ANTÔNIO AUGUSTO ALENCAR DE MORAIS

PROTÓTIPO DE UMA APLICAÇÃO DE VISUALIZAÇÃO DA INFORMAÇÃO EM TEMPO REAL BASEADA EM GEOLOCALIZAÇÃO

Lavras – MG
2013

ANTÔNIO AUGUSTO ALENCAR DE MORAIS

PROTÓTIPO DE UMA APLICAÇÃO DE VISUALIZAÇÃO DA INFORMAÇÃO EM TEMPO REAL BASEADA EM GEOLOCALIZAÇÃO

Monografia apresentada ao Colegiado do curso de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso para a obtenção do título de Bacharel em Ciência da Computação.

Orientador

Dra. Ana Paula Piovesan Melchiori

Lavras – MG

2013

ANTÔNIO AUGUSTO ALENCAR DE MORAIS

PROTÓTIPO DE UMA APLICAÇÃO DE VISUALIZAÇÃO DA INFORMAÇÃO EM TEMPO REAL BASEADA EM GEOLOCALIZAÇÃO

Monografia apresentada ao Colegiado do curso de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 29 de agosto de 2013.

Dr. Tales Heimfarth	DCC/UFLA
M.Sc. André Grützmann	DCC/UFLA

Dra. Ana Paula Piovesan Melchiori
Orientador

Lavras – MG

2013

À minha família.

DEDICO

AGRADECIMENTOS

Agradeço primeiramente a Deus por mais uma conquista.

À minha família, por ser meu porto seguro e por me apoiar sempre que preciso.

À minha companheira Bárbara Temponi Vilarino Godinho, aos meus primos Matheus Morais do Reis e Fabíola Andrade Palhares, à minha irmã Marília Gabriela Alencar de Morais e à minha tia Maria Aparecida Morais, por me ajudarem e me apoiarem na realização deste trabalho.

À Universidade Federal de Lavras (UFLA) e ao Departamento de Ciência da Computação (DCC) por proporcionar um ambiente adequado à minha formação.

Aos professores da UFLA, em especial os professores do Departamento de Ciência da Computação, pelos ensinamentos transmitidos.

Obrigado.

EPÍGRAFE

"O bom nome vale mais do que muitas riquezas."

Provérbios 22:1

"Não vos enganeis. As más companhias corrompem os bons costumes."

I Coríntios 15:33

SUMÁRIO

LISTA DE FIGURAS

RESUMO

A visualização da informação é uma ferramenta que permite ao usuário interagir com o conteúdo a ser visualizado e compreender o seu significado. Sua utilização implica na amplificação da cognição do usuário, o que facilita a realização de trabalhos em campo e/ou tomadas de decisão. Neste contexto, o presente trabalho apresenta a modelagem e o desenvolvimento de uma aplicação de visualização da informação que recupera os dados contidos no servidor e os exibe ao usuário, com base na sua geolocalização. A aplicação, cuja arquitetura é do tipo cliente-servidor, opera em tempo real e é composta por dois *softwares* implementados na linguagem Java e que se comunicam através de *socket* de rede, sendo que o software cliente é um aplicativo para a plataforma Android. A elaboração da aplicação foi baseada em conceitos de visualização importantes para o objetivo para o qual foi proposta, comparada com outros sistemas referentes à mesma temática para verificar sua atratividade e, por fim, gerada uma situação hipotética com o intuito de simular seu uso na prática. Concluiu-se que a ferramenta mostra-se viável para o auxílio de trabalhos em campo e/ou tomadas de decisão e que a mobilidade, o uso de características pré-atentivas e a interatividade são importantes quando se trabalha com visualização da informação. Ao comparar a aplicação com outras similares, verificou-se que a mesma apresenta vantagens adicionais para a finalidade proposta. A simulação, a partir de um exemplo prático hipotético, demonstrou a utilidade da aplicação e a viabilidade de seu desenvolvimento.

Palavras-chave: Visualização da Informação. Geolocalização. Tomada de decisão. Geovisualização. Aplicativo Android.

ABSTRACT

The information visualization is a tool that allows the user to interact with the content being visualized and understand its meaning. Besides that, it augments the user cognition and facilitates field work and decision making tasks. On this regard, this paper presents the modeling and development of an information visualization application which retrieves the data hosted in the server and shows it to the user, based on the user's geographic location. The application was entirely developed in Java, operates in real time and features a client-server architecture, where the client is an Android application and the communication is done through network sockets. The application's elaboration was based on information visualization concepts that are important for the goal it has been designed for and compared to other applications on the same subject to verify its attractiveness. Besides that, a hypothetical situation was created aiming to simutate the application's use in practice to verify its feasibility. It's concluded that the mobility, the use of preattentive characteristics and the interactivity are important when it comes to information visualization and that the developed application is reliable to aid field work and decision making tasks. It's also concluded that the application developed on this paper presents advantages regarding the main goal when compared to similar ones. Finally, the simulation, done from the hypothetical practical example, endorse the application's usability and its development reliability.

Keywords: Information Visualization. Geolocation. Decision making. Geovisualization. Android application.

1 INTRODUÇÃO

Desde os primórdios da humanidade, o homem busca entender o meio em que se encontra e como se localizar no mesmo. Inicialmente, os seres humanos se localizavam através das estrelas e utilizavam da sua memória para mapear o ambiente à sua volta (HUTOROWICZ; ADLER, 1911). Com o passar dos anos, novos recursos surgiram e um dos mais importantes foi a cartografia. Em International Cartographic Association (2011, p. 5, tradução nossa), define-se a cartografia como sendo "[...] a disciplina que lida com a concepção, produção, disseminação e estudo de mapas.".

