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

VINICIUS ROSA DOS SANTOS

Sistema de busca e exibição de dados georreferenciados

Trabalho de Graduação.

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, novembro de 2009.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
UDF	*User Defined Function*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
DBMS	*Database Management System*
JSON	*JavaScript Object Notation*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um dos maiores problemas encontrados por historiadores e geólogos ao trabalharem com informações históricas relativas à nossa colonização, especialmente informações geográficas e referências genealógicas, é como publicar tais dados. As ferramentas de software de georreferenciamento e geoprocessamento disponíveis no mercado carecem de uma forma simplificada e acessível de visualização e pesquisa das informações coletadas. Uma alternativa é combinar serviços e técnicas variadas em uma só solução, provendo ao usuário final uma experiência satisfatória na interação com os dados coloniais. Neste trabalho é apresentado um sistema que mescla as funcionalidades de um serviço Web de mapeamento geográfico com técnicas de busca de informações por similaridade, com o objetivo de publicar informações coloniais do Estado do Rio Grande do Sul. Desta forma, obtém-se uma fácil visualização das regiões coloniais em um mapa e uma alta flexibilidade na pesquisa dos dados cadastrados, uma vez que informações históricas são sensíveis a erros tipográficos e possuem grafias alternativas para nomes de lugares e pessoas. Além disso, a solução proposta é acessível globalmente através da Internet, podendo ser consultada por qualquer interessado.

Palavras-Chave: georreferenciamento, busca por similaridade, geoprocessamento, mapeamento

Georeferenced data search and display system

ABSTRACT

One of the main problems encountered by historians and geologists when they work with historical information related to our colonization, especially geographic information and genealogical references, is how to publish such data. The available commercial software tools for georeferencing and geoprocessing lack of a simplified and accessible way of visualization and search of the collected information. An alternative is to combine various services and techniques in a single solution, providing the end user a satisfying experience in interacting with the colonial data. This paper presents a system that combines the functionality of a web mapping service application with similarity search techniques in order to publish colonial information about Rio Grande do Sul State. Thus, easy visualization of the colonial regions on a map is obtained. Besides that, a high flexibility in the search of the registered data is acquired, as historical information are sensitive to typographical errors and have alternative spellings for places and people names. Moreover, the proposed solution is available globally via the Internet and can be consulted by anyone interested.

Keywords: georeferencing, similarity search, geoprocessing, mapping

1 INTRODUÇÃO

Muitos historiadores, geólogos e geógrafos, principalmente do Rio Grande do Sul, têm um interesse especial sobre informações históricas referentes à colonização do Estado. Fontes de dados sobre a divisão geográfica das colônias e sobre seus proprietários (possivelmente por um interesse na genealogia de tais colonos) mostram-se abundantes nos mais variados formatos, desde documentos oficiais antigos a simples relatos passados de geração em geração. Entretanto, o problema reside, principalmente, no momento dessas informações serem catalogadas, unificadas e exibidas em um meio comum que seja de fácil visualização e de acesso global. Existem vários aplicativos que auxiliam no georreferenciamento dos dados, ou seja, na transcrição das informações de uma imagem, por exemplo, para coordenadas geográficas. O mais famoso deles é o ArcGIS (http://www.esri.com/software/arcgis/), uma suíte de aplicações com vários utilitários voltados exclusivamente ao processamento de dados geográficos. O software também auxilia na catalogação de qualquer tipo de informação agregada. Contudo, além de ser um aplicativo pago, não oferece uma interface acessível a qualquer pessoa interessada na visualização dos dados. Também não resolve o problema de unificação das informações, uma vez que, já que elas vêm de diversas fontes, diferentes grafias de nomes de lugares e pessoas podem ser utilizadas. O ideal seria uma ferramenta disponível através da Web, com acesso livre a todos e que disponibilizasse uma forma flexível de consulta aos dados.

Felizmente, a Internet oferece alguns serviços gratuitos que podem auxiliar na visualização de dados geográficos através de mapas. Talvez o mais famoso deles seja o Google Maps (GOOGLE, 2009), o qual, através de seus mapas e imagens de satélite, permite que qualquer dado geográfico espacial seja exibido como se estivesse em um mapa físico. Além disso, seu atlas compreende todo o território da Terra e com níveis de detalhe impressionantes no Rio Grande do Sul, permitindo uma análise bastante apurada das informações geográficas que exibe.

Para a resolução do problema referente à unificação dos dados provindos de fontes diferentes, existem técnicas de busca de informações por similaridade. Através delas, a informação pesquisada pelo usuário é comparada por aproximação aos dados cadastrados. Com tal flexibilidade, não existe a necessidade do usuário saber a grafia exata de um determinado nome ou lugar, bastando que o mesmo informe algo “parecido”.

A solução proposta adota o Google Maps como ferramenta de visualização dos dados geográficos espaciais e a técnica apresentada em (GRAVANO, 2005) para a implementação da busca de informações por similaridade. Os dados cadastrados referem-se à colonização italiana e alemã no Rio Grande do Sul no final do século XIX. Todas as informações foram cedidas por Otavio Augusto Boni Licht, geólogo, mestre e

doutor, da L&S Consultoria Geológica Ltda, de Curitiba, PR. Os dados foram georreferenciados através do software ArcGIS e, para serem utilizados pela aplicação, foram convertidos de forma que pudessem popular um banco de dados relacional.

O presente documento está organizado conforme o que se segue. O capítulo 2 apresenta uma discussão detalhada a respeito do Google Maps e como o mesmo foi utilizado na solução proposta, além de apresentar o GeoNames, outro serviço Web que fornece algumas informações adicionais sobre dados georreferenciados. O capítulo 3 explica a técnica de busca por similaridade utilizada, mostrando exemplos conceituais e sua aplicação na solução. O capítulo 4 mostra as funcionalidades do sistema, isto é, o que ele oferece ao usuário. Finalmente, o capítulo 5 demonstra como as funcionalidades da aplicação foram implementadas através da arquitetura geral do sistema.

2 SERVIÇOS DE EXIBIÇÃO DE DADOS GEORREFENCIADOS

Dispondo de dados georreferenciados referentes às colônias italianas e alemãs do Rio Grande do Sul, é necessário um meio para exibi-los em um mapa que seja acessível globalmente através da Internet. Além disso, também seria interessante mostrar algumas informações adicionais, obtidas através de geoprocessamento, sobre os lotes coloniais. Para tanto, a aplicação faz uso do Google Maps, um serviço Web de mapeamento, o qual tem todas as ferramentas necessárias para uma detalhada visualização dos dados, e do GeoNames, outro serviço Web que mantém uma base de dados geográfica e que fornece vários tipos de informação a partir de determinadas coordenadas geográficas. Uma explicação sobre ambos os serviços e como eles são utilizados pela aplicação será apresentada durante o decorrer do capítulo.

2.1 Google Maps

O Google Maps (GOOGLE, 2009) é um serviço Web de pesquisa e visualização de mapas e imagens de satélite, fornecido e desenvolvido pela empresa Google. Oferece mapas de ruas, planejamento de rotas a pé, a carro ou por transporte público, além de vários outros serviços. Uma vez que é totalmente acessível através da Internet, torna-se um meio formidável para a publicação de dados de interesse público. A Figura 2.1 mostra a interface inicial do site.

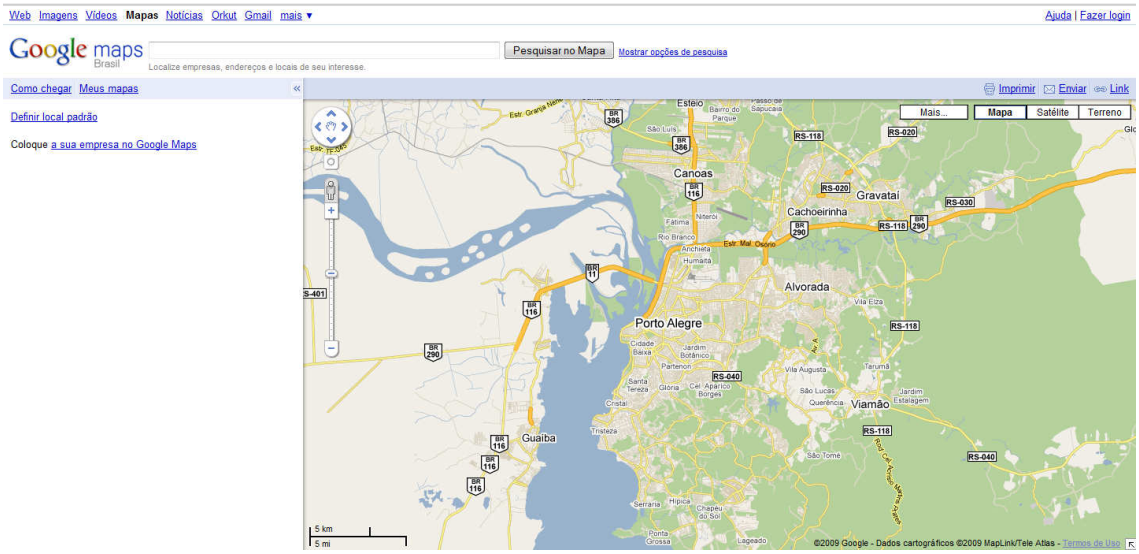

Figura 2.1: Interface do Google Maps

O serviço também oferece, de forma gratuita para uso não-comercial, uma API (*Application Programming Interface*) acessível através da linguagem de programação JavaScript, possibilitando que as funcionalidades do Google Maps sejam incorporadas a qualquer página Web. Dessa forma, os mapas podem ser manipulados a partir de vários utilitários disponíveis pela API. Dentre tais serviços, destacam-se a criação de rotas a partir de pontos de início e fim (da mesma forma já existente no site), a geocodificação direta (obtenção das coordenadas geográficas de um dado endereço) e a reversa (obtenção do endereço de uma rua através de suas coordenadas geográficas). Por causa da farta documentação disponível sobre sua API, ampla abrangência de localidades e por sua popularidade na Web, o Google Maps foi o escolhido para ser utilizado na aplicação. Outros serviços de mapeamento também foram cogitados, tais como Bing Maps (BING, 2009) e Yahoo Maps (YAHOO, 2009), mas foram descartados em detrimento do serviço da Google. Apesar das inúmeras funcionalidades disponíveis, a aplicação utiliza apenas aquelas necessárias à exibição e interação com os lotes coloniais no mapa e outras questões relacionadas a *zoom* de uma determinada região pelo usuário. Tais funcionalidades e suas respectivas estruturas de dados serão explicadas a seguir.

2.1.1 Polígonos

A API do Google Maps oferece a possibilidade do desenho de polígonos de qualquer formato no mapa. Para tanto, basta que sejam informadas as coordenadas geográficas (latitude e longitude) dos vértices do polígono. Essas formas geométricas são utilizadas na aplicação para representar um determinado lote colonial no mapa. Algumas opções de customização também são oferecidas, tais como cor e opacidade do preenchimento e da borda do polígono. Contudo, a forma geométrica não é apenas um mero desenho no mapa: é uma estrutura de dados que contém propriedades que podem ser lidas e definidas em tempo de execução, além de fornecer uma série de eventos disparados na interação com o usuário. O único evento utilizado na aplicação é o que ocorre quando o usuário clica sobre um determinado polígono. Quando ele é disparado, todas as informações disponíveis referentes ao lote clicado são buscadas no banco de dados e apresentadas em um balão de informações situado sobre o polígono, funcionalidade esta que também é fornecida pela API. Apesar de oferecer outras funcionalidades, tais como alteração dinâmica da forma de um polígono ou até mesmo a possibilidade do usuário construir sua própria forma geométrica, a solução fez uso apenas de um subconjunto básico de operações.

2.1.2 Marcadores

Os marcadores, como o próprio nome já diz, marcam no mapa uma determinada coordenada geográfica definida pelo usuário através da API. De forma semelhante aos polígonos, basta que sejam informadas a latitude e longitude do ponto para que tal localização seja visualizada no mapa através de um ícone. A aplicação utiliza marcadores para apontar a localização do lote de um determinado proprietário. A API possibilita que o ícone do marcador seja totalmente customizado, podendo ser utilizada qualquer imagem definida pelo usuário. Os marcadores fornecem ainda mais funcionalidades do que as oferecidas pelos polígonos, com mais propriedades e eventos disponíveis. Na aplicação, é usado o evento de clique sobre o marcador e, assim como acontece com os polígonos, as informações referentes ao lote apontado são buscadas no banco de dados e apresentadas no mesmo balão de informações mencionado

anteriormente. Além disso, o evento de clique com o botão direito do mouse também é tratado, pois, dessa forma, é possível que o lote seja ocultado do mapa.

2.1.3 Limites geográficos

Um limite geográfico para a API do Google Maps é um retângulo não-visual em coordenadas geográficas. Na prática, necessita de apenas duas informações: a coordenada sudoeste (canto esquerdo inferior do retângulo) e a coordenada nordeste (canto direito superior do retângulo). Através de um limite, tem-se conhecimento da *viewport* do usuário no mapa, ou seja, a área de visualização mostrada atualmente. Na aplicação, é construído um limite para cada região visualizada pelo usuário, seja ela uma colônia ou uma linha colonial. Dessa forma, é oferecida ao usuário a possibilidade do mesmo realizar um *zoom*, a qualquer momento, sobre qualquer região selecionada.

2.1.4 Coordenadas geográficas

A API do Google Maps não trabalha apenas com valores em ponto flutuante de latitude e longitude em suas coordenadas geográficas. Assim como os demais itens apresentados, uma coordenada é uma estrutura de dados rica, com funcionalidades que vão desde conversão de valores para radianos até o cálculo da distância entre duas dadas coordenadas. Contudo, o uso de tais operadores na aplicação não se mostrou necessário.

2.2 GeoNames

O GeoNames (GEONAMES, 2009) é uma base de dados geográfica, com acesso totalmente gratuito e sob uma licença da Creative Commons Attribuition, a qual é acessível através de vários web services. Contém mais de oito milhões de nomes geográficos e consiste em mais de seis milhões e meio de recursos exclusivos, dos quais mais de dois milhões são nomes de lugares povoados e outros quase dois milhões referem-se a nomes alternativos. Os dados armazenados no GeoNames podem ser editados na Web por todos os interessados. Além disso, o serviço conta com embaixadores em vários países, os quais têm como função auxiliar na localização de fontes confiáveis de dados e em questões sobre a divisão administrativa de cada nação.

Cada web service disponível pelo GeoNames provê um tipo de informação geográfica diferente (ou o mesmo tipo de informação de fontes distintas). A aplicação utiliza dois desses serviços: dados da cidade mais próxima e elevação de um determinado lote.

A obtenção dos dados da cidade mais próxima se dá através do endereço http://ws.geonames.org/findNearbyPlaceNameJSON?lat=xx.xx&lng=yy.yy, onde xx.xx é o valor de latitude e yy.yy é o valor de longitude do local pesquisado. O retorno da requisição é um objeto estruturado com várias informações a respeito da localidade mais próxima do ponto de pesquisa, tais como nome da localidade, seu estado, país, titulação (cidade, vila, etc), fuso horário, entre outros. Na aplicação, apenas o nome da cidade é utilizado.

O banco de dados já está populado com a informação de cidade de todos os lotes, ou seja, nenhuma requisição é realizada ao GeoNames durante a execução da aplicação. No pré-processamento dos dados, quando os mesmos estão sendo preparados para serem inseridos no banco, a API do Google Maps é utilizada para se obter as coordenadas

centrais (latitude e longitude) de cada lote. Então, é realizada uma requisição ao serviço para cada lote, passando-se os valores da coordenada obtida de cada um deles.

A elevação do lote (ou seja, sua altura em relação ao nível do mar) é obtida através do endereço http://ws.geonames.org/srtm3?lat=xx.xx&lng=yy.yy, onde xx.xx é o valor de latitude e yy.yy é o valor de longitude da posição pesquisada. A pesquisa é baseada nos dados do SRTM (*Shuttle Radar Topography Mission***)**, um sistema de radar especialmente modificado que voou juntamente com o *Space Shuttle Endeavour* (uma nave espacial da NASA) durante uma missão de onze dias em fevereiro de 2000. A área de amostra dos dados é de $90m^2$, isto é, a cada $90m^2$ tem-se um valor de elevação distinto. O retorno da requisição é um simples número informando a elevação, em metros, da posição pesquisada.

Assim como no caso do nome da cidade mais próxima, os dados de elevação dos lotes são obtidos antes do banco de dados ser populado. Entretanto, não é realizada apenas uma requisição por lote. Primeiramente, é pesquisada a elevação de cada coordenada do lote e cada valor é armazenado em uma lista própria do lote. Depois, novamente através da API do Google Maps, as coordenadas centrais do lote são obtidas e sua elevação é pesquisada, a qual também é inserida na lista. Por fim, é calculada a média aritmética dos valores constantes na lista, tendo-se, desta forma, a elevação média aproximada do lote.

3 TÉCNICAS DE BUSCA DE DADOS POR SIMILARIDADE

A manipulação de dados históricos provindos de várias fontes de informação diferentes, como é o caso da aplicação, leva, invariavelmente, a muitos erros tipográficos em nomes de pessoas e lugares. Além disso, existem certos termos que são escritos praticamente da mesma forma, com a mudança de apenas uma ou duas letras. Como exemplo, podemos citar o sobrenome “Schmidt”, o qual, muitas vezes, é escrito como “Schmit”, “Schmitt” ou “Shmidt”. Tanto no caso de erros quanto no caso de nomes praticamente iguais, qualquer pesquisa que utilize uma comparação de nomes exata pode levar a um resultado não tão relevante quanto o esperado. Afinal, o usuário muitas vezes não sabe exatamente a grafia de um determinado nome. Portanto, nas pesquisas por dados históricos, deve ser utilizado um mecanismo que tenha tal flexibilidade a ponto de retornar resultados aproximados (até certo ponto) ao pesquisado.

Para resolver tais questões, a solução proposta adota busca por similaridade nas pesquisas por lotes coloniais através de nome de proprietário e localidade (colônia ou linha colonial). Uma vez que tais informações estão propensas a erros tipográficos e/ou grafias alternativas, elas foram escolhidas para a implementação de tal mecanismo. Tendo em vista que os bancos de dados comerciais não oferecem de forma direta a funcionalidade de processamento por aproximação de strings, a aplicação adota a solução proposta em (GRAVANO, 2005), a qual é explicada durante o restante do capítulo.

3.1 Ideia básica da solução

A técnica foi criada tendo como objetivo correlacionar informações, em formato string, de fontes diferentes (possivelmente de bancos de dados distintos), criando-se uma visão unificada das mesmas (por exemplo, dados de clientes em bases independentes, onde, em cada uma, o nome do mesmo cliente aparece com grafias ligeiramente diferentes). Para tanto, baseia-se no conceito de junções (*joins*) em bancos de dados relacionais. Contudo, a solução foi levemente adaptada na aplicação para que fosse implementada em buscas diretas por uma determinada string de pesquisa, ou seja, não se busca a correlação de dados.

