servidor, processa o arquivo JSON retornado e atualiza o mapa com as informações dos agrupamentos e estatísticas.

O estado atual da aplicação consiste nos valores que as variáveis de controle do mapa assumem no momento da interação do usuário. Estas variáveis armazenam as informações dos filtros customizados e do estado atual do mapa. A tabela 4.2 lista as variáveis que são enviadas na requisição AJAX atravé do campo data, e as funções e variáveis JavaScript que permitem sua obtenção.

Tabela 4.2: Tabela de variáveis enviadas em uma requisição AJAX

Variável	Descrição	Origem	Obtenção
`zoom`	Nível de zoom do mapa	API GMaps	`map.getZoom()`
`minLat`	Latitude mínima do mapa	API GMaps	`map.getBounds().getSouthWest().lat()`
`maxLat`	Latitude máxima do mapa	API GMaps	`map.getBounds().getNorthEast().lat()`
`minLon`	Longitude mínima do mapa	API GMaps	`map.getBounds().getSouthWest().lon()`
`maxLon`	Longitude máxima do mapa	API GMaps	`map.getBounds().getNorthEast().lon()`
`events`	Eventos selecionados	Filtros	variável `events`
`trees`	Árvores selecionadas	Filtros	variável `selectedTrees`
`minYear`	Ano inicial selecionado	Filtros	variável `minYear`
`maxYear`	Ano final selecionado	Filtros	variável `maxYear`

Após a execução da rotina AJAX, estas variáveis são lidas pelo script PHP requisitado pela rotina. Este script realiza a busca das informações a serem demonstradas no mapa e retorna os dados no formato JSON. As rotinas implementadas neste script serão demonstradas com mais detalhes na próxima seção. A imagem 4.10 mostra o formato do arquivo JSON retornado ao browser do cliente.

```
{
    "markers": [
        {
            "size": 183,
            "lat": 45.943161,
            "lon": 24.96676,
            "maxLat": 45.943161,
            "minLat": 45.943161,
            "maxLon": 24.96676,
            "minLon": 24.96676,
            "places": ["Romania"]
        }
    ],
    "totals": {
        "0": 1724,
        "1": 196,
        "2": 478,
        "3": 733,
        "4": 100,
        "5": 62,
        "events": 3293,
        "places": 183
    }
}
```

Figura 4.10: Arquivo JSON retornado pela rotina AJAX

O atributo `markers` consiste em um array de objetos que representam os agrupamentos a serem demonstrados no mapa, com suas informações de lugares, posicionamentos e limites. Já o atributo `totals` consiste nas informações estatísticas dos eventos e lugares encontrados no banco de dados.

4.2.4 Biblioteca ClusterIcon

A partir do momento em que o browser do cliente recebe o arquivo JSON no formato descrito na imagem 4.10, o mapa então é preenchido com os dados contidos no arquivo. Para manter as mesmas características visuais e funcionais da versão anterior do MOD, foi feita uma adaptação da biblioteca `MarkerClusterer`, resultando em uma nova biblioteca JavaScript chamada `ClusterIcon`. Esta biblioteca tem o objetivo de ler os dados retornados pelo servidor e criar os agrupamentos com o mesmo comportamento visual e funcional da biblioteca `MarkerClusterer`.

Uma das tarefas da biblioteca `ClusterIcon` é a coloração de um agrupamento, que é realizada fazendo uma leitura da quantidade de lugares contidos em um agrupamento. Esta quantidade é enviada pelo servidor contida no atributo `size` do objeto JSON que representa um agrupamento. A imagem 4.11 mostra os tipos de imagens utilizadas nos agrupamentos, por ordem crescrente de densidade.

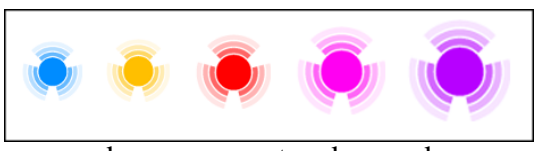

Figura 4.11: Imagens de agrupamentos de acordo com suas densidades

Outra tarefa realizada pela biblioteca `ClusterIcon` é o controle dos limites de um agrupamento. Estes limites, calculados em coordenadas geográficas, são utilizados no momento em que o usuário clique sobre um agrupamento, fazendo com que o mapa ajuste seu posicionamento e seu zoom de forma que todos os lugares do agrupamento estejam visíveis no mapa após o clique.

Na biblioteca `MarkerClusterer`, o cálculo destes limites é realizado no momento em que é executado o algoritmo de agrupamento. Na nova arquitetura, pelo fato deste algoritmo ser executado no servidor, os dados de limite de um agrupamento já chegam prontos no browser do cliente, através dos atributos `maxLat`, `minLat`, `maxLon` e `minLon`. Na próxima seção deste trabalho, é demonstrado como estes limites são calculados.

A biblioteca `ClusterIcon` também executa a tarefa de criação da caixa de diálogo que lista todos os lugares de um agrupamento. Como foi apresentado nas seções anteriores, esta caixa de diálogo é demonstrada apenas quando, ao clicar sobre um agrupamento, o usuário atinge o nível máximo de zoom permitido pelo mapa. Desta forma, a biblioteca utiliza o atributo `places`, que consiste em um array contendo o nome de todos os lugares que fazem parte do agrupamento, para criar a caixa de diálogo demonstrada na figura 4.6.