Há séculos, a cartografia vem suprindo as necessidades de localização dos homens que, principalmente entre os séculos XV e XVIII, utilizavam mapas para navegação, comércio, campanhas militares, colonização e reivindicação de novas terras, atividades missionárias cristãs, etc (LEWIS, 2001).

Com o surgimento, a evolução e depois a popularização dos computadores e da internet, empresas e desenvolvedores de *software* dedicaram-se em recriar os mapas cartográficos para o ambiente informatizado, universalizando o acesso a esse tipo de informação.

Atualmente, o sistema mais popular, utilizado por milhões de pessoas ao redor do mundo, é o Google® Maps™, que consiste em um serviço web de cartografia, fornecido pelo Google, que sustenta vários outros serviços baseados em mapas, como: Google Ride Finder™, Google Transit™, sites de terceiros que utilizam a sua *API*[1] e o próprio site do Google Maps. O seu serviço oferece mapas de ruas, planejamento de rotas para deslocamentos a pé, carro, bicicleta ou transporte público e um localizador de empresas e atividades comerciais em centros urba-

[1] *Application Programming Interface* (Interface de Programação de Aplicativos).

nos (MASHABLE, INC., 2006). Desde o seu lançamento, o Google Maps possui a funcionalidade de se visualizar informações georreferenciadas em seus mapas. Tal recurso é conhecido como visualização da informação.

A visualização da informação é uma ferramenta de exploração visual que permite ao usuário interagir com o conteúdo a ser visualizado e compreender o seu significado. Além disso, tem como objetivo prover uma representação compacta do espaço de informação e amplificar a cognição do usuário.

Usuários que operam em campo, como geólogos, arqueólogos, turistas e socorristas, muitas vezes trazem consigo folhas de papel, e.g. mapas urbanos, formulários e planos técnicos, que contem informações necessárias às suas atividades. A condição móvel de tais usuários faz com que prefiram soluções mais portáveis e manuseáveis, como o papel, mesmo que as informações das quais necessitam estejam disponíveis em formato digital. Por exemplo, é mais fácil manusear e olhar para um mapa impresso ao invés de um digital em um *laptop* enquanto se está em movimento. Contudo, o aumento da disponibilidade de dispositivos móveis de tamanho reduzido e alto poder de processamento faz com que, em conjunto com a visualização da informação, os mesmos possam ser utilizados como auxílio na realização de diferentes tarefas em campo (BURIGAT; CHITTARO, 2007). Ainda, o fato da maioria dos dispositivos móveis produzidos atualmente possuir GPS[2], os torna candidatos mais apropriados para acompanhar os usuários nessas tarefas ou em atividades que exigem alta mobilidade.

[2]Segundo El-Rabbany (2006), o GPS (*Global Positioning System*) é um sistema de navegação por satélite que foi desenvolvido, inicialmente, pelo Departamento de Defesa dos Estados Unidos *(USDoD—United States Department of Defense)* no início da década de 70 como um sistema militar para satisfazer as necessidades militares dos EUA. Mais tarde, tornou-se disponível para civis e atualmente pode ser acessado tanto por civis quanto por militares. Conforme o mesmo autor, o GPS provê informações contínuas de posicionamento e de tempo, em qualquer lugar do mundo, sob quaisquer condições de tempo.

Dado o potencial do dispositivo móvel como uma ferramenta de obtenção de informações e auxílio à tomada de decisão em trabalhos de campo, além da sua potencial utilidade em áreas importantes para a sociedade, como a área de segurança pública, propõe-se o desenvolvimento do protótipo de uma aplicação baseada em mapas e com arquitetura cliente-servidor, composta por dois *softwares* a serem executados de modo concomitante em um dispositivo móvel (*software* cliente) e em um servidor (*software* servidor). Dessa forma, a aplicação tem como objetivo geral apresentar para o usuário, em tempo real, informações georreferenciadas recuperadas de uma base de dados em comum para auxiliá-lo em seu trabalho de campo e/ou tomada de decisão. Tais informações são exibidas acerca da posição geográfica coletada através do GPS do dispositivo de modo que o usuário acesse essas informações durante a execução de atividades que exijam alta mobilidade, o que torna a execução dessas atividades mais fácil e eficiente.

Nesse contexto, os objetivos específicos consistem em: definir as tecnologias adequadas para a elaboração da aplicação; projetar a arquitetura da mesma; definir o protocolo de comunicação para que o protótipo seja funcional; empregar conceitos de visualização que sejam úteis para o objetivo para o qual a aplicação foi proposta; comparar a aplicação desenvolvida com outras similares a fim de verificar a sua real atratividade; e empregá-la em uma situação hipotética com o intuito de constatar a viabilidade de seu desenvolvimento.

2 REFERENCIAL TEÓRICO

Visto que o objetivo geral do presente estudo consiste no desenvolvimento do protótipo de uma aplicação para a visualização de informações em tempo real sobre um mapa, para auxiliar o usuário em trabalhos de campo e/ou tomadas de decisão, é necessário que se tenha um breve entendimento sobre cartografia, visualização da informação, percepção humana, atividade cognitiva, tomada de decisão, interatividade, *design* de interfaces visuais e dispositivos móveis.