Existem algumas dificuldades na implantação do processamento aproximado de strings em bancos de dados relacionais, uma vez que, conforme já mencionado, eles não oferecem tal suporte embutido. Há certas ferramentas que podem ser utilizadas fora do banco de dados, no pré-processamento das informações, o que, geralmente, não é desejável. Outra alternativa seria utilizar os algoritmos empregados por tais ferramentas

como UDFs (*User Defined Functions*) na base de dados. Contudo, tal abordagem é bastante ineficiente, principalmente tratando-se de junções, pois os bancos relacionais, ao avaliar junções envolvendo UDFs cujos argumentos incluem atributos pertencendo a múltiplas tabelas, computam o produto cartesiano das mesmas e aplicam a UDF após o processamento (GRAVANO, 2005).

Para enfrentar tais dificuldades, a solução utiliza técnicas para a identificação eficiente de todos os pares de strings com correspondência aproximada em uma base de dados. Para tanto, deve ser especificada uma métrica de aproximação. O trabalho utiliza, para tal finalidade, a chamada "distância de edição" (*edit distance*) entre duas strings. De acordo com o conceito, exclusão, inserção e substituição de um caractere são consideradas operações de custo unitário e a distância de edição entre duas strings é definida como a sequência de operações de menor custo que pode transformar uma string na outra.

A técnica de busca por similaridade abordada baseia-se no conceito de *q-grams*, que são pequenas substrings (de uma determinada palavra) de tamanho *q*, as quais são comparadas entre si em uma dada pesquisa. Ao levar-se em consideração o número total de correspondências nas comparações realizadas e as posições dos *q-grams* nessas comparações, garante-se que não haja falsas rejeições de resultados utilizando-se a métrica da distância de edição. Além disso, é realizada a identificação de um conjunto de pares candidatos com poucos "falsos positivos" (isto é, resultados que, a princípio, parecem corretos, mas que na verdade devem ser descartados) que pode ser verificado posteriormente para uma melhor correção do resultado final.

3.2 Detalhamento da solução

A fim de detalhar a técnica de busca por aproximação, inicialmente serão apresentadas algumas definições básicas referentes aos conceitos da solução. Posteriormente, será explicado como tais conceitos são aplicados em um banco de dados relacional para que a técnica funcione de maneira satisfatória.

3.2.1 Notações básicas

Algumas notações são utilizadas durante o decorrer da explicação, as quais são definidas conforme se segue.

Usa-se *R*, possivelmente com subscritos, para denotar tabelas, *A*, possivelmente com subscritos, para denotar atributos de uma tabela e *t*, possivelmente com subscritos, para denotar registros em uma tabela. Usa-se a notação $R.A_i$ para referência ao atributo A_i da tabela *R* e $R.A_i(t_j)$ para referência ao valor do atributo $R.A_i$ do registro t_j.

Usa-se $\sum$ como um alfabeto finito de tamanho $|\sum|$. Faz-se o uso de símbolos gregos em letra minúscula, tal como σ, possivelmente com subscritos, para denotar strings em $\sum^*$. Sendo $\sigma \in \sum^*$ uma string de tamanho *n*, usa-se $\sigma[i...j]$, $1 \leq i \leq j \leq n$, para denotar uma substring de σ de tamanho $j - i + 1$ começando na posição *i*.

3.2.2 *Q-grams*

Dada uma string σ, seus *q-grams* posicionais são obtidos "deslizando-se" uma janela de tamanho *q* sobre os caracteres de σ. Uma vez que os *q-grams* no início e no fim da string podem ter menos do que *q* caracteres, são introduzidos novos caracteres "#" e "$", os quais não estão em $\sum$, e a string σ é estendida conceitualmente ao ser prefixada

com q – 1 ocorrências de "#" e sufixada com q – 1 ocorrências de "$". Deste modo, cada *q-gram* contém exatamente q caracteres, apesar de alguns deles não fazerem parte do alfabeto $\sum$.

Um *q-gram* posicional de uma string σ é um par (i, $\sigma[i...i + q - 1]$), onde $\sigma[i...i + q - 1]$ é o *q-gram* de σ que inicia na posição i, levando-se em consideração a string estendida. O conjunto G_σ de todos os *q-grams* posicionais de uma string σ é o conjunto de todos os $|\sigma| + q - 1$ pares construídos de todos os *q-grams* de σ.

A ideia do uso de *q-grams* como base do processamento de strings por aproximação vem do fato de que, quando duas strings σ_1 e σ_2 estão numa pequena distância de edição entre si, elas compartilham um grande número de *q-grams* em comum (UKKONEN, 1992). O exemplo a seguir ilustra essa observação.

Os *q-grams* posicionais de tamanho q=3 da string "john_smith" são {(1, ##j), (2, #jo), (3, joh), (4, ohn), (5, hn_), (6, n_s), (7, _sm), (8, smi), (9, mit), (10, ith), (11, th$), (12, h$$)}. De forma similar, os *q-grams* posicionais de tamanho q=3 da string "john_a_smith", a qual está numa distância de edição de dois para "john_smith", são {(1, ##j), (2, #jo), (3, joh), (4, ohn), (5, hn_), (6, n_a), (7, _a_), (8, a_s), (9, _sm), (10, smi), (11, mit), (12, ith), (13, th$), (14, h$$)}. Se a informação de posição for ignorada, os dois conjuntos de *q-grams* têm onze *q-grams* em comum. Curiosamente, apenas os primeiro cinco *q-grams* da primeira string são também *q-grams* posicionais da segunda string. Entretanto, outros seis *q-grams* das duas strings diferem em apenas duas posições. Isso ilustra que, em geral, o uso de *q-grams* posicionais para o processamento de strings por aproximação envolve a comparação de posições de *q-grams* correspondentes dentro de uma certa "faixa".

3.2.3 Problema a ser resolvido

Formalmente, o problema a ser resolvido pela técnica é definido conforme se segue. Dadas tabelas R_1 e R_2 com atributos string $R_1.A_i$ e $R_2.A_j$ e um inteiro k, recuperar todos os pares de registros (t, t') $\in R_1 \times R_2$ tal que edit_distance($R_1.A_i(t)$, $R_2.A_j(t')$) $\leq k$.

Conforme já mencionado, a técnica baseia-se em dois passos principais. Primeiro, um conjunto de resultados candidatos é obtido utilizando-se um algoritmo simples e aproximado que garante nenhuma falsa rejeição. Isso é alcançado ao se realizar uma junção dos *q-grams* juntamente com alguns filtros adicionais que garantem a não-eliminação de qualquer correspondência aproximada real. Após, no segundo passo, usa-se um algoritmo custoso, em memória, para verificação da distância de edição entre cada par de strings candidato e a eliminação de todos os falsos positivos.

3.2.4 Solução simplista

O problema pode ser facilmente expressado em qualquer banco de dados relacional que ofereça suporte a UDFs, tais como Oracle ou SQL Server. Poderia ser registrada na base uma função edit_distance(*s1*, *s2*, *k*) que retorna verdadeiro se as duas strings utilizadas como argumentos estiverem dentro de uma distância de edição do argumento inteiro *k*. Então, o problema de junção de string por aproximação para a distância de edição *k* poderia ser representado, em SQL (*Structured Query Language*), como:

```
SELECT      R1.Ai, R2.Aj
FROM        R1, R2
WHERE       edit_distance(R1.Ai, R2.Aj, k)
```

Para avaliar essa expressão, os motores relacionais deveriam, essencialmente, ter que computar o produto cartesiano das tabelas R_1 e R_2 e aplicar a comparação da UDF como um filtro de pós-processamento. Entretanto, produtos cartesianos de tabelas grandes são imensos e a invocação da UDF, a qual é um elemento custoso, em cada registro do produto cartesiano faz com que o custo da operação de junção seja proibitivo. Por essas razões, a técnica busca uma solução melhor, a qual é baseada no que se segue nas próximas seções.

3.2.5 Preparação do banco de dados

Para permitir o processamento aproximado de strings em um banco de dados através do uso de *q-grams*, é necessário um mecanismo para popular a base com os *q-grams* posicionais correspondentes às strings originais do banco. A explicação que se segue detalha o processo.

R é uma tabela com colunas (A_0, A_1, ..., A_m), tal que A_0 é a chave primária de R (identificando unicamente seus registros) e alguns atributos A_i, $i > 0$, são valores string. Para cada atributo string A_i que será considerado para o processamento aproximado de strings, é criada uma tabela auxiliar $RA_iQ(A_0, Pos, Qgram)$ com três atributos. Para uma string σ do atributo A_i de um registro de R, seus $|\sigma| + q - 1$ *q-grams* posicionais são representados como registros distintos na tabela RA_iQ, onde $RA_iQ.Pos$ identifica a posição do *q-gram* contido em $RA_iQ.Qgram$. Esses $|\sigma| + q - 1$ registros compartilham o mesmo valor para o atributo $RA_iQ.A_0$, o qual é uma chave estrangeira para a tabela R.

3.2.6 Filtragem de resultados através de propriedades dos *q-grams*

Para alcançar os dois objetivos principais do uso de *q-grams* juntamente com a métrica de distância de edição – nenhuma falsa rejeição e poucos falsos positivos – a técnica utiliza-se de três propriedades chave dos *q-grams*, as quais são utilizadas em três técnicas de filtragem explicadas a seguir.

3.2.6.1 Filtro por contagem

A ideia básica do filtro por contagem é aproveitar a informação obtida através dos conjuntos $G_{\sigma 1}$ e $G_{\sigma 2}$ dos *q-grams* das strings σ_1 e σ_2, ignorando-se a informação posicional, determinando se σ_1 e σ_2 estão numa distância de edição menor ou igual a k. O princípio é que strings que estão numa distância de edição pequena entre si compartilham um grande número de *q-grams* em comum.

Considerando-se uma string σ_1 e outra string σ_2, a qual é obtida através da substituição de um único caractere de σ_1, os conjuntos de *q-grams* $G_{\sigma 1}$ e $G_{\sigma 2}$ diferem no máximo por q (o tamanho do *q-gram*). Isso ocorre pois *q-grams* que não sobrepõem-se ao caractere substituído devem ser comuns aos dois conjuntos, e há apenas q *q-grams* que sobrepõem-se ao caractere substituído. De forma similar, o mesmo ocorre para inserções e remoções de um único caractere. Em outras palavras, nesses casos, σ_1 e σ_2 devem ter pelo menos $(\max(|\sigma_1|, |\sigma_2| + q - 1) - q = \max(|\sigma_1|, |\sigma_2|) - 1$ *q-grams* em comum. Quando a distância de edição entre σ_1 e σ_2 é k, o limite inferior do número de correspondências de *q-grams* é o valor apresentado na preposição a seguir.

Preposição 3.1 Considere strings σ_1 e σ_2 de tamanhos $|\sigma_1|$ e $|\sigma_2|$ respectivamente. Se σ_1 e σ_2 estão dentro de uma distância de edição de k, então a cardinalidade de $G_{\sigma 1} \cap G_{\sigma 2}$, ignorando-se informação posicional, deve ser de pelo menos $\max(|\sigma_1|, |\sigma_2|) - 1 - (k - 1) * q$.

3.2.6.2 Filtro por posição

Enquanto o filtro por contagem é efetivo em aumentar a eficiência do processamento aproximado de *strings*, não aproveita a informação posicional de um *q-gram*.

Em geral, a relação entre as posições de correspondências de *q-grams* e a distância de edição é bastante complexa. Qualquer dado *q-gram* em uma string pode não ocorrer em outra string, e as posições de *q-grams* sucessivos podem estar fora da faixa válida devido a inserções e remoções. Além disso, existe a possibilidade de um *q-gram* em uma string ocorrer em múltiplas posições em outra string.

Um *q-gram* posicional (i, τ_1) em uma string σ_1 corresponde a um *q-gram* posicional (j, τ_2) em outra string σ_2 se $\tau_1 = \tau_2$ e (i, τ_1), após a sequência de operações de edição que converte σ_1 em σ_2, torna-se o *q-gram* (j, τ_2) na string editada (ULLMANN, 1977).

Exemplo: Considere as strings σ_1 = abaxabaaba e σ_2 = abaabaaba. A distância de edição entre essas strings é de um (remover x para transformar a primeira string na segunda). Então (7, *aba*) em σ_1 corresponde a (6, *aba*) em σ_2 mas não a (9, *aba*).

Não obstante a complexidade de corresponder *q-grams* posicionais na presença de erros de edição em strings, um filtro útil pode ser planejado baseando-se na seguinte observação.

Preposição 3.2 Se strings σ_1 e σ_2 estão dentro de uma distância de edição de k, então um *q-gram* posicional em uma string não pode corresponder a um *q-gram* posicional da outra que difira da mesma em mais de k posições.

3.2.6.3 Filtro por tamanho

Finalmente, observa-se que o tamanho da string provê informação útil para rapidamente descartar strings que não estão na distância de edição desejada.

Preposição 3.3 Se duas strings σ_1 e σ_2 estão dentro de uma distância de edição de k, seus tamanhos não podem diferir em mais de k.

3.2.7 Expressão e avaliação SQL

Algo particularmente interessante é que os filtros por contagem, posição e tamanho podem ser naturalmente expressos como uma expressão SQL na base de dados populada descrita na seção 3.2.4, e eficientemente implementada por um banco de dados relacional comercial. A Figura 3.1 mostra a expressão SQL reunindo as três propriedades descritas.

```
SELECT    R1.A0 , R2.A0 , R1.Ai , R2.Aj
FROM      R1 , R1AiQ , R2 , R2AjQ
WHERE     R1.A0 = R1AiQ.A0 AND
          R2.A0 = R2AjQ.A0 AND
          R1AiQ.Qgram = R2AjQ.Qgram AND
          |R1AiQ.Pos - R2AjQ.Pos| ≤ k AND
          |strlen(R1.Ai) - strlen(R2.Aj)| ≤ k
GROUP BY  R1.A0, R2.A0, R1.Ai, R2.Aj
HAVING    COUNT(*) ≥ strlen(R1.Ai) - 1 - (k - 1) * q AND
          COUNT(*) ≥ strlen(R2.Aj) - 1 - (k - 1) * q AND
          edit_distance(R1.Ai, R2.Aj, k)
```

Figura 3.1: Expressão SQL contendo as três propriedades dos *q-grams* (GRAVANO, 2005)

Essencialmente, a expressão SQL da Figura 3.1 realiza uma junção das tabelas auxiliares correspondentes aos atributos string $R_1.A_i$ e $R_2.A_j$ em seus atributos *Qgram*, além de fazer junções de chaves estrangeiras/chaves primárias com as tabelas originais R_1 e R_2 para recuperar os pares de strings que devem ser retornados ao usuário.

O filtro por posição é implementado como uma condição da cláusula WHERE da expressão SQL. A condição remove quaisquer pares de strings em $R_1 \times R_2$ que compartilham muitos *q-grams* em comum mas que as posições dos *q-grams* idênticos difiram substancialmente. Deste modo, tais pares de strings serão desconsiderados antes que as condições COUNT(*) na cláusula HAVING sejam testadas. Além disso, esse filtro reduz o tamanho da junção dos *q-grams*, deixando a computação da expressão mais rápida, uma vez que menos pares de *q-grams* devem ser examinados pelas cláusulas GROUP BY e HAVING.

O filtro por tamanho é implementado como uma condição adicional na cláusula WHERE da expressão, a qual compara os tamanhos de duas strings. Novamente, assim como a técnica de filtro por posição, este filtro reduz o tamanho da junção dos *q-grams* e, em consequência, o tamanho do conjunto candidato.

Finalmente, o filtro por contagem é implementado principalmente pelas condições na cláusula HAVING. Os pares de strings que compartilham somente alguns poucos *q-grams* são eliminados pelas condições do COUNT(*) na cláusula HAVING. Quaisquer pares de strings em $R_1 \times R_2$ que não compartilham *q-grams* são descartados pelas condições na cláusula WHERE.

Contudo, mesmo após os passos de filtragem, o conjunto candidato pode ainda ter falsos positivos. Deste modo, a custosa invocação da UDF edit_distance($R_1.A_i$, $R_2.A_j$, k) ainda deve ser realizada, mas, desta vez, em apenas uma pequena fração de todos os pares possíveis de strings.

3.3 Utilização da técnica na aplicação

Uma vez que o objetivo da busca por similaridade de strings na aplicação é de localizar correspondências diretas com um determinado nome informado pelo usuário (isto é, não se procura a correlação de dados), a técnica apresentada foi levemente adaptada. Utilizando-se a expressão apresentada na Figura 3.1 para comparações, as seguintes mudanças foram realizadas.

Apenas os dados da tabela R_1 são de interesse, já que a tabela R_2 apenas correlaciona seus dados com a tabela R_1. Desta forma, as referências à tabela R_2 na cláusula SELECT foram eliminadas. Nas demais cláusulas, o atributo $R_2.A_j$ foi substituído diretamente pela string de consulta. Na cláusula FROM, a tabela R_2 também foi descartada; entretanto, a tabela R_2A_jQ foi substituída por uma tabela que é populada dinamicamente a cada pesquisa efetuada. Essa tabela armazena os *q-grams* do nome pesquisado pelo usuário (valor do atributo $R_2.A_j$), uma vez que é realizada uma junção entre a mesma e a tabela de *q-grams* da tabela R_1. A cláusula WHERE mantém-se inalterada, com exceção da junção envolvendo a tabela R_2, a qual foi removida. A cláusula GROUP BY mantém apenas os campos constantes na cláusula SELECT, isto é, atributos referentes à tabela R_1. Finalmente, a chamada à UDF `edit_distance` não ocorre na cláusula HAVING. Ao invés disso, ela aparece numa cláusula ORDER BY, pois, dessa forma, os resultados já vêm ordenados conforme a distância de edição para a string de pesquisa (em ordem ascendente de distância). Uma vez que os filtros dos *q-grams* já foram aplicados antes da chamada à função e que a pesquisa limita o número de registros retornados em 5, 10, 15 ou 20 ocorrências, a invocação da UDF não é impactante no desempenho da consulta.

Como exemplo, será apresentada a expressão SQL formada para a pesquisa de um determinado proprietário através do sobrenome “Corso”, com o limite de até 20 registros. Na aplicação, a distância de edição considerada e o tamanho dos *q-grams* é de três. O banco de dados utilizado é o SQL Server (instruções TOP e LEN). A pesquisa também é realizada na tabela Lote pois apenas os proprietários que têm lote associado devem ser recuperados.

```
SELECT     TOP 20 PRO.ID, PRO.Nome, PRO.Sobrenome
FROM       Proprietario PRO, Lote LOT, Proprietario_Qgram PRQ,
           Consulta_Qgram COQ
WHERE      PRO.Chave_Lote = LOT.Chave AND
           PRO.ID = PRQ.Proprietario_ID AND
           PRQ.Qgram = COQ.Qgram AND
           ABS(PRQ.Posicao – COQ.Posicao) <= 3 AND
           ABS(LEN(PRO.Sobrenome) – LEN('Corso')) <= 3
GROUP BY   PRO.ID, PRO.Nome, PRO.Sobrenome
HAVING     COUNT(*) >= LEN(PRO.Sobrenome) – 1 – (3 - 1) * 3 AND
           COUNT(*) >= LEN('Corso') – 1 – (3 – 1) * 3
ORDER BY   edit_distance(PRO.Sobrenome, 'Corso')
```

A tabela Consulta_Qgram é a tabela populada a cada pesquisa com os *q-grams* da string sendo consultada (no exemplo, os *q-grams* da palavra Corso). Ela exerce o papel da tabela R_2A_jQ mostrada na expressão da Figura 3.1.