4.2.5 Otimizações de tráfego de rede e processamento

Para fins de otimização, tanto do tráfego de rede quanto do processamento do arquivo JSON pelo browser do cliente, o atributo `places` só é enviado nos casos em que se faz necessária sua utilização. Por exemplo, se um agrupamento possui 2.000 lugares e o usuário não está utilizando o mapa no seu nível de zoom máximo, a aplicação não necessita que o servidor envie a informação dos lugares, pois a caixa de diálogo de lugares não seria criada nesta circunstância. Porém, se o agrupamento conter apenas um lugar, este único lugar é enviado no atributo `places`, pois neste caso a aplicação criará um marcador que, ao ser clicado em cima, redirecionará o usuário para a tela nativa do TNG que mostra as informações detalhadas do lugar, fazendo-se necessário que o cliente tenha a string que descreve o lugar para realizar o redirecionamento.

Outra otimização realizada neste aspecto, é o fato de que o servidor envia o arquivo JSON ao cliente apenas contendo os agrupamentos que são visíveis no mapa de acordo com seu estado atual. Por exemplo, se o usuário está com o mapa focado exclusivamente sobre o continente europeu, qualquer agrupamento fora do campo de visão do mapa é desconsiderado na consulta ao banco de dados, visto que na nova arquitetura, apenas os agrupamentos contidos no campo de visão do mapa são renderizados e contabilizados nas estatísticas.

4.2.6 Interação que disparam a rotina AJAX

Como foi demonstrado nas seções anteriores, a rotina `getClustersFromServer` realiza um papel imprescindível para desempenhar corretamente as funções previstas na nova arquitetura da aplicação. Para que o mapa reagisse da forma esperada, foi necessário realizar um mapeamento de todas as interações de usuário que devem disparar a rotina. Desta forma, as informações do mapa são demonstradas fielmente de acordo com seu estado atual. As interações de usuário que disparam a rotina são:

- Inicialização do mapa
- Clique sobre um agrupamento
- Alteração do nível de zoom do mapa
- Alteração do posicionamento do mapa

- Alteração do tipo de mapa
- Alteração do filtro de árvores
- Alteração do filtro de eventos
- Alteração do filtro de datas
- Habilitar ou desabilitar filtro de datas

4.3 Algoritmo de Agrupamento

Conforme foi mencionado nas seções anteriores, parte das rotinas que antes eram executadas no lado do cliente foram implementadas no servidor, utilizando scripts PHP. Essas rotinas são executadas através da chamada AJAX ao arquivo `eventsMap_clustering.php`. A principal rotina executada é o algoritmo de agrupamento de lugares, cujos passos são detalhados nas próximas seções.

4.3.1 Estrutura e consulta ao banco de dados

A partir das variáveis enviadas pela requisição AJAX, demonstradas na tabela 4.2, a rotina deve fazer uma consulta ao banco de dados do TNG para obter os lugares georreferenciados que servirão como entrada do algoritmo de agrupamento implementado. A imagem 4.12 demonstra a estrutura das tabelas do banco de dados do TNG utilizadas na consulta pelos lugares onde ocorreram eventos.

Observando estas tabelas, podemos ver que os dados de latitude e longitude encontram-se apenas na tabela `tng_places`, e que não existe uma tabela específica que armazena os eventos de pessoas. As informações de data e local dos eventos estão representadas de forma dispersa nas colunas das tabelas `tng_people`, `tng_families` e `tng_events`. Para que seja possível realizar os filtros e estatísticas dos eventos do sistema, é necessário montar uma tabela de eventos de forma "artificial", através de operações de união de consultas que representem cada tipo de evento existente no sistema.

Outra informação importante é que as colunas que guardam os locais dos eventos não são chaves estrangeiras para a tabela `tng_places`. São campos do tipo text que guardam uma descrição textual do lugar, que correspondem ao mesmo valor da coluna place da tabela `tng_places`. Desta forma, para obtermos as coordenadas geográficas do lugar de um determinado evento, é necessário realizar uma junção da tabela `tng_places` com a tabela de eventos criada "artificialmente". A consulta SQL da imagem 4.13 exemplifica estes fatos. Nota-se que esta consulta já considera em seus filtros as variáveis enviadas pelo cliente na rotina AJAX, como os eventos e árvores selecionados e os limites de latitude e longitude do estado atual do mapa.

Como vimos na tabela 3.1, onde são listados os parâmetros do mapa, existe a possibilidade da consulta ao banco de dados não considerar os eventos. Neste caso, a consulta é realizada diretamente na tabela `tng_places`, sem a necessidade de realizar junções ou operações de união, o que torna a consulta muito mais eficiente. Porém, neste caso, os filtros por evento e por data não possuem mais utilidade.

Com o resultado destas consultas, o script PHP faz os cálculos estatísticos e monta um array de marcadores georreferenciados (com as informações de latitude e longitude) que serve como entrada do algoritmo de agrupamento implementado neste trabalho, que é apresentado na próxima seção.