2.1 Cartografia e Visualização da Informação

Segundo Woodward e Harley (1987, p. 1, tradução nossa), os mapas são uma das formas mais antigas de comunicação entre seres humanos. Além disso, conforme o mesmo autor, mapear—no sentido de representar, como desenhar—"[...] precede tanto a linguagem escrita quanto os sistemas numéricos e, embora mapas não fossem objetos de uso contínuo em muitos lugares do mundo até o Renascimento Europeu, existiram poucas civilizações que não os utilizaram no mundo como um todo.". Desse modo, segundo o autor, o mapa é tanto antigo, quanto difundido.

Sendo assim, de acordo com Friendly (2009), os primeiros sinais de manifestação da área da visualização surgiram antes do século XVII a partir da elaboração de mapas, para auxiliar na navegação e exploração, diagramas e tabelas. No entanto, no século XVIII, o mesmo autor afirma que os criadores de mapas começaram a tentar mostrar mais do que uma posição geográfica em um mapa, o que resultou na invenção de novas formas gráficas, como contornos e curvas de nível, e nas primeiras tentativas de mapeamento de dados geológicos, econômicos e médicos. Ainda, o autor atesta que, na primeira metade do século XIX, devido a inovações nas áreas de *design* e tecnologia, todas as formas modernas de exi-

bição de dados já haviam sido inventadas, como os gráficos de barra e pizza, e, ao final do mesmo século, escritórios de estatística já haviam se estabelecido em toda a Europa confirmando a importância da informação para a industrialização, o comércio e o transporte. Friendly (2009) também afirma que, na primeira metade do século XX, métodos gráficos foram utilizados pela primeira vez para a obtenção de novos conhecimentos em várias ciências e, a partir de meados do mesmo século, a evolução na área de ciência da computação fez com que novos métodos e técnicas de visualização fossem criados e provocou o desenvolvimento de sistemas computacionais interativos tal como o desenvolvimento de novos paradigmas de manipulação direta para a análise visual de dados.

Dada a importância e o impacto da visualização da informação conforme Friendly (2009), Chen (2004 apud BAI; WHITE; SUNDARAM, 2009, p. 94, tradução nossa) define a mesma como "[...] uma ferramenta de exploração visual que permite ao usuário interagir com o conteúdo a ser visualizado e compreender o seu significado.". Desse modo, para que essa importância seja entendida, é relevante se ter uma noção sobre a percepção humana.

2.2 Percepção humana

Segundo Healey e Enns (2012), há vários anos pesquisadores tem investigado como o sistema visual humano analisa imagens. A partir dessas pesquisas, descobriu-se que a visão humana reconhece detalhes, como forma e cor, em apenas uma pequena área do campo visual. Desse modo, os autores afirmam que, a fim de enxergar detalhes de mais de uma região, nossos olhos realizam os movimentos de *fixation-saccade*, ou seja, se movem rapidamente alternando entre curtos períodos estacionários, quando detalhes são obtidos—*fixation*, e breves períodos de cegueira, quando os olhos se movem para outro local—*saccade*. Sendo as-

sim, conforme Healey e Enns (2012, p. 1170, grifo do autor, tradução nossa), o termo *"atenção visual"* surgiu para denotar os mecanismos visuais que determinam quais regiões de uma imagem são escolhidas para se obter mais detalhes e, a partir disso, estudos sobre tal atenção em seres humanos levaram à descoberta de um conjunto de características que são detectadas rapidamente pelos processos visuais. De acordo com os mesmos autores, essas características são denominadas pré-atentivas e a sua detecção ocorre normalmente em menos de 250 milissegundos, antes da atenção focada, ou seja, tão logo ocorra o primeiro movimento de *fixation-saccade*. Nesse contexto, os autores ainda afirmam que a visualização pode tirar proveito de tais características a fim de chamar a atenção para áreas de potencial interesse em uma imagem ou em uma tela. Na Figura 1, tem-se o exemplo de uma imagem com uma característica pré-atentiva.

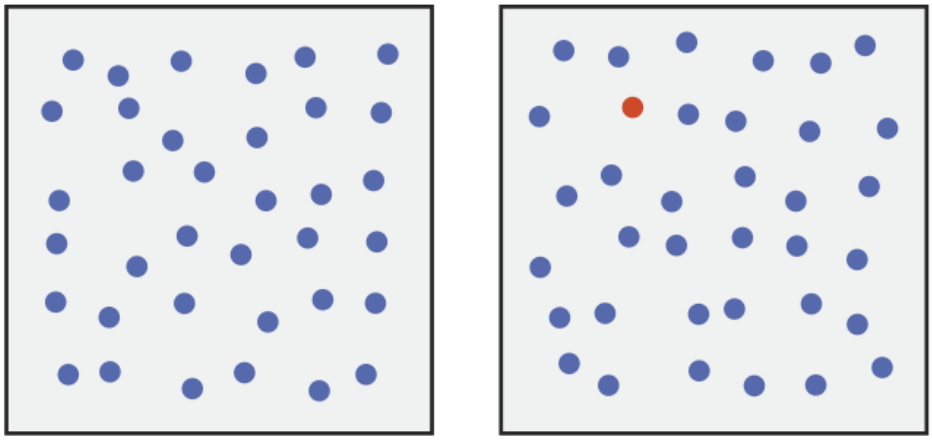


Figura 1 Detecção de alvo através do matiz
Fonte: Adaptado de Healey e Enns (2012)

Nessa figura é possível determinar, com rapidez e através do matiz[3], quando o alvo (em vermelho) está presente ou não. Healey e Enns (2012) afirmam que

[3] *Hue* em inglês.

um alvo, identificado por uma propriedade única, se permite sobressair (círculo vermelho—Figura 1); o que implica na sua detecção rápida e sem esforço independente do número de distrações (círculos azuis). Outras características também identificadas como pré-atentivas podem ser visualizadas na Figura 2.