4 FUNCIONALIDADES DA APLICAÇÃO

Para demonstrar as funcionalidades oferecidas pela aplicação (isto é, o que o sistema provê), utilizar-se-á a linguagem de modelagem UML (*Unified Modeling Language*). A UML é uma família de notações gráficas que ajuda a descrever e projetar sistemas de software, especialmente aplicações construídas utilizando-se o estilo orientado a objetos (FOWLER, 2005). Tal linguagem é largamente utilizada na área de Engenharia de Software e sua padronização foi criada e é controlada pelo OMG (*Object Management Group*).

4.1 Diagrama de casos de uso

Utilizamos diagramas de casos de uso em UML para demonstrar, graficamente e em alto nível, como os usuários interagem com o sistema. Através de um diagrama, pode-se distinguir rapidamente a quais funcionalidades do sistema cada tipo de usuário tem acesso. Na Figura 4.1, é apresentado o diagrama de casos de uso da aplicação.

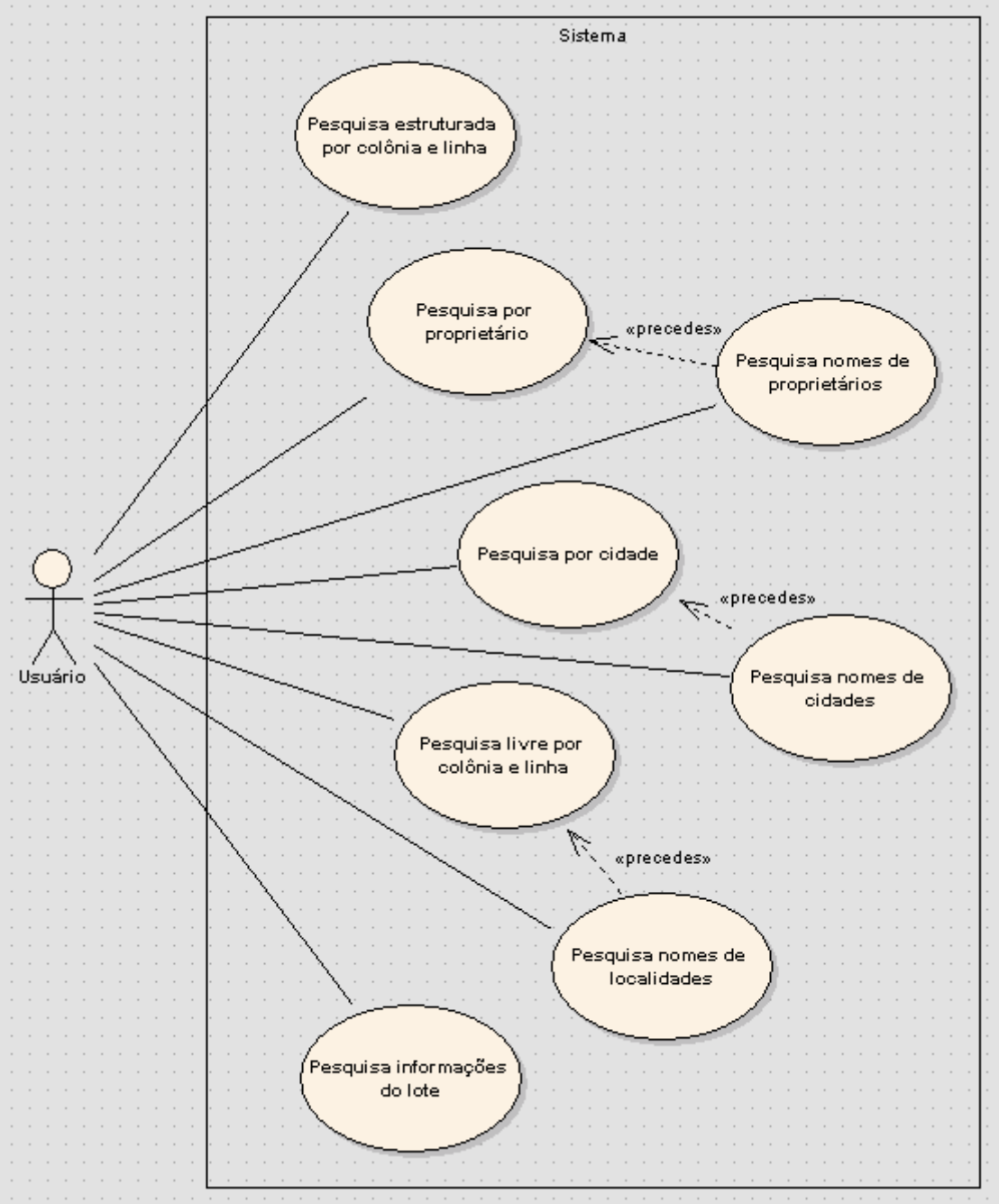

Figura 4.1: Diagrama de casos de uso da aplicação

No diagrama da Figura 4.1 podem-se visualizar três tipos principais de elementos: atores, casos de uso e associações.

Os atores são os agentes externos que interagem com o sistema, ou seja, não estão sob a influência do mesmo. Na aplicação, tem-se apenas um tipo de ator: o usuário, o qual inicia diretamente todos os casos de uso.

Os casos de uso representam unidades discretas de interação entre um ator e o sistema. Descrevem o comportamento do sistema ao responder às requisições de um determinado ator. Casos de uso não descrevem como uma determinada funcionalidade deve ser construída e sim qual será o seu comportamento (o que ela oferece).

As associações são ligações entre diferentes elementos e, no diagrama da Figura 4.1, dividem-se em dois tipos: associação entre ator e caso de uso e associação entre casos de uso. A associação entre ator e caso de uso caracteriza uma interação (requisição) entre o ator e alguma funcionalidade do sistema. A associação entre casos de uso representa como tais entidades relacionam-se entre si. Na aplicação, existe apenas um tipo de relacionamento entre casos de uso: precedência, no qual um caso de uso (o apontador) precede outro caso de uso (o apontado).

A finalidade dos casos de uso “Pesquisa estruturada por colônia e linha”, “Pesquisa por proprietário”, “Pesquisa por cidade” e “Pesquisa livre por colônia e linha” é a mesma: exibir lotes coloniais no mapa. A diferença entre eles é o meio (dados selecionados e/ou informados pelo usuário) através do qual se chega aos resultados. O caso de uso “Pesquisa informações do lote” é um complemento de qualquer um dos outros quatro casos de uso mencionados anteriormente, pois apresenta ao usuário informações pertinentes a um dado lote colonial selecionado no mapa.

Os casos de uso precedentes – “Pesquisa nomes de proprietários”, “Pesquisa nomes de cidades” e “Pesquisa nomes de localidades” – têm como finalidade auxiliar o usuário na identificação das informações cadastradas no banco de dados. Deste modo, por exemplo, ao informar o nome de uma cidade, ele já fica ciente das cidades cujos nomes iniciam pelos dados já informados. A relação de precedência indica sequência no tempo e não dependência – com exceção do caso de uso “Pesquisa informações do lote” (o qual necessita que haja lotes coloniais exibidos no mapa), todos os casos de uso são independentes entre si.

4.2 Descrições textuais de casos de uso e telas de interface

O diagrama de casos de uso apresentado na Figura 4.1 é útil para caracterizar, graficamente, as funcionalidades oferecidas pela aplicação. Entretanto, não especifica quais são os passos necessários, na interação entre usuário e sistema, para se chegar a tais fins. Para tanto, faz-se o uso de descrições textuais de casos de uso, as quais são divididas em UML, usualmente, em duas categorias: alto nível e expandidas. Contudo, ambas serão apresentadas de forma única neste trabalho, visando a facilitação da leitura.

As descrições de casos de uso explicitam, passo a passo, as interações entre os atores e os casos de uso do sistema. Além disso, detalham todas as possíveis variações do fluxo principal de eventos da interação. Também apresentam a lógica do processo do caso de uso, enriquecendo as informações já fornecidas pelo diagrama.

Acompanhando as descrições, serão apresentadas as telas de interface para cada caso de uso, demonstrando, na prática, a interação entre o usuário e o sistema. A Figura 4.2 apresenta a tela inicial do sistema, antes de qualquer caso de uso ter sido iniciado pelo usuário.

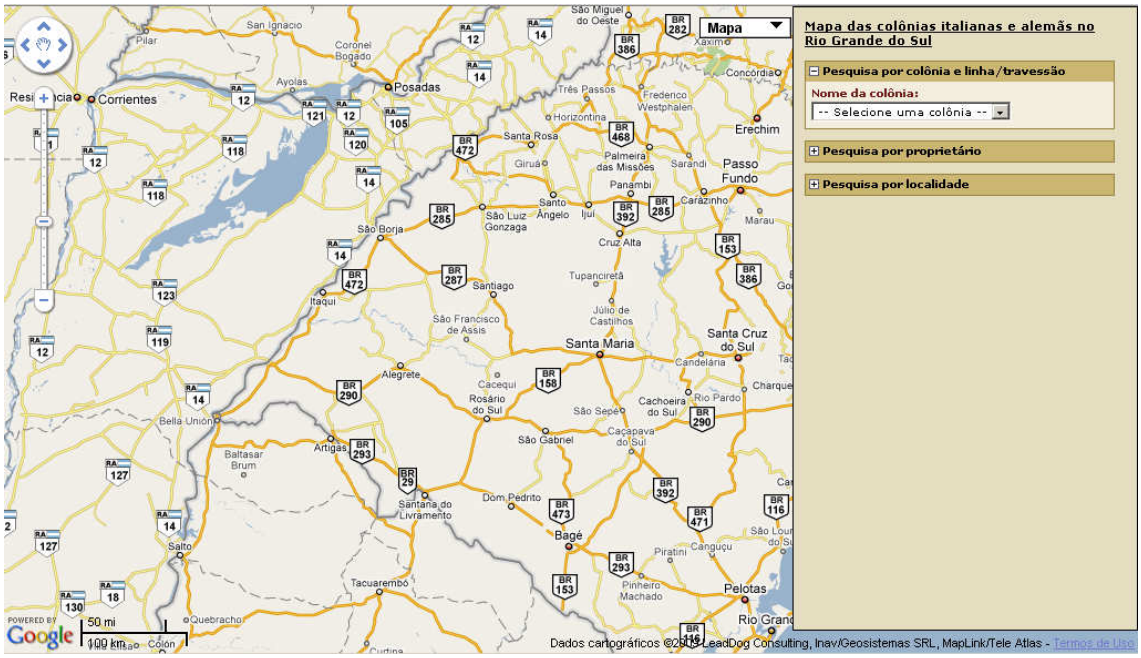

Figura 4.2: Tela de interface inicial da aplicação

A Figura 4.2 mostra que existem três seções principais no menu da aplicação: “Pesquisa por colônia e linha”, “Pesquisa por proprietário” e “Pesquisa por localidade”. Cada seção representa um tipo de pesquisa diferente e um caso de uso diferente, exceto a seção “Pesquisa por localidade”, a qual abrange os casos de uso “Pesquisa por cidade” e “Pesquisa livre por colônia e linha”. A seção “Pesquisa por colônia e linha” é a única expandida por padrão, sendo considerada a principal do menu.

Tabela 4.1: Descrição do caso de uso “Pesquisa estruturada por colônia e linha”

UC 01 – Pesquisa estruturada por colônia e linha	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Lotes da linha colonial selecionada exibidos no mapa.
Descrição:	Usuário pesquisa por lotes selecionando uma colônia através de seu nome e, após isso, seleciona uma linha pertencente à colônia selecionada, também através de seu nome. Relação de lotes da linha selecionada é mostrada ao usuário no mapa.
Sequência de eventos:	
Ator	**Sistema**
1. Seleciona uma colônia através de seu	

nome em uma caixa de seleção.	
	2. Apresenta, em uma caixa de seleção, uma lista de linhas pertencentes à colônia selecionada.
3. Seleciona uma linha através de seu nome na caixa de seleção apresentada.	
	4. Exibe no mapa os lotes da linha colonial selecionada.
5. O usuário visualiza os lotes da linha selecionada e o caso de uso é encerrado.	

Na Tabela 4.1 é detalhada a pesquisa estruturada de lotes através de colônia e linha, constante na seção “Pesquisa por colônia e linha” do menu da aplicação.

Uma pré-condição é o estado no qual o sistema deve estar antes do caso de uso ser iniciado. Uma pós-condição é o estado no qual o sistema ficará após o caso de uso ser finalizado. Conforme já mencionado na seção 4.1, com exceção do caso de uso “Pesquisa informações do lote”, não existe dependência entre os casos de uso do sistema e, portanto, nenhum deles exige pré-condição. Uma vez que o único ator da aplicação é o usuário, todas as interações são realizadas entre o usuário e o sistema.

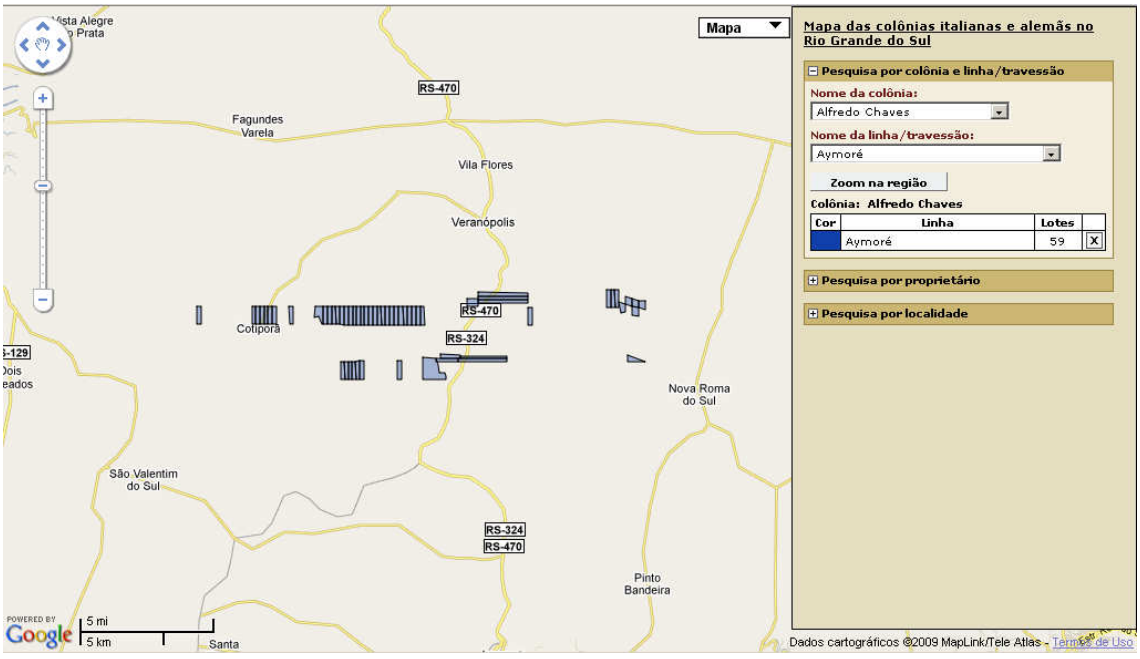

Figura 4.3: Tela de interface do caso de uso “Pesquisa estruturada por colônia e linha”

A Figura 4.3 mostra a pós-condição do caso de uso “Pesquisa estruturada por colônia e linha”, ou seja, lotes coloniais exibidos no mapa. Conforme detalhado na Tabela 4.1, o usuário seleciona o nome da colônia através de uma caixa de seleção e, após isso, o nome da linha pertencente à colônia através de outra caixa de seleção.

Dessa forma, os lotes coloniais da linha são apresentados no mapa, além de ser criada uma tabela no menu contendo as linhas atualmente exibidas para aquela colônia.

É possível que o usuário selecione mais de uma linha colonial na colônia selecionada, sendo que, dessa forma, várias linhas são exibidas no mapa e na tabela do menu. Tal situação é demonstrada na Figura 4.4.

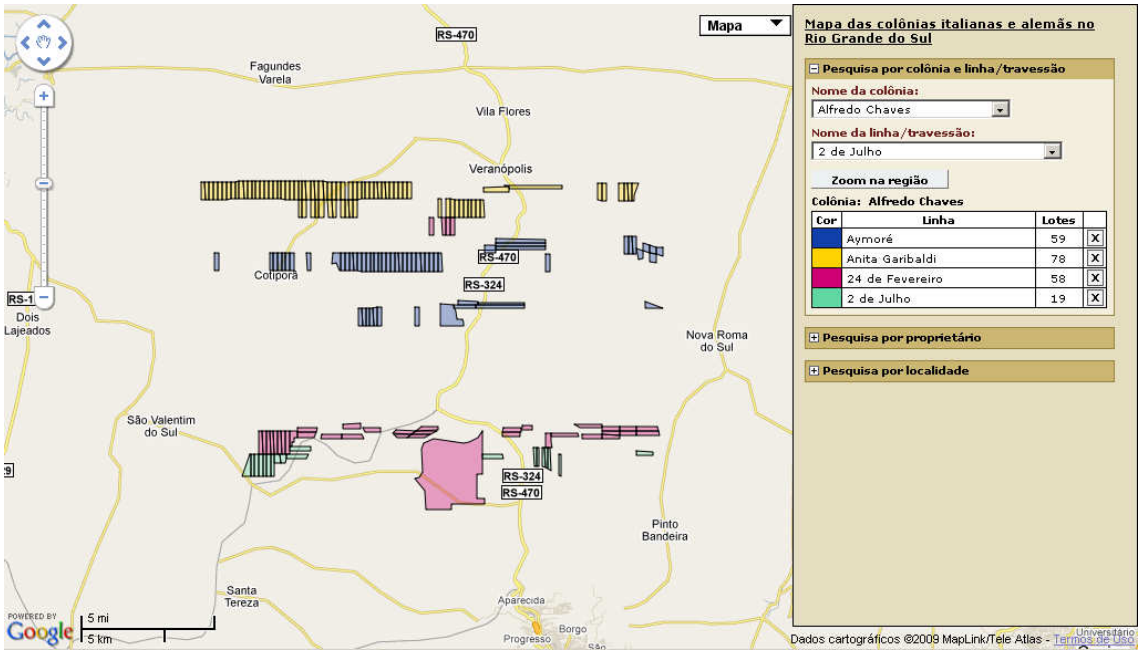

Figura 4.4: Tela de interface do caso de uso “Pesquisa estruturada por colônia e linha” com mais de uma linha colonial selecionada

O menu de linhas coloniais da colônia selecionada contém uma cor para cada linha, facilitando, dessa forma, a distinção dos lotes exibidos no mapa.