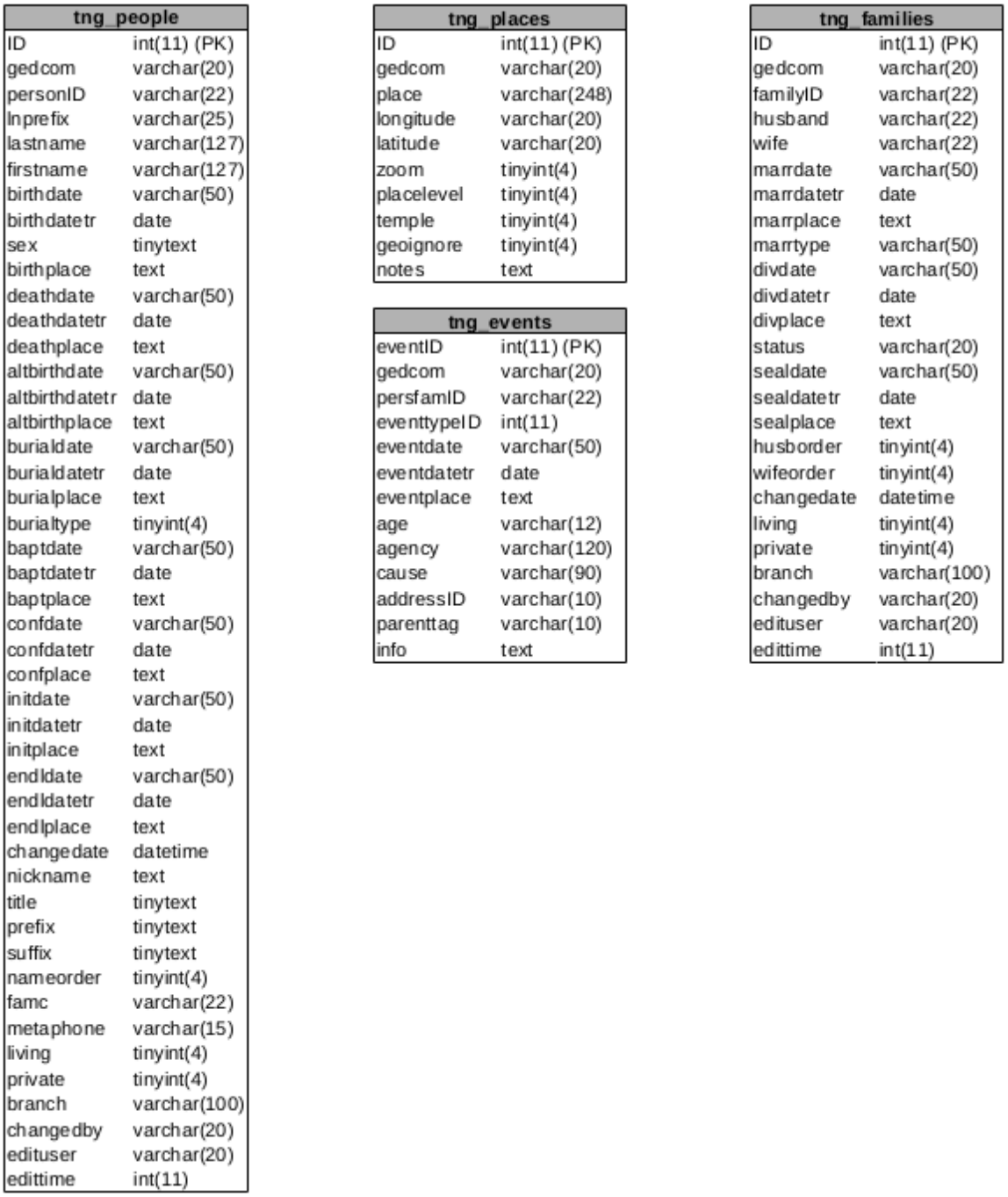

Figura 4.12: Tabelas do TNG utilizadas para consulta de dados

```
SELECT DISTINCT TPL.ID, TPL.gedcom, TPL.place,
                TPL.latitude, TPL.longitude, year, event
FROM (
  SELECT DISTINCT gedcom,
                  birthplace AS place,
                  SUBSTR(birthdatetr,1,4) as year,
                  0 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  deathplace AS place,
                  SUBSTR(deathdatetr,1,4) as year,
                  3 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  burialplace AS place,
                  SUBSTR(burialdatetr,1,4) as year,
                  4 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  altbirthplace AS place,
                  SUBSTR(altbirthdatetr,1,4) as year,
                  1 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  marrplace AS place,
                  SUBSTR(marrdatetr,1,4) as year,
                  2 as event FROM `tng\_families`
  UNION ALL
  SELECT DISTINCT gedcom,
                  baptplace AS place,
                  SUBSTR(baptdatetr,1,4) as year,
                  5 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  confplace AS place,
                  SUBSTR(confdatetr,1,4) as year,
                  5 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  initplace AS place,
                  SUBSTR(initdatetr,1,4) as year,
                  5 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  endlplace AS place,
                  SUBSTR(endldatetr,1,4) as year,
                  5 as event FROM `tng\_people`
  UNION ALL
  SELECT DISTINCT gedcom,
                  eventplace AS place,
                  SUBSTR(eventdatetr,1,4) as year,
                  5 as event FROM `tng\_events`
  UNION ALL
  SELECT DISTINCT gedcom,
                  divplace AS place,
                  SUBSTR(divdatetr,1,4) as year,
                  2 as event FROM `tng\_families`
) AS TPE, `tng\_places` AS TPL
WHERE TPE.place = TPL.place
AND TPL.gedcom = TPE.gedcom
AND latitude IS NOT NULL AND latitude <> ''
AND latitude  >= -64.38326667068164 AND latitude  <= 71.81128562798182
AND longitude >= -180 AND longitude <= 180
AND TPE.event IN (0, 1, 2, 3, 4, 5)
AND TPE.gedcom IN ('corfield', 'gedr8207', 'heusers')
```