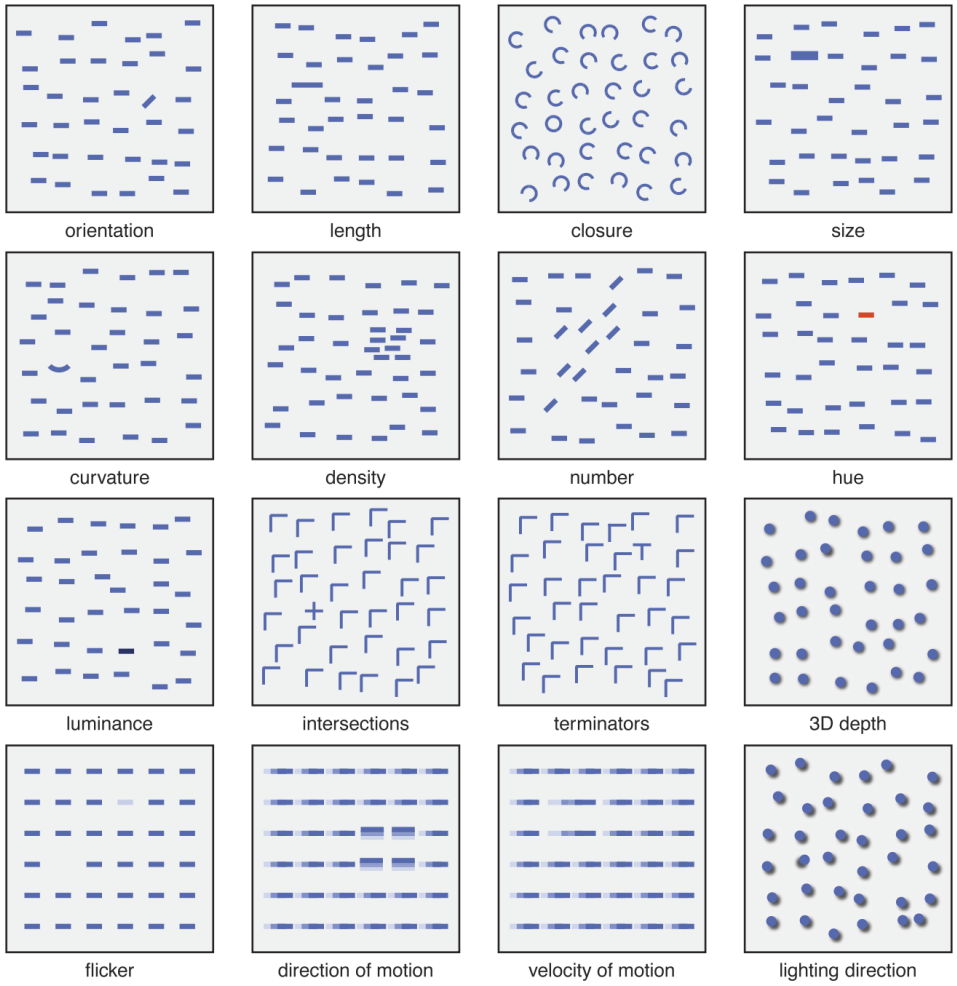


Figura 2 Características visuais pré-atentivas
Fonte: Adaptado de Healey e Enns (2012)

De forma mais resumida, Jun, Landry e Salvendy (2011 apud GOUVEIA, 2013) subdividem a percepção visual em dois componentes: extração de caracte-

rísticas e percepção de padrões. O primeiro funciona através de informação sensorial e permite a extração de diversas características em paralelo que chamem a atenção, como cor, bordas, forma, textura e movimento. O segundo componente determina a forma como essas características se associam e a quais objetos pertencem, criando uma identidade para os mesmos. Assim, Gouveia (2013) afirma que o valor agregado dessa identidade é tal que, mesmo quando alguma característica está obstruída ou ausente, a identificação do objeto é possível. Por fim, Healey e Enns (2012) atestam que um entendimento sobre percepção permite que se melhore a visualização de informações tanto em qualidade quanto em quantidade.

2.3 Atividade cognitiva

Conforme Card, Mackinlay e Shneiderman (1999 apud DARADKEH, 2012, p. 34, tradução nossa), a visualização da informação é definida como "[...] o uso do computador para dar suporte à representação visual e interativa de dados abstratos para se amplificar a cognição.". De acordo com Hutchins (1995 apud WARE, 2004), a maior parte da cognição é feita através da interação com ferramentas cognitivas, como lápis, papel, calculadora e sistemas de informação baseados em computador. Essa afirmação é reforçada por Ware (2004), que afirma que a cognição ocorre como um processo em sistemas que contem várias pessoas e vários instrumentos cognitivos. Sendo assim, o mesmo autor declara que o ato de pensar ocorre através da interação entre indivíduos que utilizam ferramentas cognitivas e operam em redes sociais. Além disso, Ware (2004) aponta algumas vantagens da visualização na qualidade de instrumento cognitivo, como permitir a compreensão de grandes quantidades de dados, propiciar que problemas com os dados se tornem aparentes, favorecer o entendimento das características dos dados, tanto em larga quanto em baixa escala, e facilitar a formação de hipóteses.