Tabela 4.2: Descrição do caso de uso “Pesquisa nomes de proprietários”

UC 02 – Pesquisa nomes de proprietários	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Lista com sobrenomes de proprietários que iniciem pelo texto informado apresentada.
Descrição:	Usuário pesquisa por lotes informando o nome do proprietário e, enquanto isso, é apresentada ao mesmo uma lista com todos os nomes que iniciem pelo texto já informado (precede o caso de uso “Pesquisa por proprietário”).
Sequência de eventos:	

Ator	Sistema
1. Começa a informar o nome da família de um proprietário para pesquisa de seus lotes, conforme UC 02.	
	2. Apresenta uma lista com sobrenomes de proprietários que iniciem pelo texto já informado pelo usuário.
3. O usuário visualiza a lista de sobrenomes e o caso de uso é encerrado.	
RN 01 – Regra de negócio 01:	
1. A pesquisa por nomes de proprietários deve ser iniciada somente a partir da 2ª (segunda) letra informada pelo usuário.	
RN 02 – Regra de negócio 02:	
2. A pesquisa por nomes de proprietários deve ser realizada sempre que o usuário informar uma nova letra do nome (a partir da 2ª, conforme RN 01).	

Na Tabela 4.2 é detalhada a pesquisa automática de nomes de proprietários, constante na seção “Pesquisa por proprietário” do menu da aplicação.

O caso de uso apresenta duas regras de negócio, as quais são complementos da especificação do caso de uso e que sempre devem ser obedecidas. Elas ficam em uma seção à parte da sequência de eventos pois são mutáveis, isto é, podem ser modificadas ou removidas.

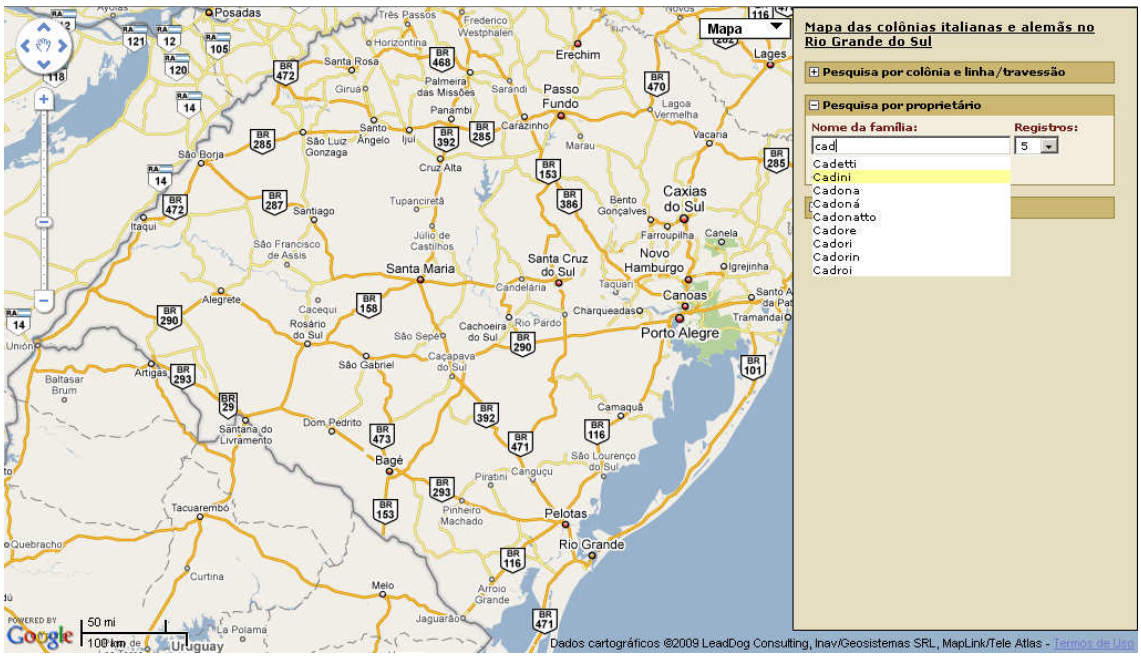

Figura 4.5: Tela de interface do caso de uso “Pesquisa nomes de proprietários”

A Figura 4.5 mostra a pós-condição do caso de uso “Pesquisa nomes de proprietários”, ou seja, lista de sobrenomes de proprietários iniciados pelo texto informado pelo usuário é apresentada. Conforme detalhado na Tabela 4.2, assim que o usuário começa a digitar (a partir da segunda letra) o nome desejado, o sistema apresenta uma lista com os nomes que iniciam com o texto já informado, atualizando-a a cada nova letra. Dessa forma, o usuário pode selecionar um dos nomes da lista e iniciar o caso de uso “Pesquisa por proprietário” de forma rápida.

Tabela 4.3: Descrição do caso de uso “Pesquisa por proprietário”

UC 03 – Pesquisa por proprietário	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Lotes do proprietário selecionado exibidos no mapa.
Descrição:	Usuário pesquisa por lotes informando o nome da família do proprietário. Relação de proprietários com sobrenomes iguais e/ou similares ao informado é apresentada ao usuário. Usuário seleciona um dos proprietários da lista e os lotes do mesmo são mostrados no mapa.
Sequência de eventos:	
Ator	**Sistema**
1. Informa o nome da família do proprietário e o número máximo de registros a serem retornados.	
	2. Apresenta uma lista de proprietários com sobrenome igual e/ou similar ao informado.
3. Seleciona um dos proprietários da lista.	
	4. Exibe no mapa os lotes do proprietário selecionado.
5. O usuário visualiza os lotes do proprietário selecionado e o caso de uso é encerrado.	
Fluxo alternativo 01:	
1.1. Informa um nome de família	

inexistente e sem similaridade com qualquer nome existente.	
	2.1. Informa que não há registros de proprietários com o sobrenome informado. O caso de uso é encerrado.

Na Tabela 4.3 é detalhada a pesquisa de lotes através de proprietário, constante na seção “Pesquisa por proprietário” do menu da aplicação.

O caso de uso apresenta um fluxo alternativo, o qual apresenta a sequência de eventos caso a interação entre ator e sistema não ocorra exatamente conforme descrita no fluxo principal. Nesse caso, se o usuário informar um nome inexistente para a pesquisa (sem similaridade com os nomes cadastrados), o sistema deve apresentar uma mensagem informando que não há registros para o nome informado, encerrando o caso de uso.

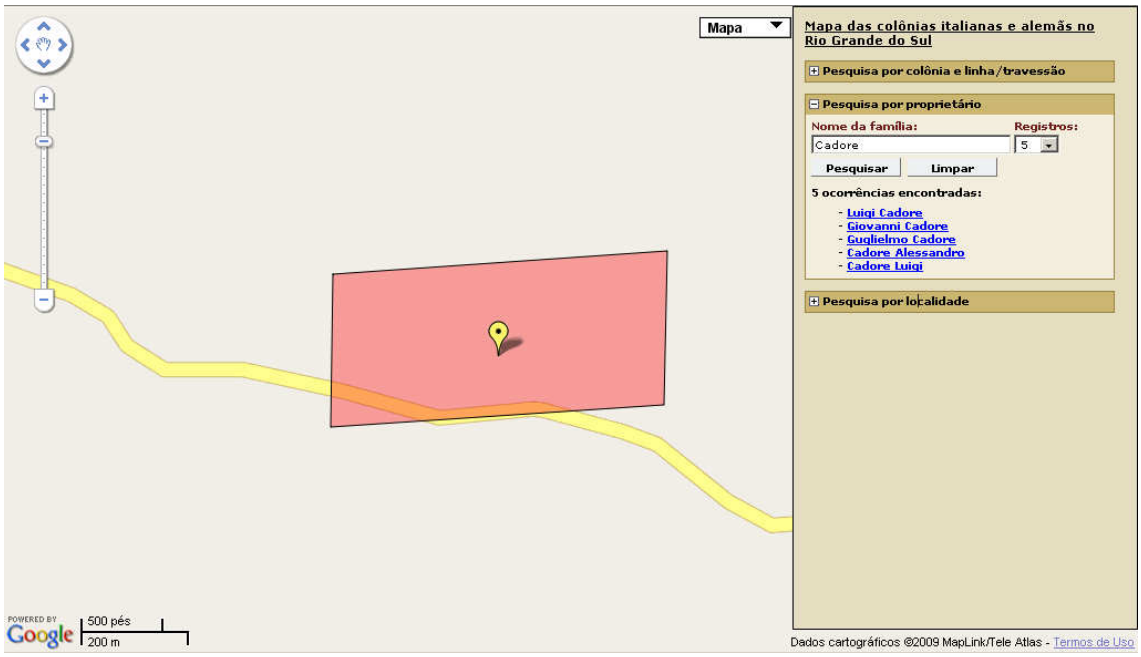

Figura 4.6: Tela de interface do caso de uso “Pesquisa por proprietário”

A Figura 4.6 mostra a pós-condição do caso de uso “Pesquisa por proprietário”, ou seja, o(s) lote(s) do proprietário exibido(s) no mapa. Conforme detalhado na Tabela 4.3, após o usuário ter informado o nome da família desejada, o sistema apresenta uma lista com os proprietários de sobrenome igual e/ou similar ao informado. Depois disso, basta o usuário selecionar um dos proprietários da lista e seu(s) lote(s) é(são) exibido(s) no mapa.

Figura 4.7: Tela de interface do caso de uso “Pesquisa por proprietário” em seu fluxo alternativo

A Figura 4.7 mostra o fluxo alternativo do caso de uso “Pesquisa por proprietário”, no qual o sistema informa ao usuário que não há registros com o nome informado e nem similares a ele.

Tabela 4.4: Descrição do caso de uso “Pesquisa nomes de cidades”

UC 04 – Pesquisa nomes de cidades	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Lista com nomes de cidades que iniciem pelo texto informado apresentada.
Descrição:	Usuário pesquisa por lotes informando o nome da cidade e, enquanto isso, é apresentada ao mesmo uma lista com todos os nomes que iniciem pelo texto já informado (precede o caso de uso “Pesquisa por cidade”).
Sequência de eventos:	
Ator	**Sistema**
1. Começa a informar o nome da cidade para pesquisa de seus lotes, conforme UC 03.	

	2. Apresenta uma lista com nomes de cidades que iniciem pelo texto já informado pelo usuário.
3. O usuário visualiza a lista de nomes e o caso de uso é encerrado.	
RN 01 – Regra de negócio 01:	
1. A pesquisa por nomes de cidades deve ser iniciada somente a partir da 2ª (segunda) letra informada pelo usuário.	
RN 02 – Regra de negócio 02:	
2. A pesquisa por nomes de cidades deve ser realizada sempre que o usuário informar uma nova letra do nome (a partir da 2ª, conforme RN 01).	

Na Tabela 4.4 é detalhada a pesquisa automática de nomes de cidades, constante na seção “Pesquisa por localidade” do menu da aplicação.

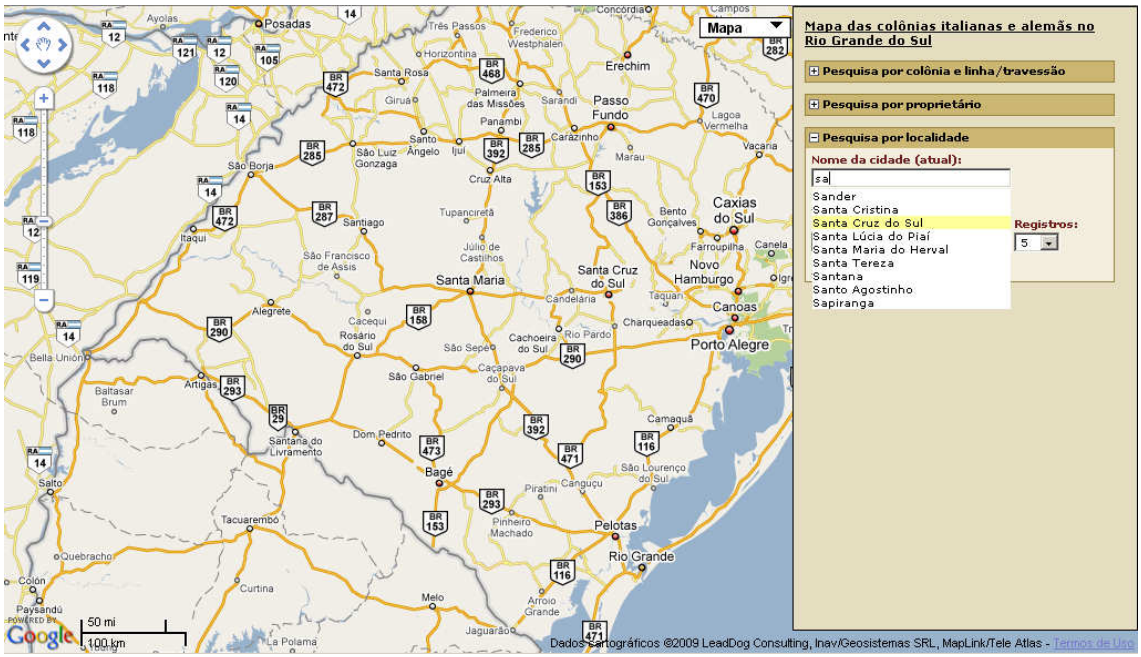

Figura 4.8: Tela de interface do caso de uso “Pesquisa nomes de cidades”

A Figura 4.8 mostra a pós-condição do caso de uso “Pesquisa nomes de cidades”, ou seja, lista de nomes de cidades iniciados pelo texto informado pelo usuário é apresentada. Sua forma de funcionamento é idêntica ao caso de uso “Pesquisa nomes de proprietários” descrita anteriormente.

Tabela 4.5: Descrição do caso de uso “Pesquisa por cidade”

UC 05 – Pesquisa por cidade	
Atores:	Usuário.

Pré-condição:	Nenhuma.
Pós-condição:	Lotes de uma determinada linha colonial situada na cidade informada exibidos no mapa.
Descrição:	Usuário pesquisa por lotes informando o nome da cidade. Relação de linhas coloniais com lotes na cidade informada é apresentada ao usuário. Usuário seleciona uma das linhas da lista e os lotes da mesma são mostrados no mapa.
Sequência de eventos:	
Ator	**Sistema**
1. Informa o nome da cidade.	
	2. Apresenta uma lista de linhas coloniais que tenham lotes na cidade informada.
3. Seleciona uma das linhas coloniais da lista.	
	4. Exibe no mapa os lotes da linha selecionada.
5. O usuário visualiza os lotes da linha selecionada e o caso de uso é encerrado.	
Fluxo alternativo 01:	
1.1. Informa um nome de cidade inexistente ou sem registro de linhas coloniais.	
	2.1. Informa que não há registros de linhas coloniais na cidade informada.

Na Tabela 4.5 é detalhada a pesquisa de lotes através de cidade, constante na seção “Pesquisa por localidade” do menu da aplicação.

O caso de uso apresenta um fluxo alternativo semelhante ao caso de uso “Pesquisa por proprietário”. Entretanto, o nome informado pelo usuário deve ser idêntico ao cadastrado no sistema, uma vez que não é efetuada pesquisa por similaridade em nomes de cidades.

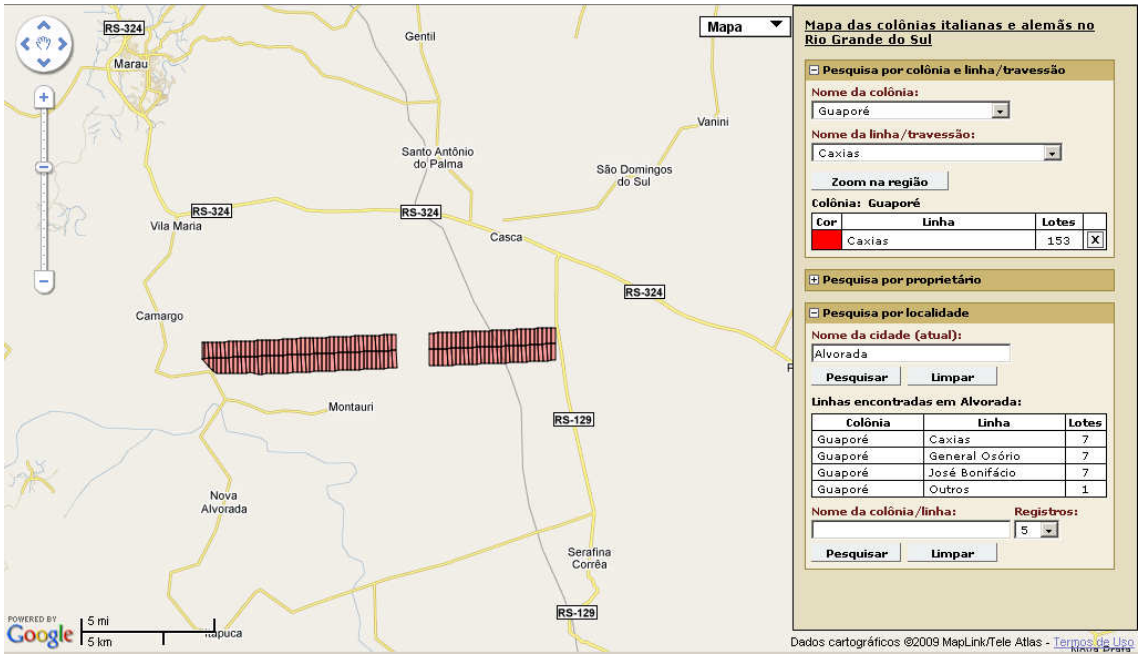

Figura 4.9: Tela de interface do caso de uso “Pesquisa por cidade”

A Figura 4.9 mostra a pós-condição do caso de uso “Pesquisa por cidade”, ou seja, lotes de uma determinada linha colonial exibidos no mapa. Conforme detalhado na Tabela 4.5, o usuário informa o nome da cidade e o sistema exibe uma tabela com as linhas coloniais que tem lotes na cidade informada, exibindo o nome da colônia a qual a linha pertence e o número de lotes que pertencem à cidade pesquisada. Então, o usuário seleciona uma das linhas da tabela e seus lotes são mostrados no mapa, além de ser criada uma entrada para a linha colonial na seção “Pesquisa por colônia e linha”, a principal da aplicação.

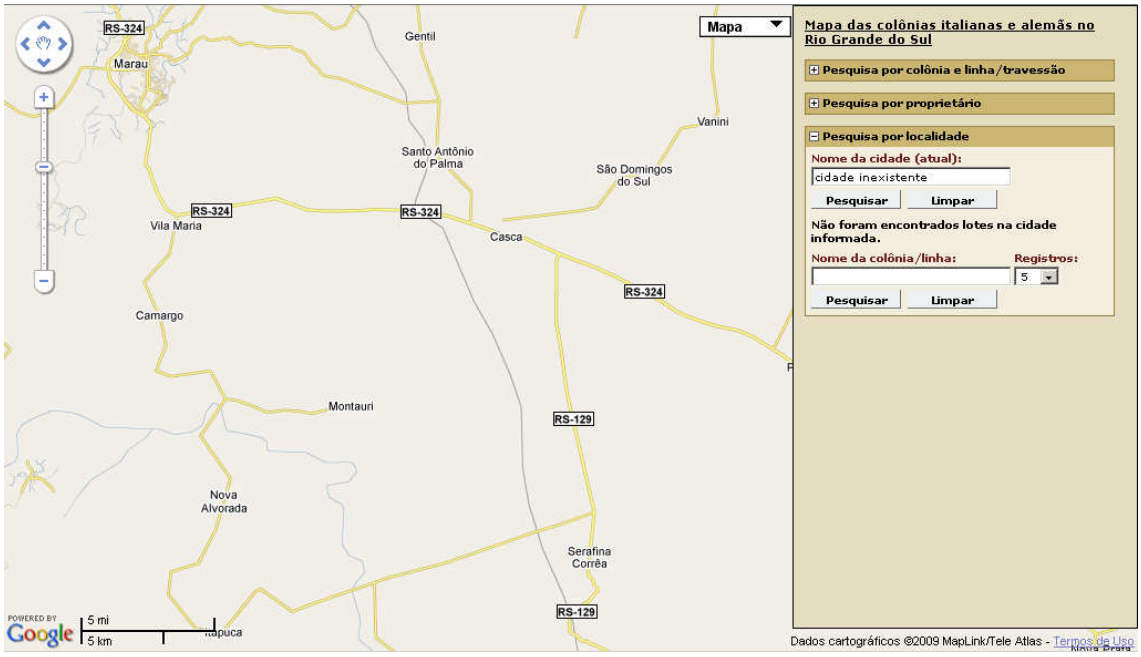

Figura 4.10: Tela de interface do caso de uso “Pesquisa por cidade” em seu fluxo alternativo

A Figura 4.10 mostra o fluxo alternativo do caso de uso “Pesquisa por cidade”, no qual o sistema informa ao usuário que não há registros com o nome informado.