Figura 4.13: Consulta SQL realizada para buscar lugares de eventos

4.3.2 Implementação do Algoritmo

Para realizar a tarefa de agrupar os marcadores a serem demonstrados no mapa, foi implementado o algoritmo "Estrela", ou "Stars" (ASLAM; PELEKHOV; RUS, 2006), que calcula a distância euclideana entre os marcadores e agrupa-os conforme sua proximidade. Esta escolha foi feita pelo fato de que, desta maneira, a implementação ficaria mais simples e coerente com o resultado esperado. O estudo da implementação de algoritmos mais eficientes fica como sugestão para estudo em trabalhos futuros. A implementação está descrita resumidamente na imagem 4.14 em português estruturado:

```
M = array de marcadores georreferenciados
R = Raio de de agrupamento
enquanto M não estiver vazio
    retira item P de M
    para i = 0 até i < tamanho(M)
        se  distancia (P, Mi) < R
            agrupar(P, Mi)
            remove Mi de M
        fim se
    fim para
fim enquanto
```

Figura 4.14: Pseudocódigo do algoritmo de agrupamento de marcadores

Como foi mencionado no capítulo anterior, a biblioteca `MarkerClusterer` implementa o mesmo algoritmo, mas com algumas particularidades. O espaço considerado por um agrupamento neste algoritmo é retangular, e não possui um parâmetro de distância que define a abrangência de um agrupamento. Desta forma, na nova versão do mapa, foi implementado um espaço circular de agrupamento, através da definição de um raio que estabelece uma distância máxima que dois marcadores devem ter para que se forme um agrupamento entre eles.

No que diz respeito ao cálculo da distância entre dois pontos, para que este cálculo seja preciso de acordo com a projeção de Mercator utilizada pela API do Google Maps, foi necessário realizar a conversão das coordenadas geodésicas para coordenadas cartesianas. Mais especificamente, os valores de latitude e longitude devem ser convertidos em valores nos eixos x e y no plano cartesiano, para que então o cálculo da distância possa ser feito com o resultado em pixels. A imagem 4.15 mostra uma comparação da projeção de Mercator com a esfera que representa o globo terrestre.

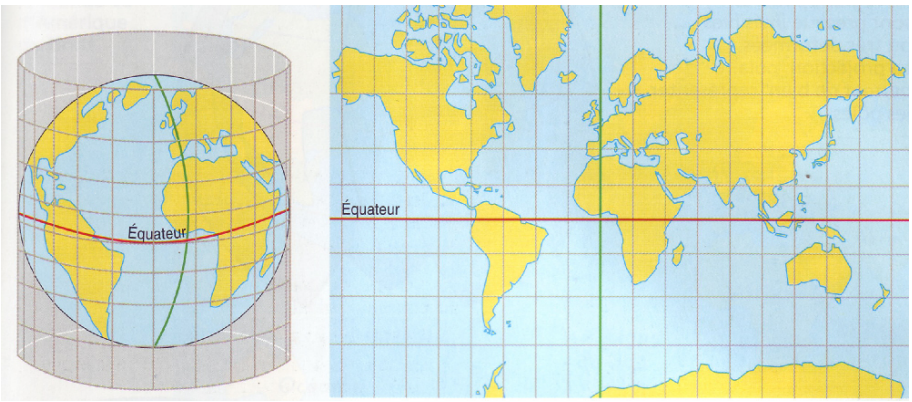

Figura 4.15: Projeção de Mercator em relação à esfera terrestre

Para entender como a conversão foi realizada, foi necessário partir das definições fundamentais de coordenadas geográficas (CAMARA; DAVIS; MONTEIRO, 2001). Latitude consiste no ângulo formado entre a linha do Equador e um ponto qualquer na Terra em relação ao ponto central da esfera terrestre. Longitude consiste no ângulo formado entre o meridiano de Greenwich e um ponto qualquer na Terra em relação ao ponto central da esfera terrestre. A imagem abaixo ilustra de forma mais prática o cálculo de latitude e longitude, onde o ângulo ϕ representa a latitude e o ângulo λ representa a longitude.