2.4 Tomada de decisão

No contexto de tomada de decisão, Harris (1998, tradução nossa) afirma que a mesma é definida como "[...] o processo de reduzir suficientemente a incerteza e a dúvida sobre alternativas, de modo a permitir que uma escolha sensata entre as mesmas seja feita.". Ainda, Bettman, Luce e Payne (1998) sugerem alguns objetivos secundários que são desejáveis ao se tomar uma decisão, como minimizar o esforço cognitivo necessário para a tomada da mesma, minimizar as experiências ruins ao tomá-la e ter um embasamento melhor para se justificar a decisão. Nesse contexto, Dull e Tegarden (1999) afirmam que é a habilidade da visualização, de amplificar a cognição humana, que permite maior rigor e exatidão na tomada de decisões. Dessa forma, Gröller (2002 apud DARADKEH, 2012) atesta que a visualização reduz a carga cognitiva necessária às tarefas de tomada de decisão, deslocando tal carga para o sistema de percepção humano. Assim, Ware (2004) corrobora tais colocações ao afirmar que a visualização tem um papel importante sobre os sistemas cognitivos, pelo fato do ser humano adquirir mais informações através da visão do que através de todos os outros sentidos combinados, dado que os cerca de 20 bilhões de neurônios do nosso cérebro, dedicados a analisar informações visuais, nos fornecem um mecanismo para o reconhecimento de padrões que é fundamental para a nossa atividade cognitiva. Desse modo, Tarantino (2000) afirma que a visualização da informação tem sido utilizada como suporte principal nas tarefas de tomada de decisão.

2.5 Interatividade e *Design* de interfaces visuais interativas

Ware (2004) também afirma que a interatividade é tão importante para a cognição quanto a própria visualização em si. A fim de ilustrar, ele alega que da mesma forma que as pessoas, ao explorar uma cidade, constroem um modelo

espacial cognitivo através de pontos de referência, algo similar ocorre quando exploram espaços de informação. Dessa forma, o mesmo autor destaca alguns pontos essenciais ao *design* de visualizações interativas, que são: (1) os dados devem ser apresentados de forma que padrões de informação sejam fáceis de perceber, o que permite aproveitar a habilidade do nosso processo visual de reconhecê-los; (2) o impacto cognitivo da interface deve ser minimizado, de modo que o raciocínio do usuário seja aplicado sobre o problema e não sobre a interface; e (3) a interface deve ser otimizada para a busca rápida de informações. Além disso, o mesmo autor enfatiza que as melhores visualizações não são imagens estáticas para serem utilizadas, por exemplo, em livros; mas sim, componentes visuais dinâmicos que respondem às necessidades de uma perspectiva diferente ou de informações mais detalhadas.

Shneiderman (1996, p. 337) resume algumas diretrizes básicas para o *design* de interfaces visuais interativas em uma filosofia: *"Overview first, zoom and filter, then details-on-demand"*. O mesmo autor explica as partes da mesma como se segue:

- *Overview:* Ter-se uma visão global das informações.
- *Zoom:* Aproximar o que se vê, sobre os itens de interesse.
- *Filter:* Ocultar os itens que não se tem interesse.
- *Details-on-demand:* Selecionar um item, ou grupo de itens, para se obter detalhes quando necessário.

Embora Shneiderman (1996) tenha estabelecido nessa filosofia uma ordem para as ações (primeiro obter uma visão global, depois aproximar, filtrar e então obter detalhes), Ware (2004) afirma que as pessoas são igualmente propensas a verem um detalhe primeiro, para depois obter uma visão global, encontrar alguma outra

informação e então obter detalhes dessa outra informação. Entretanto, independente dessa colocação, o autor destaca que o importante é que uma interface de visualização suporte todas essas atividades.

2.6 Dispositivos móveis

Dados os benefícios da visualização da informação, Chittaro (2006) atesta que portar as técnicas de visualização para os dispositivos móveis é uma progressão natural, a fim de aproveitar tais benefícios em qualquer lugar e a qualquer momento. Ainda, Chittaro (2006, p. 3, tradução nossa) enfatiza que a "sensibilidade ao contexto"[4] acrescenta valor à visualização em dispositivos móveis, dos quais os aplicativos podem ser sensíveis, por exemplo, às condições de iluminação, o que permite adaptar o brilho e o contraste da visualização, e à localização geográfica, o que admite exibir para o usuário o mapa ou diagrama apropriado.

Além disso, Burigat e Chittaro (2008) atestam que os avanços nas tecnologias de comunicação móvel afetam significativamente a natureza e a disponibilidade de dados georreferenciados, uma vez que os dispositivos móveis são cada vez mais importantes tanto como fontes de informações georreferenciadas quanto como meios de acesso às mesmas. Desse modo, além de ser uma progressão natural, os autores afirmam que estender as técnicas de exploração e análise visual de dados ao cenário móvel permitiria auxiliar nas decisões do usuário exatamente onde e quando estas últimas fossem necessárias. Ainda, os mesmos autores atestam que os dispositivos móveis já tem tornado possível o desenvolvimento de aplicações que empregam tais técnicas para o uso de dados georreferenciados em trabalhos de campo. Com base nessa última colocação, Burigat e Chittaro (2008) evidenciam a utilidade dos dispositivos e das aplicações móveis, dos dados geor-

[4] *"context-awareness"*.

referenciados e da visualização da informação no contexto de tomada de decisão e também citam algumas áreas que podem se beneficiar disso, como turismo, negócios, gerenciamento de recursos naturais, etc.