Tabela 4.6: Descrição do caso de uso “Pesquisa nomes de localidades”

UC 06 – Pesquisa nomes de localidades	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Lista com nomes de localidades (colônias e/ou linhas) que iniciem pelo texto informado apresentada.
Descrição:	Usuário pesquisa por lotes informando o nome de colônia e/ou linha e, enquanto isso, é apresentada ao mesmo uma lista com todos os nomes que iniciem pelo texto já informado (precede o caso de uso “Pesquisa livre por colônia e linha”).
Sequência de eventos:	
Ator	**Sistema**
1. Começa a informar o nome da colônia e/ou linha para pesquisa de seus lotes, conforme UC 04.	
	2. Apresenta uma lista com nomes de localidades que iniciem pelo texto já informado pelo usuário.
3. O usuário visualiza a lista de nomes e o caso de uso é encerrado.	
RN 01 – Regra de negócio 01:	
1. A pesquisa por nomes de localidades deve ser iniciada somente a partir da 2ª (segunda) letra informada pelo usuário.	
RN 02 – Regra de negócio 02:	
2. A pesquisa por nomes de localidades deve ser realizada sempre que o usuário informar uma nova letra do nome (a partir da 2ª, conforme RN 01).	

Na Tabela 4.6 é detalhada a pesquisa automática de nomes de localidades, constante na seção “Pesquisa por localidade” do menu da aplicação.

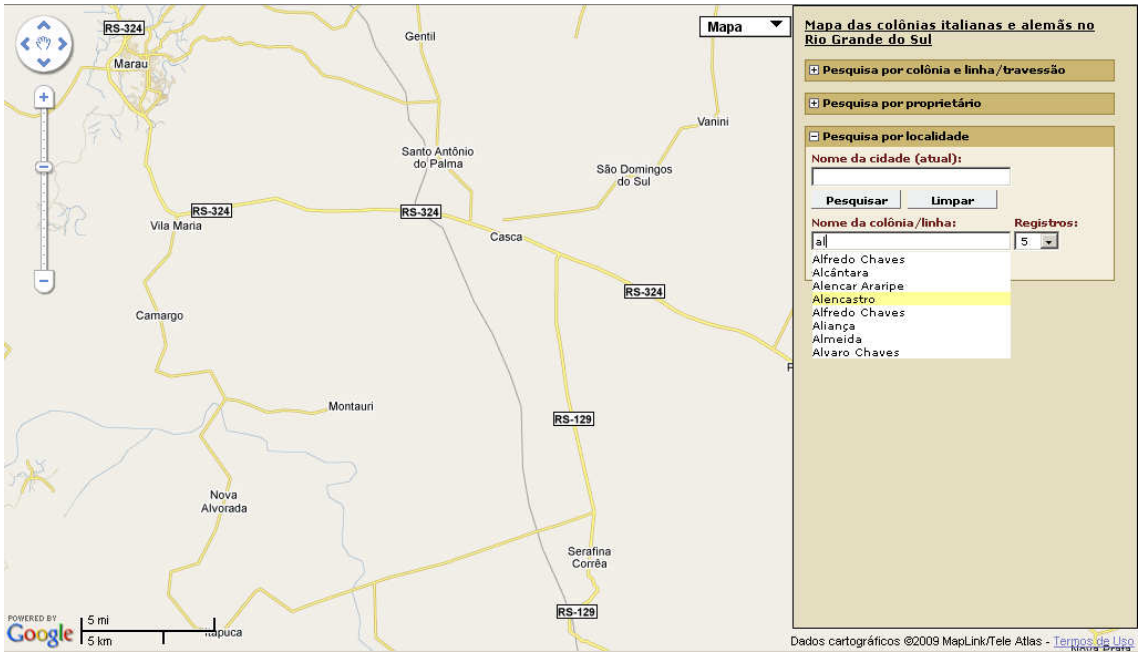

Figura 4.11: Tela de interface do caso de uso “Pesquisa nomes de localidades”

A Figura 4.11 mostra a pós-condição do caso de uso “Pesquisa nomes de localidades”, ou seja, lista de nomes de localidades (colônias e/ou linhas) iniciadas pelo texto informado pelo usuário é apresentada. Sua forma de funcionamento é idêntica aos casos de uso “Pesquisa nomes de proprietários” e “Pesquisa nomes de cidades” descritas anteriormente.

Tabela 4.7: Descrição do caso de uso “Pesquisa livre por colônia e linha”

UC 07 – Pesquisa livre por colônia e linha	
Atores:	Usuário.
Pré-condição:	Nenhuma.
Pós-condição:	Colônia selecionada na caixa de seleção correspondente (no caso da localidade ser colônia) ou lotes da linha selecionada exibidos no mapa (no caso da localidade ser linha).
Descrição:	Usuário pesquisa por lotes informando o nome da colônia e/ou linha. Relação de colônias e/ou linhas com nomes iguais e/ou similares ao informado é apresentada ao usuário. Usuário seleciona uma das localidades da lista: caso seja colônia, a mesma é selecionada; caso seja linha, os lotes da mesma são mostrados no mapa.
Sequência de eventos:	

Ator	Sistema
1. Informa o nome da localidade (colônia e/ou linha) e o número máximo de registros a serem retornados.	
	2. Apresenta uma lista de localidades (colônias e/ou linhas) com nome igual e/ou similar ao informado.
3. Seleciona uma das localidades da lista.	
	4. Se a localidade selecionada for colônia, seleciona a mesma na caixa de seleção correspondente; caso tratar-se de uma linha, exibe os lotes da mesma no mapa.
5. O usuário visualiza a colônia selecionada ou os lotes da linha selecionada e o caso de uso é encerrado.	
Fluxo alternativo 01:	
1.1. Informa um nome de localidade inexistente e sem similaridade com qualquer nome existente.	
	2.1. Informa que não há registros de localidades com o nome informado.

Na Tabela 4.7 é detalhada a pesquisa livre de lotes através de colônia e linha, constante na seção “Pesquisa por localidade” do menu da aplicação.

O caso de uso apresenta um fluxo alternativo idêntico ao caso de uso “Pesquisa por proprietário”, isto é, o sistema apresenta uma mensagem informando que não há registros com o nome informado quando não há pelo menos um nome similar cadastrado.

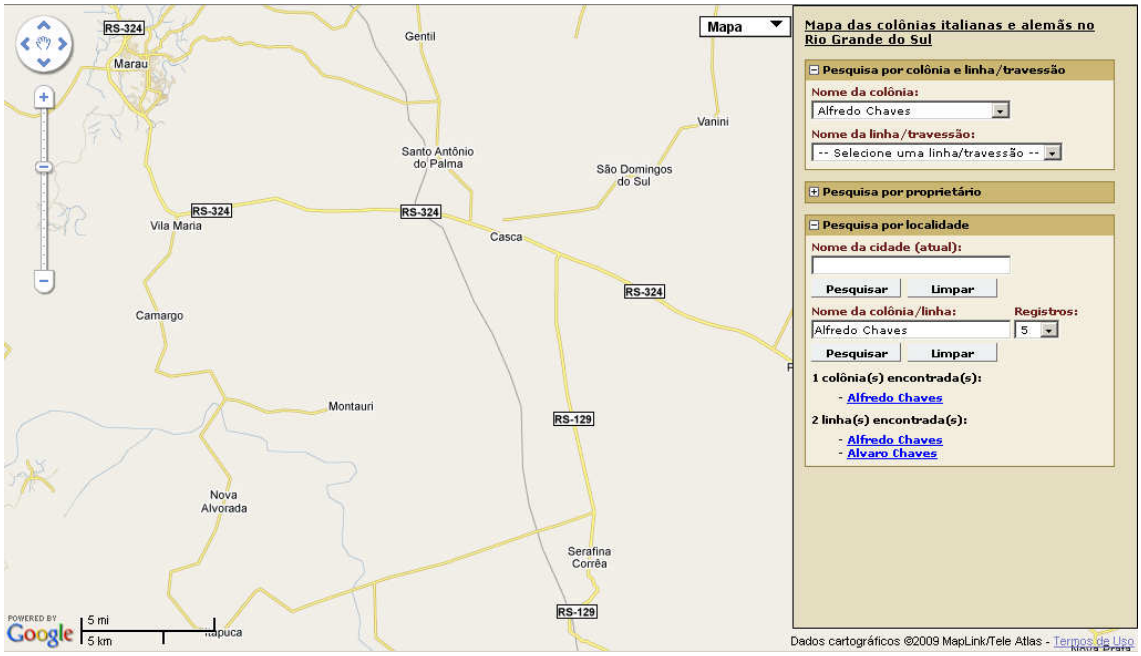

Figura 4.12: Tela de interface do caso de uso “Pesquisa livre por colônia e linha” com seleção de colônia

A Figura 4.12 mostra uma das pós-condições do caso de uso “Pesquisa livre por colônia e linha” – quando o usuário seleciona uma colônia na lista de resultados da pesquisa, a mesma é selecionada na seção “Pesquisa por colônia e linha”, a principal do menu da aplicação. Conforme detalhado na Tabela 4.7, após o usuário ter informado o nome da localidade, o sistema exibe uma lista de localidades (colônias e/ou linhas). Na Figura 4.12, existe uma colônia e uma linha com o nome “Alfredo Chaves” e o usuário escolheu a colônia, sendo que a mesma foi selecionada na caixa de seleção correspondente.

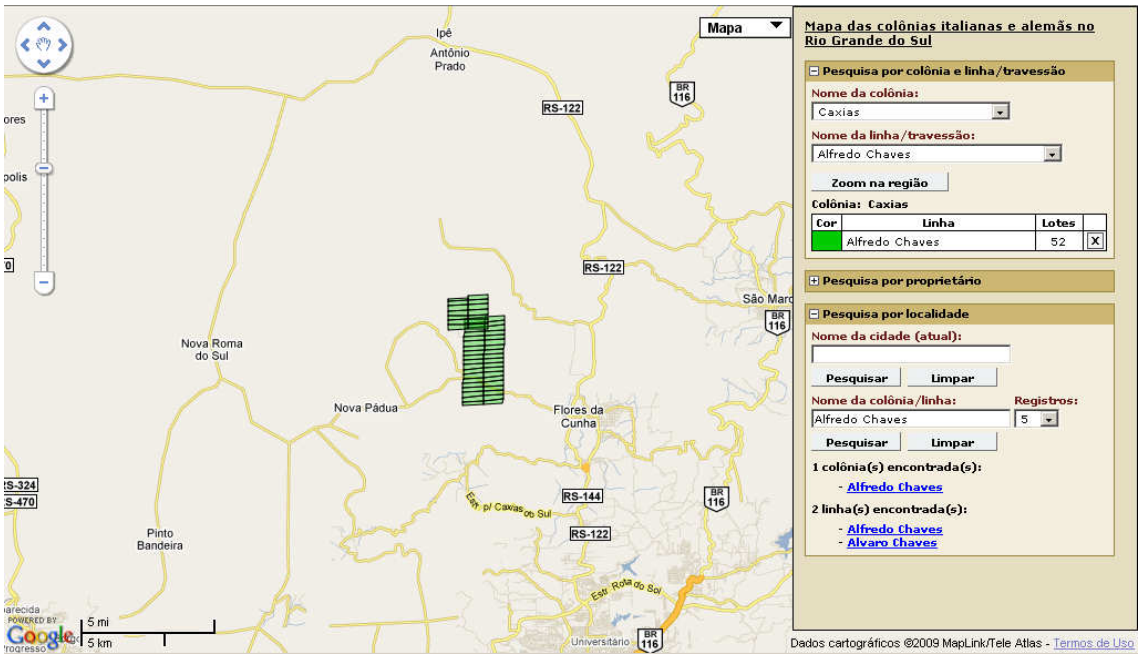

Figura 4.13: Tela de interface do caso de uso “Pesquisa livre por colônia e linha” com seleção de linha

A Figura 4.13 mostra outra das pós-condições do caso de uso “Pesquisa livre por colônia e linha” – quando o usuário seleciona uma linha na lista de resultados da pesquisa, seus lotes são exibidos no mapa, além de ser criada uma entrada para a mesma na seção “Pesquisa por colônia e linha”, a principal da aplicação. Na Figura 4.13, existe uma colônia e uma linha com o nome “Alfredo Chaves” e o usuário escolheu a linha, sendo que seus lotes são mostrados no mapa.

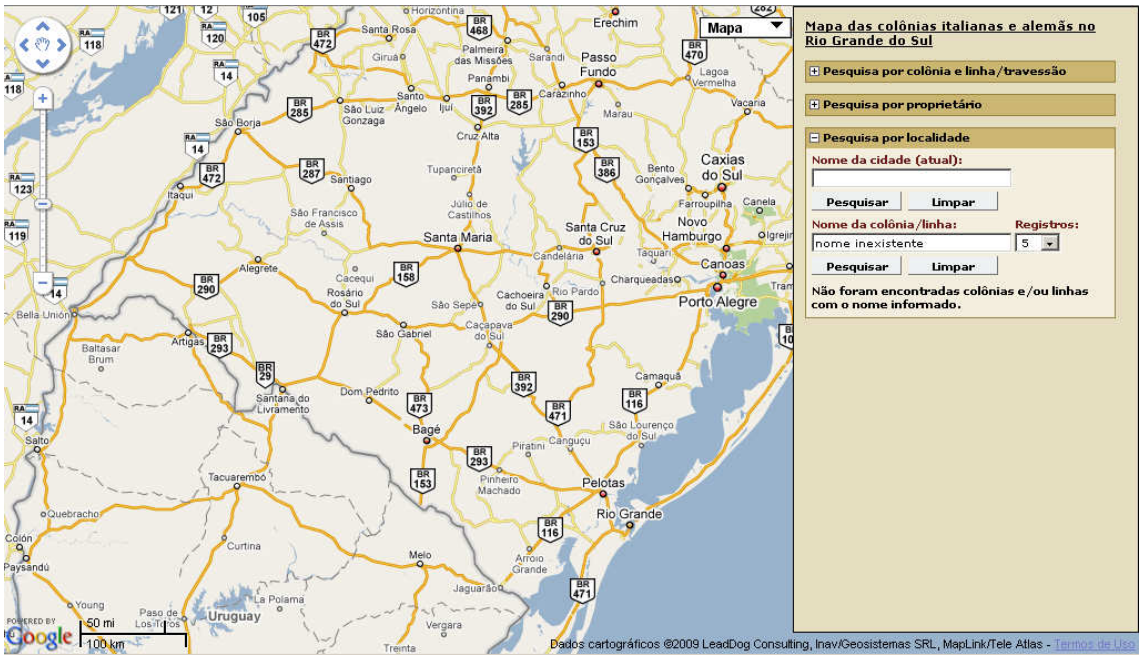

Figura 4.14: Tela de interface do caso de uso “Pesquisa por localidade” em seu fluxo alternativo

A Figura 4.14 mostra o fluxo alternativo do caso de uso “Pesquisa por localidade”, no qual o sistema informa ao usuário que não há registros com o nome informado e nem similares a ele.

Tabela 4.8: Descrição do caso de uso “Pesquisa informações do lote”

UC 08 – Pesquisa informações do lote	
Atores:	Usuário.
Pré-condição:	Lote exibido no mapa.
Pós-condição:	Tabela de informações do lote apresentada.
Descrição:	Usuário pesquisa pelas informações do lote selecionando-o no mapa. Tabela de informações referentes ao lote é apresentada ao usuário.
Sequência de eventos:	
Ator	**Sistema**

1. Seleciona, através do clique do mouse, um lote exibido no mapa.	
	2. Apresenta uma tabela com informações referente ao lote selecionado.
3. O usuário visualiza as informações do lote e o caso de uso é encerrado.	

Na Tabela 4.8 é detalhada a pesquisa por informações de um determinado lote do mapa. O caso de uso é bastante simples, não apresentando fluxos alternativos ou regras de negócios. Entretanto, requer pré-condição, ou seja, que haja ao menos um lote exibido no mapa – pós-condição dos casos de uso “Pesquisa estruturada por colônia e linha”, “Pesquisa por proprietário”, “Pesquisa por cidade” e “Pesquisa livre por colônia e linha”.

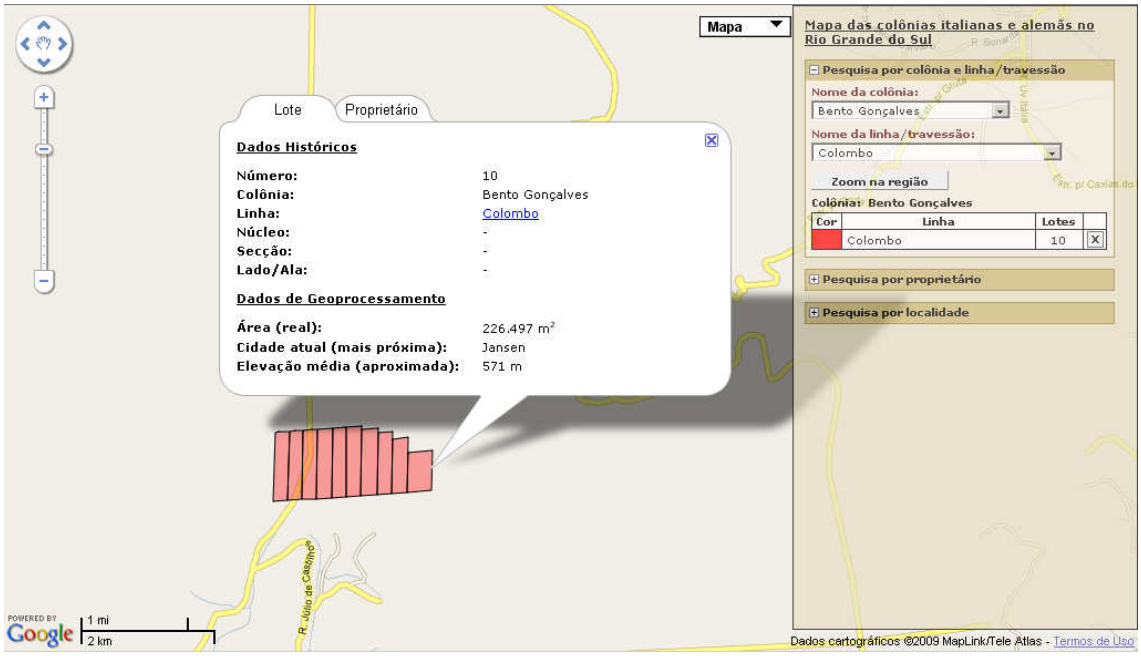

Figura 4.15: Tela de interface do caso de uso “Pesquisa informações do lote”

A Figura 4.15 mostra a pós-condição do caso de uso “Pesquisa informações do lote”, ou seja, tabela de informações do lote selecionado é exibida. Conforme detalhado na Tabela 4.8, o usuário seleciona um lote no mapa clicando sobre o mesmo e o sistema exibe suas informações básicas, dados obtidos através de geoprocessamento e informações sobre seu(s) proprietário(s) (se houver).