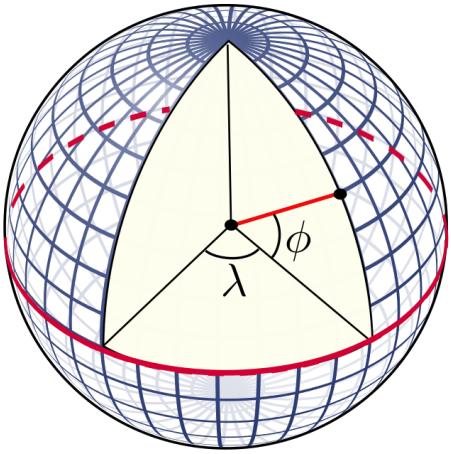

Figura 4.16: Representação de latitude e longitude no globo terrestre

Como as linhas do Equador e Greenwich servem como base para definir as coordenadas geográficas, o ponto de intersecção entre essas duas linhas servirá de ponto de partida para realizar esta conversão. Desta forma, foi feita uma análise da API do Google Maps para compreender o processo de renderização do mapa, com o objetivo de achar uma representação em pixels do ponto que corresponde a intersecção das linhas Equador e Greenwich.

A imagem do mapa renderizado consiste em um conjunto de pequenas imagens chamadas tiles[9], que são demonstradas de acordo com o posicionamento e nível de zoom do mapa. A imagem 4.17 exemplifica uma divisão do mapa no nível de zoom 2.

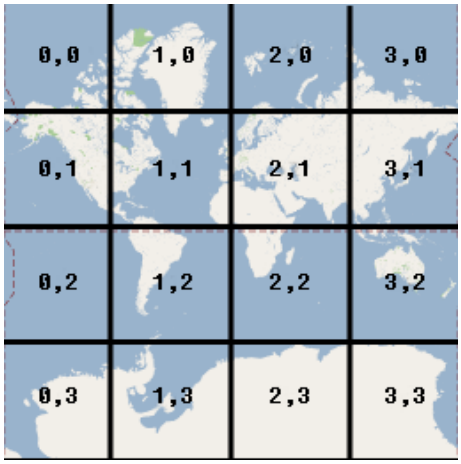

Figura 4.17: Divisão do Google Maps em Tiles

Cada tile possui uma resolução de 256x256 pixels. Na imagem 4.17, o mapa inteiro é

[9] https://developers.google.com/maps/documentation/javascript/maptypes

composto por 16 tiles, totalizando uma imagem de resolução 1024x1024 pixels. A cada aumento de nível de zoom, o numero de tiles dobra em relação a cada dimensão x e y, resultando em um mapa com uma resolução quatro vezes maior que o nível de zoom anterior. No exemplo da imagem 4.17, ao aumentar o nível de zoom de 2 para 3, o mapa passaria a ser composto por 64 tiles, totalizando uma imagem de 2048x2048 pixels.

Com esta lógica, podemos calcular as dimensões máximas da imagem do mapa, que corresponde ao mapa em seu nível de zoom máximo. Na tabela 4.1, vimos que o nível máximo de zoom que o mapa pode assumir é 21, considerando todos os tipos de mapa que a API do Google Maps permite. Desta forma, a imagem de maior tamanho possível a ser renderizada no mapa possui uma resolução de 536870912x536870912 pixels, tendo o ponto de intersecção das linhas do Equador e Greenwich localizados no ponto (268435456,268435456) da imagem.

Tendo o ponto de referência em pixels para realizar a conversão, foi implementada a fórmula matemática que realiza a conversão de latitude e longitude para x e y. O código da imagem 4.18 demonstra a implementação realizada neste trabalho, com a constante OFFSET representando a posição central do mapa em pixels.

```
define('OFFSET', 268435456);
define('RADIUS', 85445659.4471); /* $offset / pi() */

function lonToX($lon) {
    return round(OFFSET + RADIUS * $lon * pi() / 180);
}

function latToY($lat) {
    return round(OFFSET - RADIUS *
                log((1 + sin($lat * pi() / 180)) /
                (1 - sin($lat * pi() / 180))) / 2);
}
```

Figura 4.18: Implementação da conversão de tipos de coordenadas

Assim, para calcularmos a distância entre dois pontos em coordenadas cartesianas, podemos utilizar o Teorema de Pitágoras, conforme a implementação demonstrada na imagem 4.19.

```
function pixelDistance($lat1, $lon1, $lat2, $lon2, $zoom) {
    $x1 = lonToX($lon1);
    $y1 = latToY($lat1);

    $x2 = lonToX($lon2);
    $y2 = latToY($lat2);

    return sqrt(pow(($x1-$x2),2) + pow(($y1-$y2),2)) >> (21 - $zoom);
}
```

Figura 4.19: Implementação do cálculo da distância entre dois pontos

O resultado encontrado no teorema de Pitágoras considera sempre o mapa em sua resolução máxima em pixels. Por isso é importante destacar a operação de shift right realizada na linha 8 do código. Esta operação é realizada para converter a distância encontrada em relação ao nível de zoom atual do mapa. Como o mecanismo de tiles utilizado pelo Google Maps modifica as dimensões da imagem do mapa em potências de 2, basta aplicar um número de shifts equivalente à diferença entre o nível máximo e o nível atual do mapa.

Para dar mais flexibilidade ao uso do mapa, foi criado um novo parâmetro para o MOD, que define a distância máxima permitida para que um marcador faça parte de um agrupamento. Esta distância é definida em pixels, e a imagem abaixo mostra o reflexo da alteração deste parâmetro na criação dos agrupamentos no mapa. À esquerda, o parâmetro foi configurado com 25 pixels, e à direita, com 50 pixels.