Sendo assim, Payne et al. (2009) reforçam o valor dos dispositivos móveis como ferramentas de tomada de decisão em campo, ao atestarem que os mesmos estão cada vez mais aptos a realizar localmente tarefas que antes só eram possíveis em computadores de mesa devido ao processamento exigido. Ainda, os mesmos autores afirmam que, conforme o poder de processamento dos dispositivos móveis aumenta, um número maior de indústrias que utilizam força de trabalho móvel se beneficiará da coleta, análise e apresentação de informações *in loco*[5], à medida que recorrerem a aplicações móveis para suprirem suas necessidades informacionais. Além disso, Payne et al. (2009) atestam que, pelo fato de 85% de toda informação gerada mundialmente possuir um componente espacial, como por exemplo endereço, código postal ou coordenada geográfica, é oportuno que dispositivos móveis que possuem algum mecanismo de localização forneçam informações úteis em campo.

Além do mais, Burigat e Chittaro (2013) alegam que os dispositivos móveis atuais tem poder suficiente para exibir mapas, imagens e outros espaços de informação complexos. Os mesmos autores também afirmam que sistemas baseados em mapas, websites interativos, *softwares* para lidar com imagens e outras aplicações e serviços não são mais exclusividade dos computadores de mesa. Entretanto, Burigat e Chittaro (2013) atestam que, infelizmente, visualizar informações de forma efetiva em dispositivos móveis não é trivial e, portanto, não há garantias que boas soluções de visualização para computadores de mesa possam ser aplicadas com sucesso em dispositivos móveis, pelo fato destes últimos pos-

[5] "No próprio local; *in situ*." (PRIBERAM INFORMÁTICA, S.A., 2010b).

suírem telas menores, menor poder de processamento e diferentes mecanismos de entrada. Desse modo, os autores salientam que a maioria dessas limitações não desaparecerá em um futuro próximo sem que a portabilidade seja sacrificada. No entanto, apesar desses empecilhos, aplicações móveis tem sido usadas com sucesso nas áreas da saúde (DALA-ALI; LLOYD; AL-ABED, 2011), monitoramento de tráfego (CHEN; CHEN; CHANG, 2011), segurança pública (MONARES et al., 2011), turismo (AHAS et al., 2008), educação (CHANG; CHEN; HSU, 2011; RUCHTER; KLAR; GEIGER, 2010) e agricultura de precisão (ZHENG et al., 2011).

2.7 Programas existentes

Considerando-se os conceitos acima apresentados, programas que se beneficiam dos mesmos tem sido desenvolvidos a fim de auxiliar o usuário na realização/resolução de tarefas/problemas específicos.

2.7.1 CIVIL

Nesse contexto, Wu et al. (2009) apresentam em seu trabalho o CIVIL[6], um protótipo de um sistema web colaborativo, baseado em mapas, para dar suporte à tomada de decisão através da visualização da informação. Segundo os autores, o sistema aumenta a interface do usuário com ferramentas de visualização que permitem o compartilhamento de informações entre as pessoas de uma equipe. A ideia principal desse sistema consiste em possibilitar o compartilhamento de informações entre os membros de pequenos grupos, cujos participantes são pessoas especializadas em domínios específicos, para auxiliá-los na tomada de decisão, por

[6] *geo-Collaboration Information VIsuaLization.*

exemplo, no planejamento de soluções para situações de emergência. Na Figura 3 pode-se visualizar a interface gráfica do usuário do sistema.

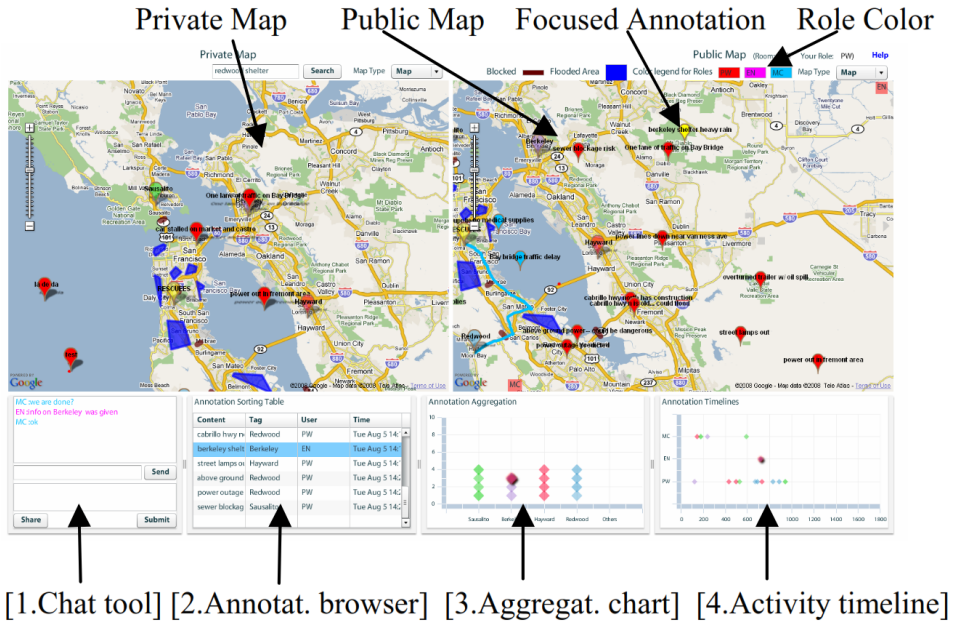


Figura 3 Interface gráfica do CIVIL
Fonte: Wu et al. (2009)

Como se vê na figura, o sistema possui dois mapas, um privado e outro público, uma área de bate-papo, um navegador de anotações, uma tabela de agregação e um cronograma de atividades. Segundo Wu et al. (2009), o suporte a múltiplos mapas para atividades públicas e privadas é essencial quando os integrantes da equipe estão distribuídos em diversos locais, de modo que é possível que cada membro faça análises específicas do seu domínio de conhecimento no mapa privado antes de compartilhar qualquer informação com os outros usuários no mapa público. Ainda conforme os mesmos autores, a tabela de agregação permite aos integrantes da equipe classificar as informações como melhor lhe convêm, visto que o trabalho colaborativo frequentemente envolve a integração de informa-

ções de diferentes fontes. Além disso, o cronograma de atividades possibilita que a sequência das atividades realizadas pelos membros do grupo seja armazenada e acessada a qualquer momento, uma vez que, segundo Wu et al. (2009), os dados do processo adotado pela equipe são tão importantes quanto as informações compartilhadas.