5 ARQUITETURA DA APLICAÇÃO

Para demonstrar a arquitetura da solução (isto é, como o sistema provê suas funcionalidades), continuará se utilizando a UML através de seus diagramas de classes e de sequência. Entretanto, antes disso, é muito importante detalhar a estrutura geral da solução proposta, citando as tecnologias envolvidas e como elas se relacionam entre si e, além disso, o modelo de dados utilizado para armazenamento das informações da aplicação.

5.1 Arquitetura geral

A solução proposta apresenta várias tecnologias relacionando-se entre si formando a arquitetura do sistema. A Figura 5.1 demonstra, graficamente, as associações entre tais tecnologias.

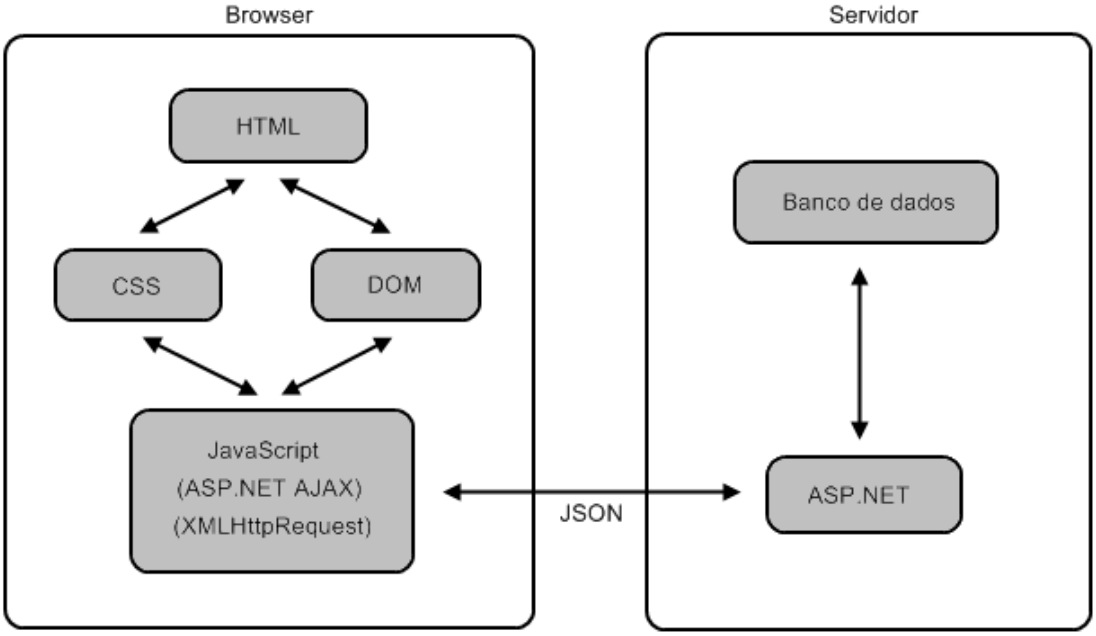

Figura 5.1: Arquitetura geral do sistema

É necessário detalhar cada elemento (e seus relacionamentos) da Figura 5.1 para um bom entendimento da estrutura proposta.

HTML (*Hypertext Markup Language*) – linguagem de marcação estruturada utilizada na aplicação para denotar os elementos da página, tais como caixas de seleção, caixas de texto e títulos. Não configura o visual dos componentes: sua função é definir apenas a estrutura da página.

CSS (*Cascading Style Sheets*) – linguagem de estilos utilizada na aplicação para caracterizar o visual de seus componentes (os quais foram definidos através de HTML), ou seja, a cor de um objeto, seu posicionamento em relação aos demais elementos, entre outras formatações.

DOM (*Document Object Model*) – convenção utilizada por *browsers* para representar uma página HTML como um conjunto de objetos que podem ser acessados programaticamente através de JavaScript.

JavaScript – linguagem de script executada em *browsers* que permite o dinamismo em uma página Web, isto é, permite o acesso e a manipulação programática dos objetos de uma página (através da interface DOM), além de orquestrar as requisições ao servidor (através do objeto XMLHttpRequest).

ASP.NET AJAX – conjunto de extensões do *framework* ASP.NET com a finalidade de fornecer uma interface de programação Ajax mais simplificada e produtiva. Divide-se em dois *frameworks*: do lado cliente (biblioteca JavaScript) e do lado servidor (biblioteca de classes .NET). O *framework* do lado cliente (chamado de Microsoft Ajax Library) é utilizado extensivamente na aplicação, pois, além das facilidades em Ajax, minimiza as discrepâncias de implementação da DOM e do JavaScript em diferentes *browsers*.

XMLHttpRequest – objeto que é utilizado dentro do JavaScript para enviar e receber requisições HTTP diretamente ao/do servidor Web, dando vida ao paradigma de programação Ajax (*Asynchronous JavaScript and XML*). Dessa forma, é possível que o *browser* realize transferências de dados assíncronas com o servidor, sem a necessidade de um *refresh* da página.

JSON (*JavaScript Object Notation*) – formato de intercâmbio de dados utilizado principalmente como alternativa ao XML na transferência de informações em requisições Ajax. Sua maior vantagem é o fato de ser um formato bastante leve e muito semelhante a objetos literais do JavaScript.

ASP.NET – *framework* para o desenvolvimento de aplicações Web utilizado no sistema para construção da página Web e para o acesso aos dados. Para tanto, é usado em conjunto com alguma linguagem de programação .NET – no caso da solução proposta, a linguagem C#.

Banco de dados – coleção de registros logicamente conectados onde todos os dados da aplicação ficam armazenados. A solução proposta oferece suporte para dois DBMSs (*Database Management System*): SQL Server e MySQL.

O usuário, através de seu *browser*, visualiza apenas HTML e estilos dos elementos aplicados através de CSS. Conforme ele vai interagindo com a página, JavaScript é utilizado para realizar todo o tipo de procedimento, desde mostrar ou esconder uma seção do menu até pesquisar informações no banco de dados (através de requisições Ajax). Para manipular os elementos da página, JavaScript utiliza a DOM e, dependendo do caso, aplica estilos CSS dinamicamente. Para a interação com o servidor, a biblioteca ASP.NET AJAX encapsula as funções da API XMLHttpRequest e trata do envio e do recebimento das requisições Ajax realizadas. Apesar do nome, a XMLHttpRequest não trabalha apenas com XML – no caso do ASP.NET AJAX, os dados entre cliente e servidor são trafegados através de JSON. Para tanto, existe a serialização e a desserialização automática de JSON para objetos válidos em JavaScript (lado cliente) e C# (lado servidor), tudo realizado automaticamente pelo *framework* do ASP.NET

AJAX. No lado do servidor, o ASP.NET, através de classes especializadas, realiza as devidas requisições ao banco de dados, o qual fornece as informações necessárias para cada requisição do cliente. As chamadas entre cliente e servidor serão vistas com mais detalhe quando forem apresentados os diagramas de sequência dos casos de uso da solução proposta na seção 5.4.

O modelo de desenvolvimento adotado na solução proposta é chamado de Modelo de desenvolvimento centrado no (lado) cliente (tradução do termo, em inglês, *client-centric development model*). Tal modelo se caracteriza pelo fato da lógica de apresentação e de negócios da aplicação estar concentrada no lado cliente, isto é, controlada através de JavaScript. As interações entre cliente e servidor limitam-se apenas ao acesso aos dados estritamente necessários em cada momento, tal como buscar a lista de linhas de uma dada colônia ou pesquisar as informações de um dado lote no momento que o usuário assim o desejar. Quando tais dados chegam ao lado cliente, o código JavaScript atualiza apenas as porções da página que dizem respeito àquelas informações. Este modelo tem como princípio oferecer uma interatividade bem maior entre o usuário e a aplicação rodando em seu *browser*, resultando em uma experiência mais rica e intuitiva (GALLO, 2007).

5.2 Modelo de dados

A modelagem do banco de dados da aplicação foi baseada na modelagem previamente existente em formato .dbf (DBMS dBase) fornecida pelo Dr. Otávio Licht. Tal formato foi utilizado tendo em vista que o software ArcGIS, onde a plotagem inicial dos dados georreferenciados foi realizada, armazena todas as suas informações (com exceção das coordenadas dos lotes coloniais) em arquivos .dbf.

A Figura 5.2 mostra a modelagem ER (entidade-relacionamento) do banco de dados da aplicação.

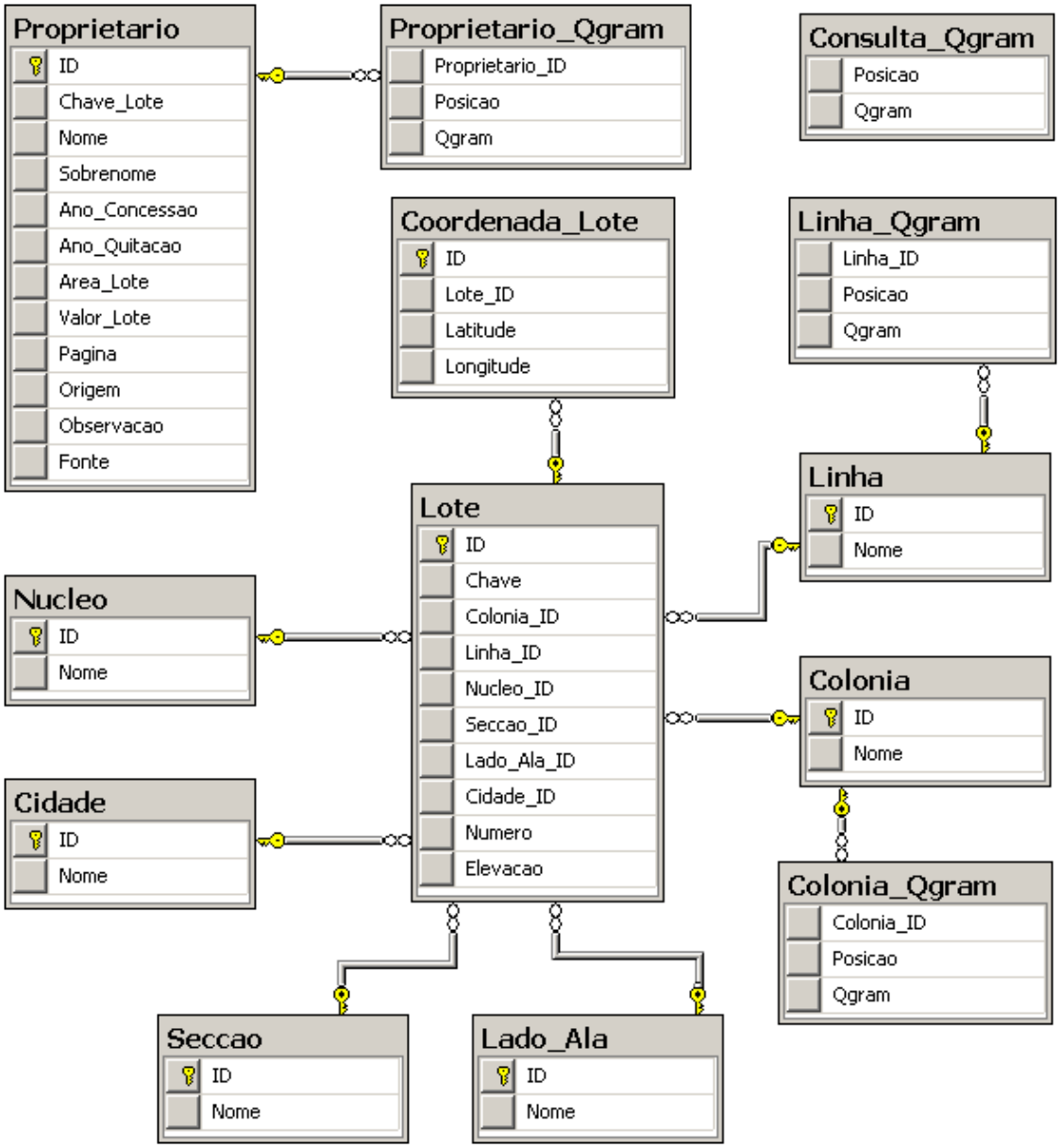

Figura 5.2: Modelo ER da aplicação

A principal tabela de dados da aplicação é Lote, a qual contém as informações básicas de um dado lote colonial, tal como a identificação de sua colônia, sua linha e um campo de chave que serve para manter a relação com seu(s) proprietário(s). Tendo em vista a existência de vários núcleos, cidades, secções, lados/alas, colônias e linhas compartilhados entre vários lotes, há uma tabela de dados para cada um desses elementos, com uma chave estrangeira para cada um. Com exceção das informações sobre cidade e elevação média (as quais são provindas dos dados obtidos através do serviço GeoNames), todos os dados foram construídos através da tabela .dbf já existente.

A tabela Coordenada_Lote mantém uma relação de muitos-para-um com a tabela Lote pois armazena todas as coordenadas da borda de um lote. Dessa forma, por exemplo, se um lote tiver um formato estritamente retangular, ele terá associação com, pelo menos, quatro registros da tabela Coordenada_Lote (um para cada esquina do lote, no mínimo).

A tabela Proprietario não mantém informações referentes apenas ao dono do lote, e sim também sobre os dados de concessão do mesmo. Devido à fraca integridade dos dados constantes na tabela em dBase original, não foi possível mapear cada proprietário individualmente e manter uma tabela de dados à parte para as informações de cada pessoa e outra apenas para os dados da concessão. Dessa forma, foi utilizada a estrutura já existente. O campo Chave_Lote referencia o campo Chave da tabela Lote, associando um proprietário a seu lote. Contudo, não se trata de uma chave estrangeira, uma vez que, nos dados iniciais das tabelas em dBase, existiam proprietários sem relação com lote e vice-versa.

As tabelas com sufixo _Qgram armazenam os dados referentes aos *q-grams* detalhados no capítulo 3. Dessa forma, por exemplo, Linha_Qgram mantém os *q-grams* de todos os nomes de linhas existentes na tabela Linha. A única exceção é a tabela Consulta_Qgram a qual, conforme já explicado no capítulo 3, é populada dinamicamente com os *q-grams* da string de consulta realizada.

5.3 Diagramas de classes

Visando obter uma melhor estruturação da solução proposta, o código-fonte no lado servidor foi organizado em três tipos de classes: negócio, dados e úteis. Dessa forma, a página ASP.NET (no servidor) não contém lógica alguma, delegando tais tarefas às referidas classes. É importante uma breve explicação sobre a função de cada categoria de classe.

Negócio – categoria que também poderia ser chamada de “Transporte”, espelha a modelagem das tabelas do banco de dados. É utilizada na transferência dos dados entre as classes de dados e a página ASP.NET e entre a última e a página no lado cliente (JavaScript).

Dados – realiza a tarefa de obter as informações solicitadas junto ao banco de dados. Para tanto, faz uso das classes de negócio, criando um objeto específico a partir das informações de um registro de uma dada tabela do banco, e das classes úteis, para realizar procedimentos comuns.

Úteis – categoria que também poderia ser chamada de “Suporte”, realiza procedimentos repetitivos para a classe de dados e outras funções que não se encaixam nas atribuições dos demais tipos de classes.

Através da separação das classes em categorias distintas, a manutenção do código fica bem mais facilitada, uma vez que os erros podem ser descobertos mais facilmente. Além disso, alterações pontuais internas às classes podem ser realizadas de maneira a não afetar as interações com os demais tipos de classe, desde que se mantenha uma interface consistente entre eles.

A Figura 5.3 mostra a estrutura das classes de negócio da aplicação.

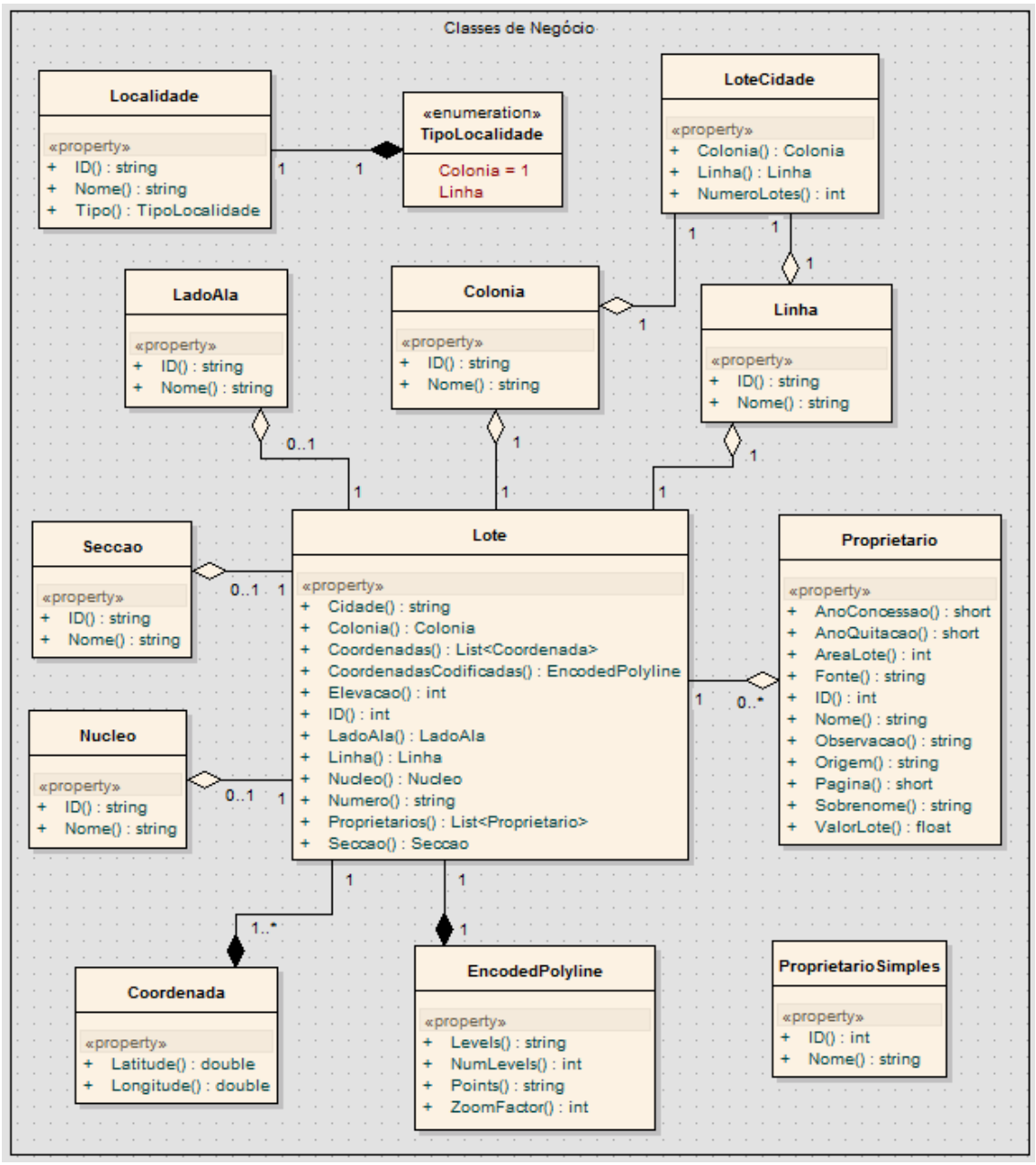

Figura 5.3: Diagrama de classes: classes de negócio

A modelagem das classes de negócio, conforme já mencionado, é espelhada na modelagem do banco de dados. Com exceção das tabelas de *q-grams* (as quais são utilizadas somente nas buscas por similaridade) e Cidade (a qual foi substituída pela classe LoteCidade, explicada adiante), todas as demais tabelas do banco foram mapeadas para classes. Entretanto, respeitando a programação orientada a objetos, ao invés de haver chaves estrangeiras, as associações são realizadas através de referências a objetos (por exemplo, Lote contém uma instância da classe Colonia). Tais associações dividem-se em composição (o objeto referenciado não tem sentido sem o objeto que o referencia), caracterizada pela ligação com o losango preenchido, e agregação (o objeto referenciado é independente do objeto que o referencia, ou seja, mesmo que o último seja destruído, o primeiro continua fazendo sentido), caracterizada pela ligação com o losango vazio.