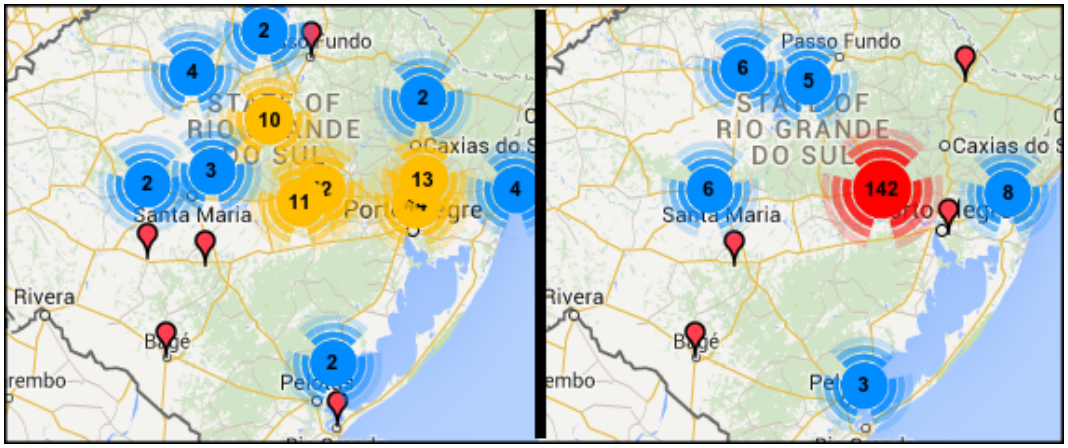

Figura 4.20: Modificação na distância máxima de agrupamento

Em relação ao posicionamento dos agrupamentos no mapa, este é realizado em seus pontos centrais, que são calculados em relação aos valores dos atributos `maxLat`, `minLat`, `maxLon` e `minLon`. Pelo fato de sua implementação ser mais viável no contexto deste trabalho, optou-se pelo posicionamento central, mas existe a possibilidade de calcular o posicionamento considerando a concentração de marcadores em um determinado lugar, fazendo um cálculo da média ponderada dos posicionamentos dos marcadores de um agrupamento.

Após realizar todos os passos vistos acima, o script PHP faz a montagem do arquivo JSON a ser enviado ao cliente, conforme o formato visto nas seções anteriores. Desta forma, é finalizada a rotina executada pelo servidor quando é realizada a requisição AJAX pelo browser do cliente.

4.4 Resultados Obtidos

4.4.1 Análise de Desempenho

Com o objetivo mensurar o desempenho da rotina AJAX implementada, foi realizada uma análise de desempenho do tempo de execução das rotinas implementadas. Essa análise foi feita em cima de 4 bases de dados de tamanhos diferentes, simulando 4 cenários distintos em cada base. Estes cenários consistem na modificação dos itens listados abaixo:

- Parâmetro “Mostrar Eventos” (ativado ou desativado)
- Posiçãocoordenadas geográficas (terra firme ou por todo o mapa)

O primeiro item como objetivo analisar o impacto da consulta ao banco de dados no tempo total da rotina AJAX. Já o segundo item, o objetivo é analisar o impacto do algoritmo de agrupamento no tempo total de execução da rotina, mostrando que, se os marcadores estiverem mais próximos, o desempenho do algoritmo é melhor. Neste caso, as coordenadas geográficas foram geradas aleatoriamente através de manipulação direta no banco de dados. A tabela 4.3 mostra as bases de dados utilizadas na análise.

Tabela 4.3: Tabela de bases de dados utilizadas na análise de desempenho

Base	Lugares	Eventos
tng1/tng1f	634	3548
tng2/tng2f	8069	24962
tng3/tng3f	28869	60926
tng4/tng4f	64240	137485

As bases “tng1”, “tng2”, “tng3” e “tng4” consistem nas bases com coordenadas geográficas geradas em terra firme, enquanto que as bases “tng1f”, “tng2f”, “tng3f” e “tng4f” consistem nas bases com coordenadas geográficas geradas no mapa inteiro.

Para simular a execução da rotina AJAX no browser, foi utilizada a aplicação cURL[10], realizando requisições HTTP ao script `eventsMap_clustering.php` com alterações nos parâmetros de envio para simular os cenários desejados na análise. Para calcular o tempo de execução das rotinas, foi utilizada a função `microtime()` do PHP. A figura 4.21 mostra um exemplo de como foi utilizada esta rotina, obtendo o tempo de execução total da rotina no final.

```
$time_start = microtime(true);

// rotina...

$time_end = microtime(true);
$execution_time = ($time_end - $time_start);
```

Figura 4.21: Exemplo de uso da função microtime do PHP

Cada requisição HTTP realizada tem como objetivo simular a inicialização do mapa de lugares. Ou seja, é considerado que todos os eventos e árvores estejam selecionados, que a barra de rolagem de anos está desabilitada e que todos os lugares georreferenciados contidos no banco de dados sejam selecionados. Para obter resultados mais aproximados, foram realizadas 5 requisições sobre cada cenário distinto, calculando-se a média aritmética dos tempos obtidos.