2.7.2 MoViSys

Outro exemplo de aplicativo é o MoViSys[7], apresentado por Vaz et al. (2008), que é o protótipo de um sistema de visualização de informações georreferenciadas em dispositivos móveis. De acordo com os autores, o sistema permite a visualização de informações que possuem mais de um atributo e são classificadas em categorias, além de ter sido otimizado para a filtragem dessas informações de modo a reduzir a densidade de ícones exibidos na tela. Segundo Vaz et al. (2008), a ideia principal consiste na exibição de pontos de interesse georreferenciados relevantes de acordo com os critérios de busca estabelecidos pelo usuário. Na Figura 4 tem-se a interface gráfica do usuário do MoViSys.

[7] *Mobile Visualization System*.

Figura 4 Interface gráfica do MoViSys
Fonte: Vaz et al. (2008)

Como pode ser visualizado na figura, o sistema é composto pelo mapa de navegação e pela interface na qual os critérios de filtragem são definidos. Conforme os autores, os pontos de interesse são armazenados em uma base de dados SQL Server® e recuperados diretamente do dispositivo móvel, enquanto que os mapas de navegação são obtidos através do Google Maps. Ainda, os autores afirmam que o sistema atualiza a área do mapa a ser visualizada de acordo com o posicionamento geográfico do dispositivo móvel, coletado através do GPS, e, à medida que tal área é atualizada, novas requisições são feitas ao SQL Server, que por sua vez retorna a lista de pontos de interesse para o usuário. Esse funcionamento pode ser verificado na (Figura 5), que ilustra a arquitetura do sistema.

SQL Server é uma marca registrada da Microsoft Corporation.

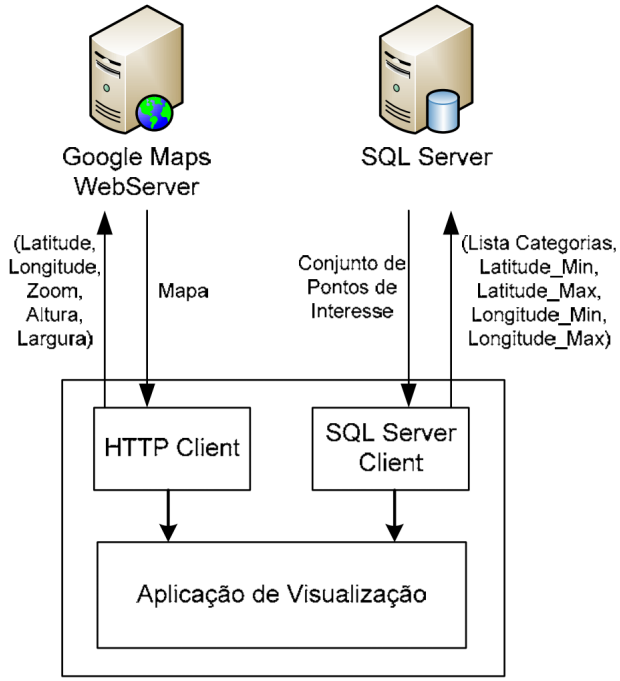


Figura 5 Arquitetura do MoViSys
Fonte: Adaptado de Vaz et al. (2008)

3 METODOLOGIA

Segundo Jung (2004), Silva e Menezes (2001) e Moresi (2003) as pesquisas podem ser classificadas quanto à sua natureza, quanto à forma de abordar o problema, quanto aos objetivos e quanto aos meios de investigação. Sendo assim, segundo as definições das classificações dadas pelos mesmos autores, esta pesquisa classifica-se quanto a sua natureza como aplicada ou tecnológica, dado que conhecimentos básicos e tecnologias existentes são empregados para a solução do problema, e classifica-se, quanto a forma de abordar o problema, como pesquisa qualitativa, uma vez que este estudo não envolve e não é quantificado em números. Ainda, o presente estudo classifica-se quanto aos objetivos como pesquisa exploratória, visto que procurou-se aplicar ao problema teorias e conceitos existentes além da realização do estudo de um caso hipotético e, por fim, este trabalho é classificado, quanto ao meios de investigação, como estudo de caso.

O presente trabalho foi realizado nos laboratórios de graduação do Departamento de Ciência da Computação (DCC) da Universidade Federal de Lavras (UFLA), ao longo do primeiro semestre do ano de 2013. Ainda, a obtenção das referências bibliográficas se deu, principalmente, no portal de periódicos da Capes[8], nos portais da *ScienceDirect*[9], *Scopus*[10], *ACM*[11], *IEEE Xplore Digital Library*[12], Microsoft® Academic Search[13], Google Scholar™[14] e Google Books™[15], além do acervo impresso da referida universidade. Desse modo, a realização do traba-

[8] *http://www.periodicos.capes.gov.br*
[9] *http://www.sciencedirect.com*
[10] *http://www.scopus.com*
[11] *http://dl.acm.org*
[12] *http://ieeexplore.ieee.org*
Microsoft é uma marca registrada da Microsoft Corporation.
[13] *http://academic.research.microsoft.com*
[14] *http://scholar.google.com*
Google Scholar e Google Books são marcas comerciais do Google Inc.
[15] *http://books.google.com*

lho se deu através do desenvolvimento de um protótipo de uma aplicação para a visualização de informações em mapas, como um projeto auxiliar aos projetos dos grupos de pesquisa GRA[16] e GTIP[17] do DCC/UFLA. Além disso, o protótipo da aplicação aqui proposta foi desenvolvido de modo modular, o que permite a reutilização de código, e com suporte aos idiomas inglês e português, com possibilidade de expansão para outras línguas.