Além das entidades existentes no banco de dados, foram criadas outras classes auxiliares. Uma breve explicação sobre tais elementos faz-se importante.

ProprietarioSimples – é utilizada no caso de uso “Pesquisa por proprietário” quando o sistema exibe uma lista de proprietários com nome igual e/ou similar ao informado pelo usuário. Tal classe é utilizada pois a classe Proprietario tem atributos que não são necessários pelo caso de uso e, visando buscar um maior rendimento na pesquisa ao banco e na transferência de dados entre servidor e cliente, tal decisão de projeto foi tomada.

LoteCidade – substitui a tabela Cidade, uma vez que não é apresentada ao usuário uma lista com os nomes de todas as cidades cadastradas. É utilizada no caso de uso “Pesquisa por cidade” após o usuário ter informado o nome da cidade desejada. Então, o sistema exibe uma tabela com as linhas coloniais que contenham lotes na cidade, mostrando o nome da colônia e o número de lotes naquela localidade.

Localidade – é utilizada no caso de uso “Pesquisa por localidade” quando o sistema exibe uma lista de colônias e/ou linhas com nome igual e/ou similar ao informado pelo usuário. Conta com um campo do tipo enumeração, TipoLocalidade, para caracterizar o objeto como representando uma colônia ou linha.

EncodedPolyline – é utilizada apenas para fins de melhoria de desempenho na “plotagem” (desenho) dos lotes no Google Maps. Apesar de a classe Lote manter uma lista de objetos da classe Coordenada, os polígonos referentes aos lotes são desenhados no mapa através de um objeto EncodedPolyline, o qual contém todas as coordenadas do lote em um formato codificado (uma breve explicação sobre tal codificação será apresentada junto à classe PolylineEncoder das classes úteis). A classe Coordenada só é utilizada no código JavaScript para determinar onde será realizado o *zoom* sobre uma determinada região selecionada pelo usuário.

A Figura 5.4 mostra a estrutura das classes de dados da aplicação.

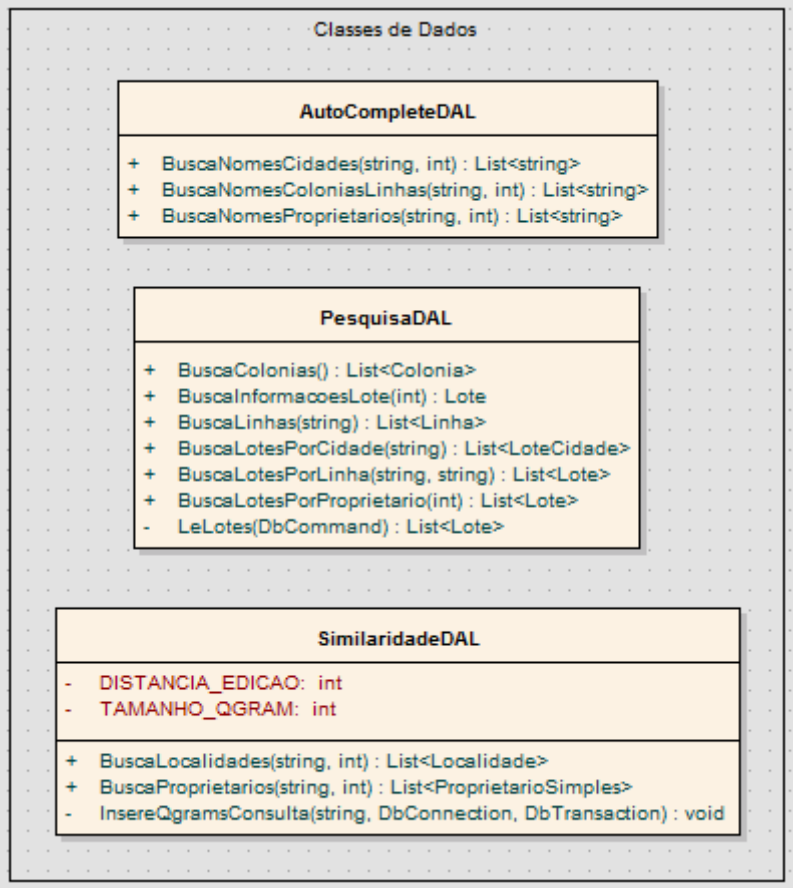

Figura 5.4: Diagrama de classes: classes de dados

As classes de acesso a dados foram categorizadas conforme o tipo de pesquisa que realizam.

A classe AutoCompleteDAL é responsável pelas pesquisas realizadas nos casos de uso “Pesquisa nomes de proprietários”, “Pesquisa nomes de cidades” e “Pesquisa nomes de localidades” (funções de auto-completar). Seus métodos recebem como parâmetros o texto já informado e o número máximo de registros a serem retornados.

A classe PesquisaDAL é responsável pelas pesquisas gerais da aplicação. Todos os métodos que não se encaixam em uma categoria específica pertencem a esta classe.

A classe SimilaridadeDAL é responsável pelas pesquisas realizadas nos casos de uso “Pesquisa por proprietário” e “Pesquisa livre por colônia e linha” (buscas por similaridade de nomes). Assim como na classe AutoCompleteDAL, seus métodos públicos recebem como parâmetros o texto informado e o número máximo de registros (esse último argumento escolhido pelo usuário).

A Figura 5.5 mostra a estrutura das classes úteis da aplicação.

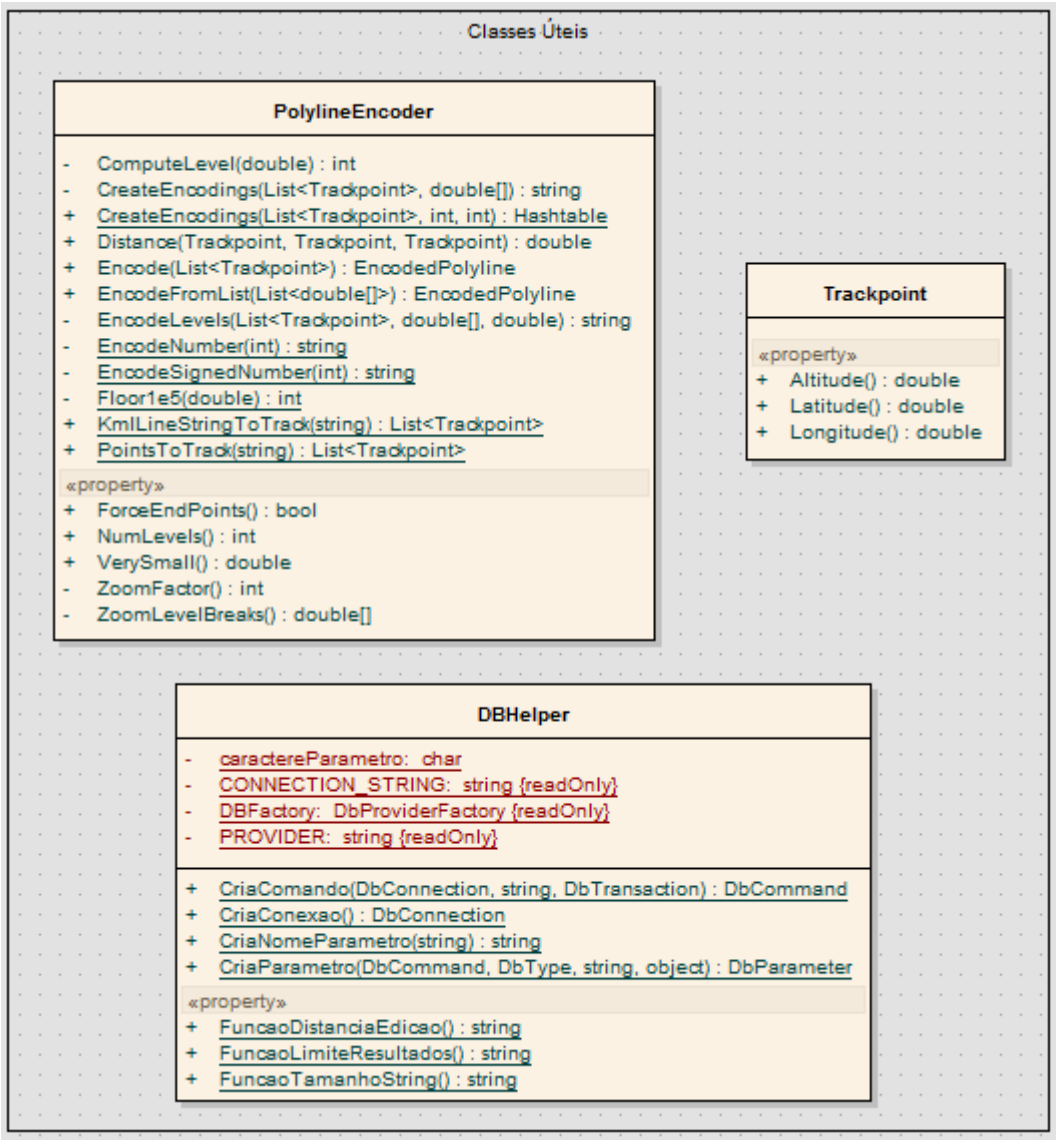

Figura 5.5: Diagrama de classes: classes úteis

A classe DBHelper é utilizada exclusivamente pelas classes de dados para realizar alguns procedimentos comuns, tal como criar uma conexão com o banco de dados, e fornecer uma forma de abstração independente do banco de dados utilizado (SQLServer ou MySQL). Dessa forma, por exemplo, a classe de dados não tem o dever de saber qual é a função utilizada pelo DBMS sendo utilizado para determinar o tamanho de uma string: basta ler o valor da propriedade FuncaoTamanhoString da classe DBHelper.

As classes PolylineEncoder e Trackpoint, assim como a classe de negócios EncodedPolyline, foram obtidas no site http://www.svennerberg.com (acesso em agosto de 2009) e estão sob a licença GPL (*General Public License*), ou seja, podem ser usadas e alteradas livremente (contudo, na aplicação estão sendo utilizadas exatamente conforme foram projetadas). A finalidade de tais entidades é melhorar o desempenho da plotagem de polígonos (os quais representam os lotes coloniais na solução proposta) no Google Maps. O algoritmo que codifica as coordenadas remove algumas que são irrelevantes e também controla o número de pontos mostrados no mapa quando o usuário altera o zoom. Dessa forma, mais lotes podem ser exibidos no mapa ao mesmo tempo com uma menor degradação da performance da aplicação.

5.4 Diagramas de sequência

Os diagramas de sequência da UML são importantes para demonstrar a interação entre os objetos da aplicação, dando ênfase à sequência com que tais relacionamentos acontecem. A partir deles, a arquitetura geral tende a ficar mais clara, uma vez que mostram todas as requisições (e respectivas respostas) realizadas entre cliente (JavaScript) e servidor (ASP.NET).

Normalmente, é construído um diagrama de sequência para cada caso de uso (quando este é muito grande, pode ser decomposto em mais de um diagrama) e assim foi feito para a solução proposta. A Figura 5.6 mostra o diagrama de sequência para o caso de uso “Pesquisa estruturada por colônia e linha”.

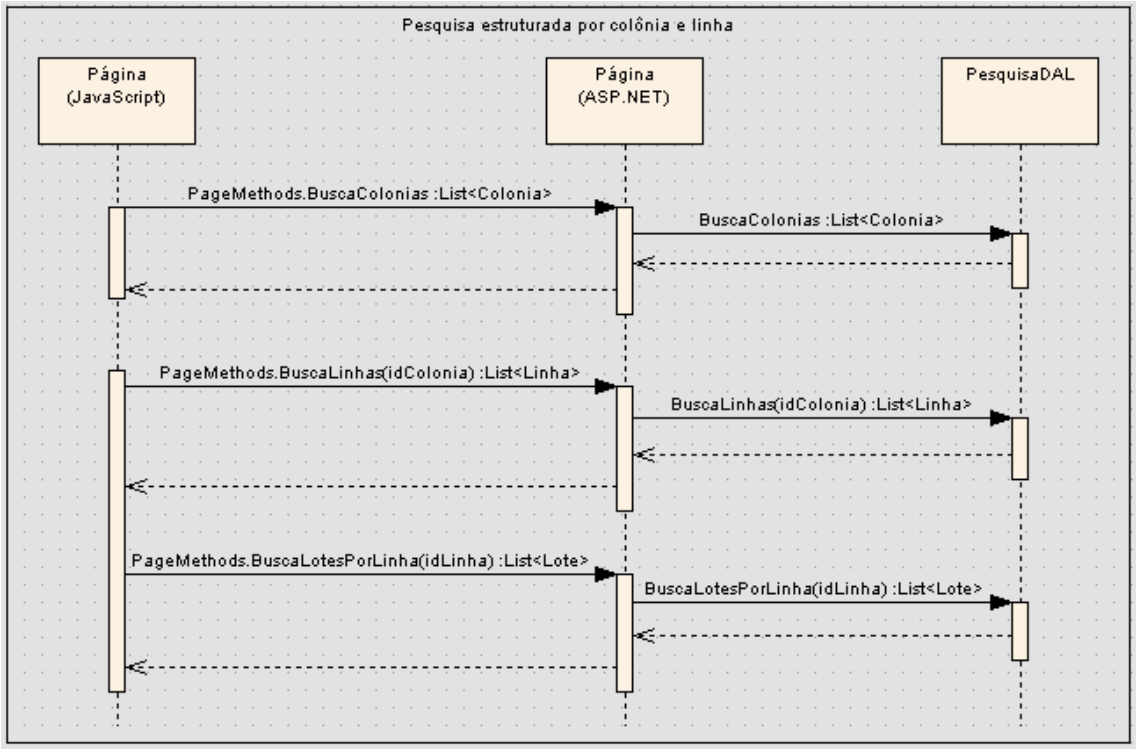

Figura 5.6: Diagrama de sequência do caso de uso “Pesquisa estruturada por colônia e linha”

Primeiramente, algumas considerações devem ser feitas. Conforme já mencionado anteriormente, o *framework* ASP.NET AJAX, através da biblioteca Microsoft Ajax Library para JavaScript, abstrai a funcionalidade do objeto XMLHttpRequest a fim de facilitar a realização de requisições Web ao servidor. Na programação Ajax, o método comum (sem auxílio de bibliotecas) requer que o programador primeiro verifique o *browser* que está acessando a página: dependendo do navegador, o objeto XMLHttpRequest é criado de uma forma diferente. Após isso, deve definir uma série de parâmetros e realizar algumas checagens, algo que se torna tedioso e propenso a erros. Ao encapsular as funcionalidades do XMLHttpRequest (através de certos *proxies* criados automaticamente entre o cliente e o servidor), o ASP.NET AJAX permite que um método definido no servidor (na página ASP.NET) seja invocado pelo código JavaScript através de seu nome, exatamente como se ambos estivessem no lado servidor. Para tanto, basta prefixar o nome do método com “PageMethods.” e passar como parâmetros os tipos de dados requisitados (os quais são serializados e desserializados em JSON, conforme já mencionado). Os tipos de dados entre cliente e

servidor são convertidos para tipos válidos em ambos os lados. Por exemplo, não existe um tipo *List* em JavaScript (como existe em C#) mas, ao serializá-lo para JSON, o ASP.NET AJAX converte o tipo para um *array* válido em JavaScript (e o contrário também é verdadeiro). Dessa forma, a integração entre cliente e servidor ocorre de forma transparente ao programador.

A Figura 5.6 detalha a sequência de requisições realizadas ao servidor no caso de uso “Pesquisa estruturada por colônia e linha”. A lógica é bastante simples: logo que a aplicação é iniciada, a lista de colônias é pesquisada no banco de dados e armazenada no lado cliente. Após isso, assim que o usuário selecionar uma das colônias da caixa de seleção correspondente, suas linhas são automaticamente pesquisadas. Com isso, basta o usuário selecionar uma das linhas coloniais e seus lotes são pesquisados e posteriormente exibidos no mapa. Conforme já explicado na seção 5.3, a página ASP.NET simplesmente repassa as requisições para a classe de dados responsável, a qual, nesse caso, é PesquisaDAL.

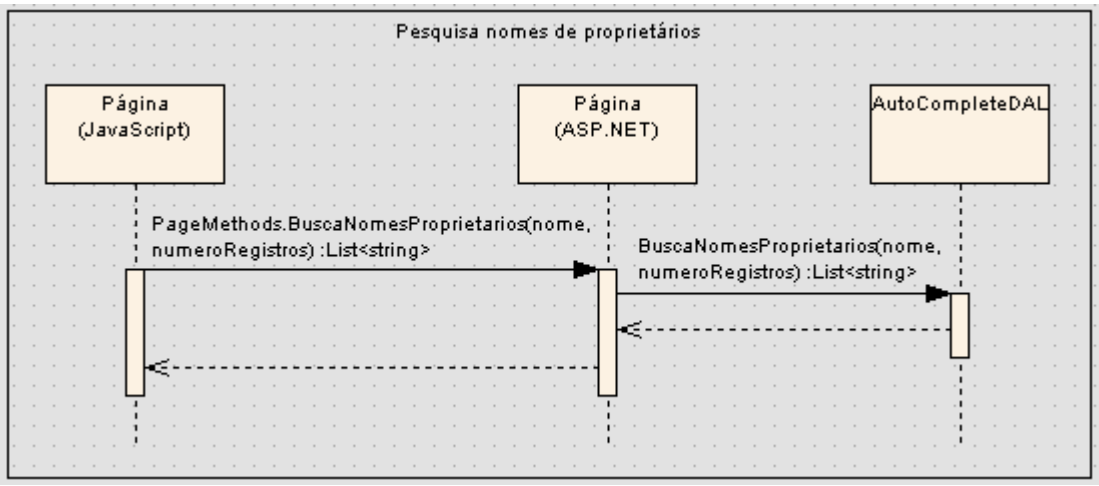

Figura 5.7: Diagrama de sequência do caso de uso “Pesquisa nomes de proprietários”

A Figura 5.7 detalha a sequência de interações entre cliente e servidor no caso de uso “Pesquisa nomes de proprietários”. Apenas uma requisição é realizada a cada nova letra digitada pelo usuário na caixa de texto correspondente.