A especificação do ambiente onde foi realizada a análise de desempenho está na listagem abaixo:

- Sistema Operacional: Linux Ubuntu 14.04 LTS (64 bits)
- Processador: Intel I5
- Memória RAM: 4GB
- Servidor Web Apache 2.4.7 (executando localmente)
- PHP 5.5.9
- MySQL 5.5.40 (hospedado localmente)
- Todos os mecanismos de cache desabilitados

O gráfico da imagem 4.22 mostra os resultados obtidos com o parâmetro “Mostrar Eventos” habilitado, enquanto que a imagem 4.23 mostra os resultados obtidos com o parâmetro desativado.

[10] `http://curl.haxx.se/`

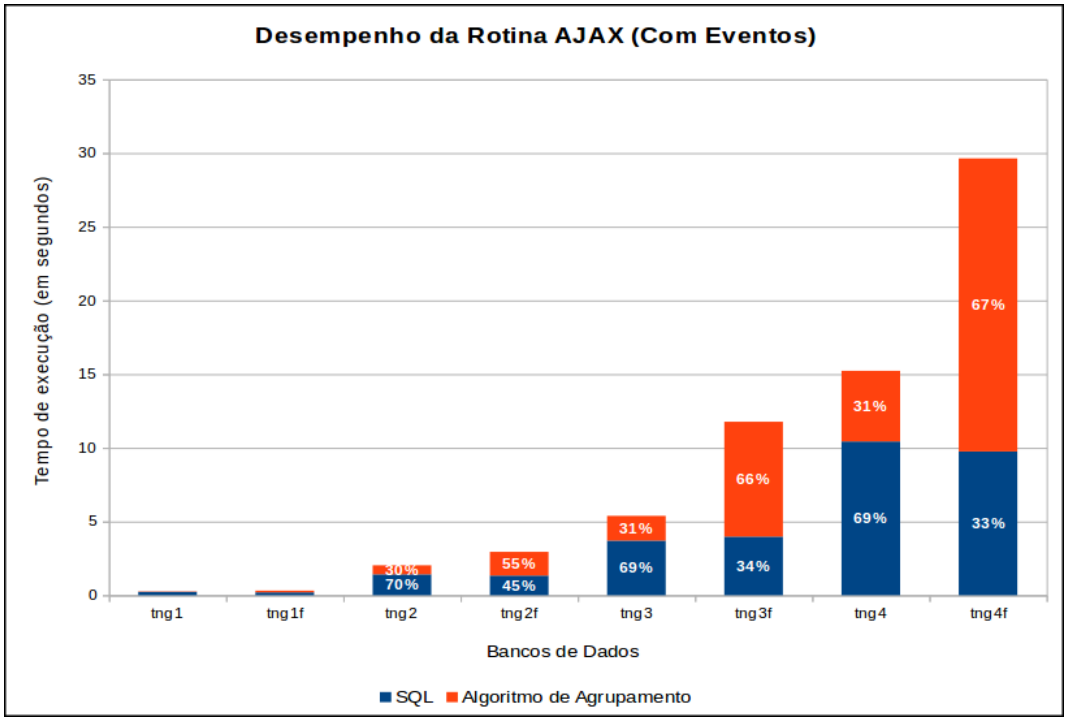

Figura 4.22: Análise de desempenho da rotina AJAX com eventos

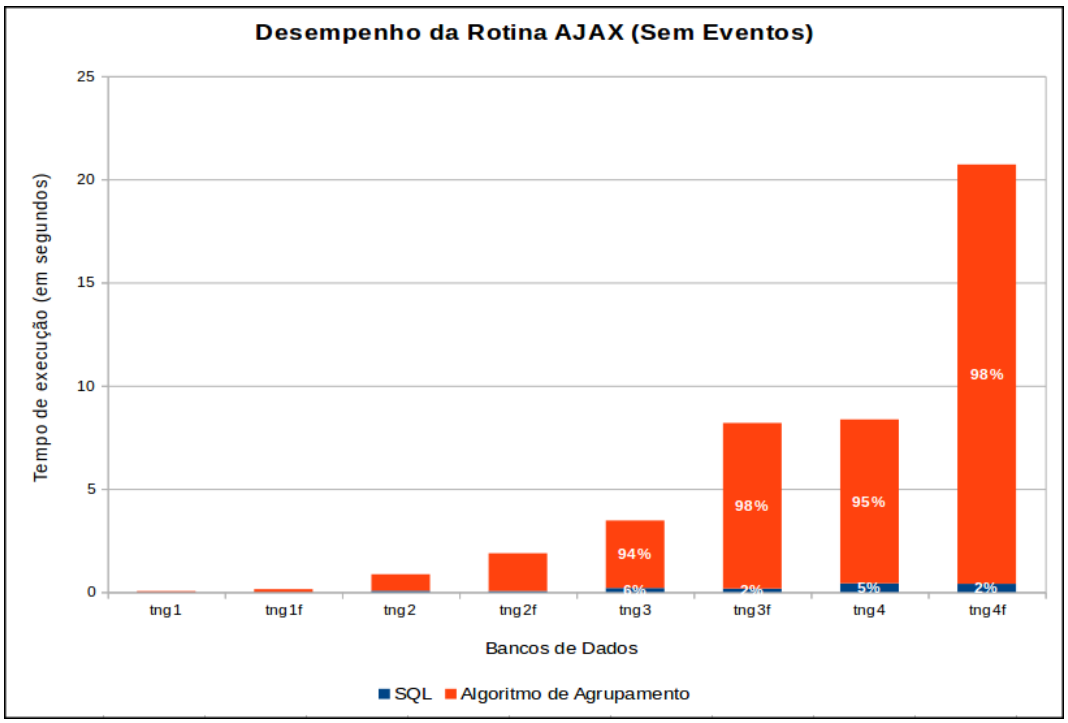

Figura 4.23: Análise de desempenho da rotina AJAX sem eventos

No gráfico 4.22, podemos notar nas partes em azul, que de fato a consulta ao banco de dados se torna um gargalo quando o parâmetro de eventos está ativado, podendo chegar a um tempo de execução de cerca de 10 segundos por requisição AJAX nos piores casos. Isto acaba por comprometer o uso do mapa de forma rápida quando o parâmetro está ativo. Já no gráfico 4.23, podemos ver que a consulta ao banco de dados consome uma porção muito pequena do tempo total de execução da rotina.

No que diz respeito ao algoritmo de agrupamento, podemos ver em ambos os gráficos que a localização dos marcadores (terra firme ou mapa inteiro) influencia bastante no desempenho da rotina. Porém, o desempenho do algoritmo está longe de ser o ideal para bases de dados muito grandes. No pior caso, o algoritmo de agrupamento sozinho demora cerca de 20 segundos para executar.

5 CONCLUSÃO

O objetivo principal deste trabalho foi realizar a mudança na arquitetura do mapa de lugares de eventos, que foi desenvolvido como um MOD do sistema de gerência de árvores genealógicas TNG. Na nova arquitetura, rotinas de processamento pesado que antes eram implementadas no lado do cliente, agora são implementadas no lado do servidor.

A partir de testes práticos realizados, foi constatado que o algoritmo de agrupamento Euclideano implementado nos scripts PHP resolveu o problema de travamento na inicialização do mapa. A base de dados do TNG utilizada para realizar esses testes possuía entre 4.000 e 5.000 lugares.

Porém, ao analisar a estrutura do banco de dados implementada pelo TNG, vimos que a busca por informações de eventos de pessoas se torna uma tarefa custosa, principalmente em bases de dados de grande porte. O fato de não existir uma tabela de eventos específica no sistema e o fato das colunas de lugares de eventos não possuírem chaves estrangeiras para a tabela de lugares torna as consultas bastante devagar a cada requisição AJAX realizada ao servidor. Uma reestruturação do banco de dados visando melhorar a performance das consultas resolveria este problema, porém, como o TNG é um software comercial, qualquer alteração na estrutura do banco de dados do sistema está fora de escopo. Assim, no que diz respeito à contornar este problema de performance do banco de dados, a única alternativa existente continua sendo a desativação do tratamento de eventos através do respectivo parâmetro do MOD.

No que diz respeito ao algoritmo de agrupamento implementado neste trabalho, optou-se por utilizar a solução mais simples para a resolução do problema. Esta solução simples desconsidera qualquer otimização de performance. Uma melhoria deste algoritmo que pode ser realizada em um trabalho futuro, é a implementação de uma estrutura hierárquica de dados, como Árvores R, para armazenar os dados georreferenciados resultantes da consulta ao banco de dados do TNG. Com estas estruturas, a comparação da distância entre lugares seria realizada somente entre outros lugares com forte potencial de fazerem parte do agrupamento, tornando o algoritmo mais eficiente e apto a suportar bases de dados ainda maiores.

Outra melhoria que pode ser explorada em trabalhos futuros é a respeito da precisão do posicionamento de um agrupamento, realizando o cálculo considerando uma distribuição ponderada dos lugares do agrupamento. Se o parâmetro de distância máxima for configurado com valores muito altos, os erros de precisão deste posicionamento se tornam mais evidentes. Como mencionado no trabalho, a implementação do algoritmo K-Means por completo pode vir a resolver (ou melhorar) este problema de precisão.

As melhorias de usabilidade e de estética realizadas no mapa, em conjunto com a alteração da arquitetura do mapa, atenderam bem as expectativas. Neste âmbito, podem ser exploradas outras melhorias visuais, como implementar um mecanismo de distinção

visual no mapa de cada árvore genealógica selecionada nos filtros, ou permitir a filtragem de eventos customizados a partir do campo de filtros.

REFERÊNCIAS

ASLAM, J.; PELEKHOV, E.; RUS, D. **Grouping Multidimensional Data**. [S.l.]: Springer Berlin Heidelberg, 2006.

CAMARA, G.; DAVIS, C.; MONTEIRO, A. M. V. **Introdução à Ciência da Geoinformação**. São José dos Campos, SP, Brasil: INPE, 2001.

GOOGLE. **Google Maps Javascript API V3 Documentation**. Acesso em novembro de 2014, Disponível em: `https://developers.google.com/maps/documentation/javascript/`.

NIEDERAUER, J. **Web Interativa com AJAX e PHP**. Porto Alegre, RS, Brasil: Novatec, 2007.

SILVEIRA, F. M. da. **Navegação Georreferenciada de uma Base de Dados de Árvores Genealógicas**. Junho de 2013.

TNG. **The Next Generation of Genealogy Sitebuilding©**. Acesso em novembro de 2014, Disponível em: `http://www.tngsitebuilding.com/`.