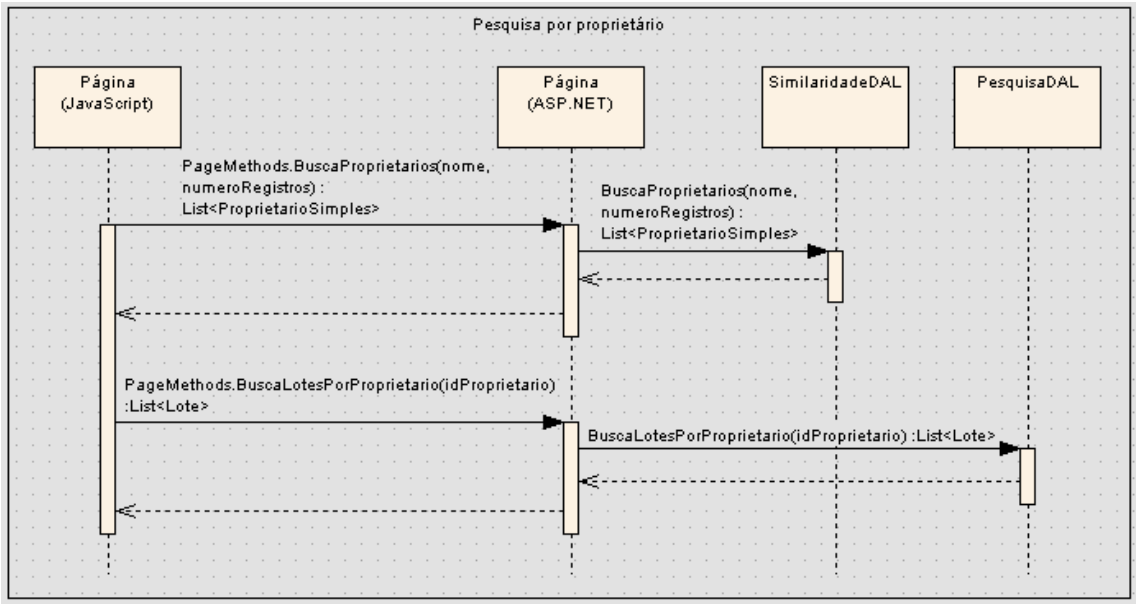

Figura 5.8: Diagrama de sequência do caso de uso “Pesquisa por proprietário”

A Figura 5.8 detalha a sequência de requisições realizadas entre cliente e servidor no caso de uso “Pesquisa por proprietário”. Nesse caso, duas classes de dados estão envolvidas, uma vez que primeiramente é buscada uma lista de proprietários com nome igual e/ou similar ao informado pelo usuário (SimilaridadeDAL) e, após o usuário selecionar o proprietário, seus lotes são pesquisados (PesquisaDAL).

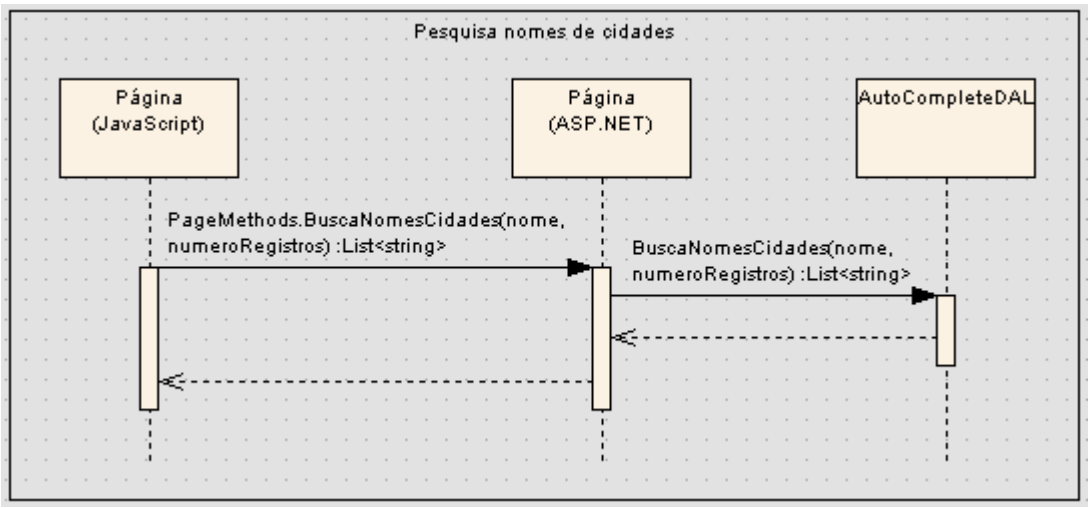

Figura 5.9: Diagrama de sequência do caso de uso “Pesquisa nomes de cidades”

A Figura 5.9 detalha a sequência de interações entre cliente e servidor no caso de uso “Pesquisa nomes de cidades”. Afora o nome dos métodos chamados, o fluxo é idêntico ao do caso de uso “Pesquisa nomes de proprietários”.

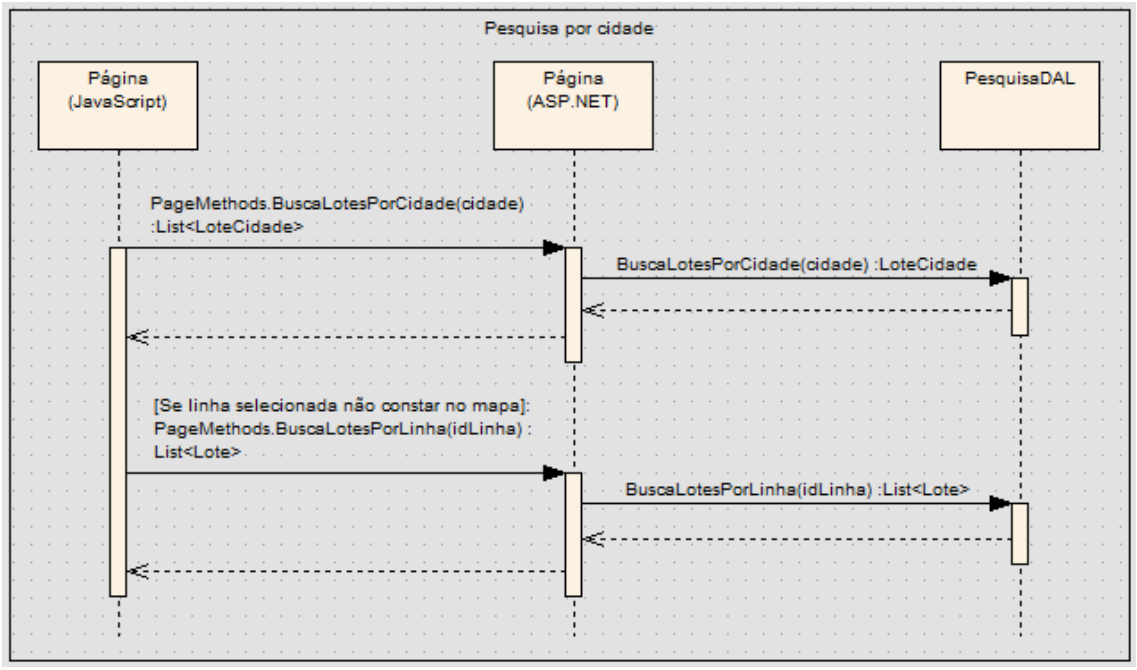

Figura 5.10: Diagrama de sequência do caso de uso “Pesquisa por cidade”

A Figura 5.10 detalha a sequência de requisições realizadas ao servidor no caso de uso “Pesquisa por cidade”. O diagrama mostra que a segunda chamada ao ASP.NET só é realizada através de uma condição: se a linha selecionada pelo usuário já não constar no mapa. Tal condição também existe nos demais casos de uso de pesquisa por lotes, mas foram omitidos por simplificação. Se os lotes selecionados já estão sendo exibidos no mapa, não se faz necessária uma nova requisição ao banco de dados.

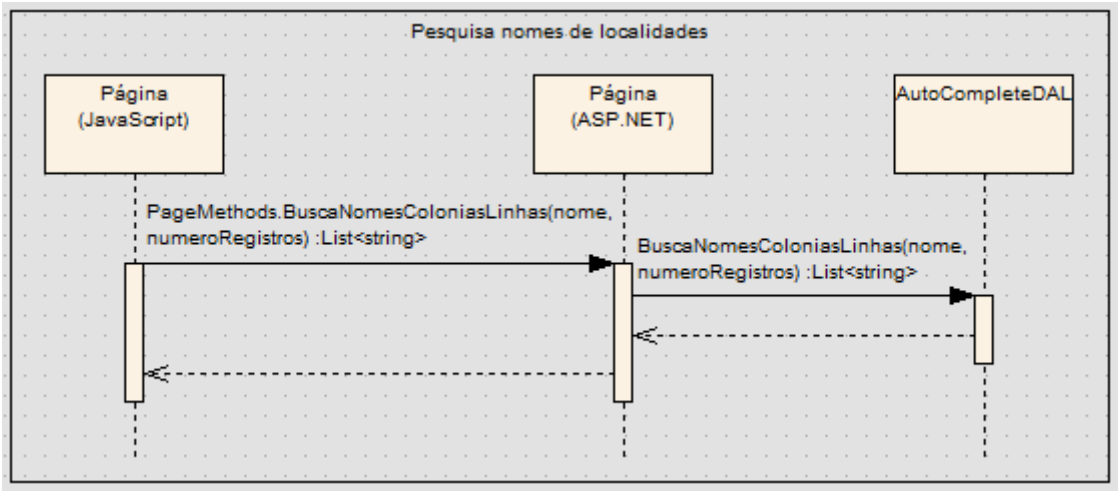

Figura 5.11: Diagrama de sequência do caso de uso “Pesquisa nomes de localidades”

A Figura 5.11 detalha a sequência de interações entre cliente e servidor no caso de uso “Pesquisa nomes de localidades”. Com exceção dos métodos chamados, o fluxo é idêntico ao dos casos de uso “Pesquisa nomes de proprietários” e “Pesquisa nomes de cidades”.

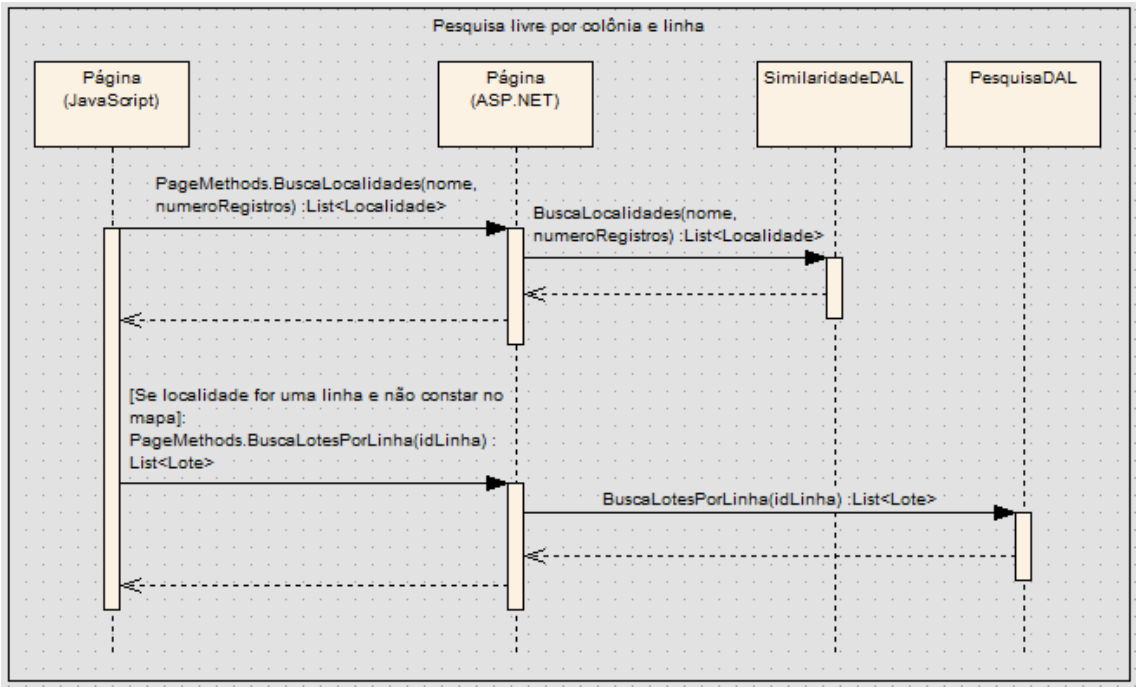

Figura 5.12: Diagrama de sequência do caso de uso “Pesquisa livre por colônia e linha”

A Figura 5.12 detalha a sequência de requisições realizadas ao servidor no caso de uso “Pesquisa livre por colônia e linha”. O diagrama apresenta uma condição, assim como na Figura 5.10, para buscar os lotes no banco de dados. Para tanto, o usuário deve ter selecionado uma linha colonial na lista de localidades retornadas, uma vez que, quando ele escolhe uma colônia, esta é apenas selecionada na caixa de seleção correspondente.

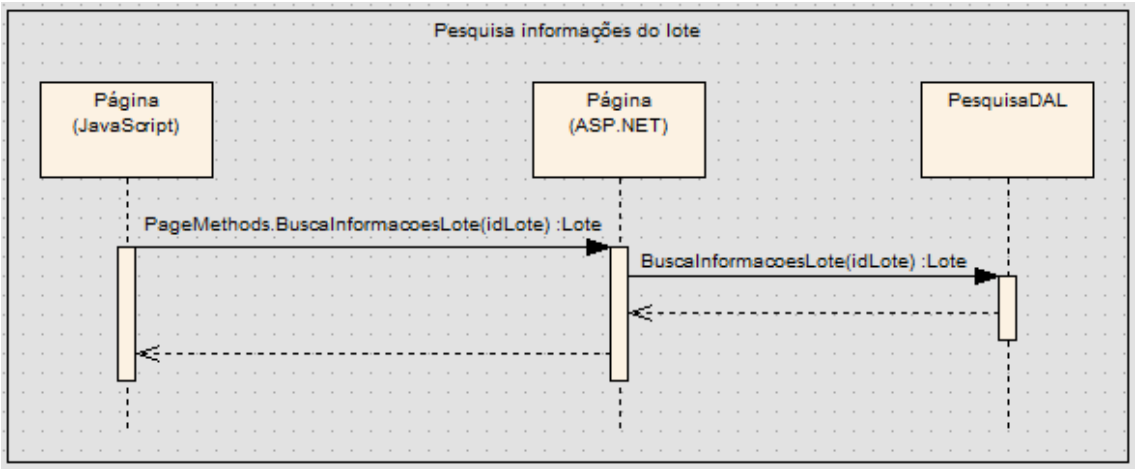

Figura 5.13: Diagrama de sequência do caso de uso “Pesquisa informações do lote”

A Figura 5.13 detalha a sequência de interações entre cliente e servidor no caso de uso “Pesquisa informações do lote”. Os dados do lote selecionado são simplesmente pesquisados no banco de dados através do identificador único do lote.

6 CONCLUSÃO

Este trabalho apresentou um sistema para busca e exibição de dados georreferenciados referentes à colonização do Estado do Rio Grande do Sul no final do século XIX. Para tanto, fez uso de diversas tecnologias diferentes interagindo entre si, além de dois componentes básicos: um serviço Web de mapeamento e uma técnica de busca de informações por similaridade.

A aplicação fez uso do Google Maps como ferramenta de visualização das regiões coloniais em um mapa. Tal serviço oferece, de forma gratuita, uma API que permite a inclusão de suas funcionalidades em qualquer página Web. A partir delas, é possível oferecer ao usuário final uma interação intuitiva com os lotes coloniais, uma vez que qualquer informação geográfica pode ser representada nos mapas disponíveis.

Para facilitar na pesquisa pelas informações coloniais, tanto geográficas quanto referentes aos proprietários de lotes, a técnica apresentada em (GRAVANO, 2005) foi utilizada para a busca por similaridade. Tendo em vista que informações históricas são suscetíveis a erros tipográficos e grafias alternativas para nomes de lugares e pessoas, foi implementada uma solução para que o usuário não tivesse que saber a grafia exata de um determinado dado de pesquisa. Dessa forma, pesquisas críticas do sistema – por localidades coloniais e por proprietário – são realizadas a partir de aproximações com a informação procurada pelo usuário.

Uma ideia interessante a ser desenvolvida em trabalhos futuros seria a possibilidade de a aplicação oferecer, aos usuários interessados, a oportunidade deles mesmos cederem informações a respeito dos lotes coloniais, nos mesmos moldes apresentados pelo Wikipedia (http://www.wikipedia.org), uma enciclopédia livre e gratuita na Web, e pelo próprio GeoNames, serviço utilizado na solução. Qualquer tipo de nova informação poderia ser aceito, desde dados puramente genealógicos sobre os proprietários até a inclusão de novos lotes coloniais no mapa. Através de um registro de usuários cadastrados, seria possível saber a fonte dos dados, podendo ser formada até mesmo uma pseudo rede social, onde seus integrantes poderiam trocar informações entre si, criar grupos de pesquisa, entre outras opções. Dessa forma, o site seria uma fonte ainda mais rica de dados históricos.

REFERÊNCIAS

GOOGLE MAPS. Disponível em: http://maps.google.com. Acesso em: nov. 2009.

BING MAPS. Disponível em: http://www.bing.com/maps. Acesso em: nov. 2009.

YAHOO MAPS. Disponível em: http://maps.yahoo.com. Acesso em: nov. 2009.

GEONAMES. Disponível em: http://www.geonames.org. Acesso em: nov. 2009.

FOWLER, Martin. **UML Distilled: A Brief Guide to the Standard Object Modeling Language**. 3rd ed. Boston: Addison-Wesley Professional, 2005.

GALLO, Alessandro; BARKOL, David; RAMA, Krishna V. **ASP.NET AJAX in Action**. 1st ed. Greenwich: Manning Publications, 2007.

GRAVANO, Luis; et. al. **Approximate String Joins in a Database (Almost) for Free.** Proceedings of the 27th International Conference on Very Large Data Bases. San Francisco, CA, EUA. ACM 2001. p. 491-500.

UKKONEN, Esko. **Approximate string matching with q-grams and maximal matches**. Theoretical Computer Science, 92(1):191-211. 1992.

ULLMANN, Julian R. **A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words.** The Computer Journal, 20(2):141-147. 1977.