3.1 Ferramentas

A aplicação foi inteiramente desenvolvida na *IDE*[18] Eclipse[19] versão 4.2.1, conhecida como Eclipse Juno. No caso do *software* cliente, o Kit de Desenvolvimento de Sistema do Android™[20,21] e as Ferramentas de Desenvolvimento do Android[21,22] foram utilizados por se integrarem ao Eclipse e facilitarem o processo de desenvolvimento e depuração.

3.2 Dependências

O *software* cliente da aplicação é um aplicativo para a plataforma Android na sua versão 2.3.3 ou superior. A execução adequada desse aplicativo depende dos aplicativos Google Maps e Google Play Services. Portanto, é necessário que ambos estejam instalados no dispositivo móvel do usuário. O funcionamento do *software* servidor depende da Máquina Virtual Java™[23] (versão 1.7 ou superior),

[16] Grupo de Realidade Virtual e Aumentada.
[17] Grupo de pesquisa e desenvolvimento de Tecnologias para Inovações Pedagógicas.
[18] *Integrated Development Environment.*
[19] Encontrado em *http://www.eclipse.org/downloads/*.
[20] *Android System Development Kit*—Android *SDK*.
[21] Ambos podem ser encontrados em *http://developer.android.com/sdk/index.html*.
Android é uma marca comercial do Google Inc.
[22] *Android Development Tools*—Android *ADT*.
Java é uma marca comercial da Oracle Corporation.
[23] *Java Virtual Machine (JVM)*—pode ser instalada através do *Java Development Kit (JDK)* ou do *Java Runtime Environment (JRE)*, encontrados em *http://www.java.com/pt_BR/download/*.

instalada no computador que atuará como servidor, e do Sistema Gerenciador de Banco de Dados MySQL™ (versão 5.0.10 ou superior) instalado no computador que atuará como o servidor de banco de dados.

3.3 Plataforma móvel

Os usuários da aplicação proposta nesta pesquisa interagem com um dispositivo móvel, a fim de obterem informações de pontos geográficos durante atividades que exigem alta mobilidade. Sendo assim, houve a necessidade de se escolher uma plataforma móvel alvo para o desenvolvimento do *software* cliente. Com o intuito de tornar o projeto sem custo, a única plataforma para dispositivos móveis, gratuita e com o código aberto[24], disponível até a data da execução deste estudo, é o Android. As outras opções, que seriam o Ubuntu OS e o Firefox OS, ainda estão em desenvolvimento (BROWN, 2013) ou foram lançadas em julho de 2013 (NYMAN, 2013), quando este projeto já estava em andamento. Além disso, segundo a IDC Corporate USA (2013), ao final do ano de 2012, 70,1% dos dispositivos móveis do planeta rodavam Android, o que torna os aplicativos desenvolvidos para essa plataforma mais acessíveis.

O Android é uma plataforma aberta projetada para dispositivos móveis, mantida pela *Open Headset Alliance* (um grupo sem fins lucrativos formado por operadoras de serviços móveis, fabricantes e outros). Essa plataforma foi criada com o objetivo de acelerar a inovação na área de dispositivos móveis e oferecer uma experiência melhor, mais rica e menos onerosa aos consumidores. (GARGENTA, 2011)

MySQL é uma marca comercial da Oracle Corporation.

[24] "código aberto": código fonte do programa disponível, para que outros desenvolvedores de *software* possam customizá-lo, alterá-lo, corrigir erros etc.

3.4 Linguagem de programação

A linguagem de programação Java, que é padrão no desenvolvimento de aplicativos para a plataforma Android, foi escolhida para a implementação da aplicação proposta no presente trabalho, por ser uma linguagem moderna que enfatiza a portabilidade e a segurança. Além disso, conforme Anguish, Buck e Yacktman (2002), tal linguagem é ideal para o desenvolvimento de *softwares* que rodarão processos de longa duração em um servidor e que farão alto uso de recursos de rede (para comunicação) e de banco de dados (para obtenção de informação). Ainda, segundo o mesmo autor, a coleta de lixo do Java simplifica o desenvolvimento e evita potenciais erros de gerenciamento de memória dinâmica. Por fim, o Java é gratuito e possui o seu código aberto (o que mantém este projeto sem custo), e é amplamente utilizado (9 milhões de desenvolvedores) em todo o mundo (ORACLE CORPORATION, 2012).

4 RESULTADOS E DISCUSSÃO

A fim de atingir os objetivos do presente estudo, foi desenvolvida uma aplicação para a visualização de informações georreferenciadas na qual, como pode ser visto na Figura 6, o usuário realiza o *login* (1) e tem acesso a um mapa de navegação (2). Desse modo, ao navegar pelo mapa, ícones aparecem em tempo real na tela do dispositivo indicando para o usuário sobre quais locais há informações que podem ser visualizadas (3) e, ao interagir com esses ícones, o usuário tem acesso às informações (4).

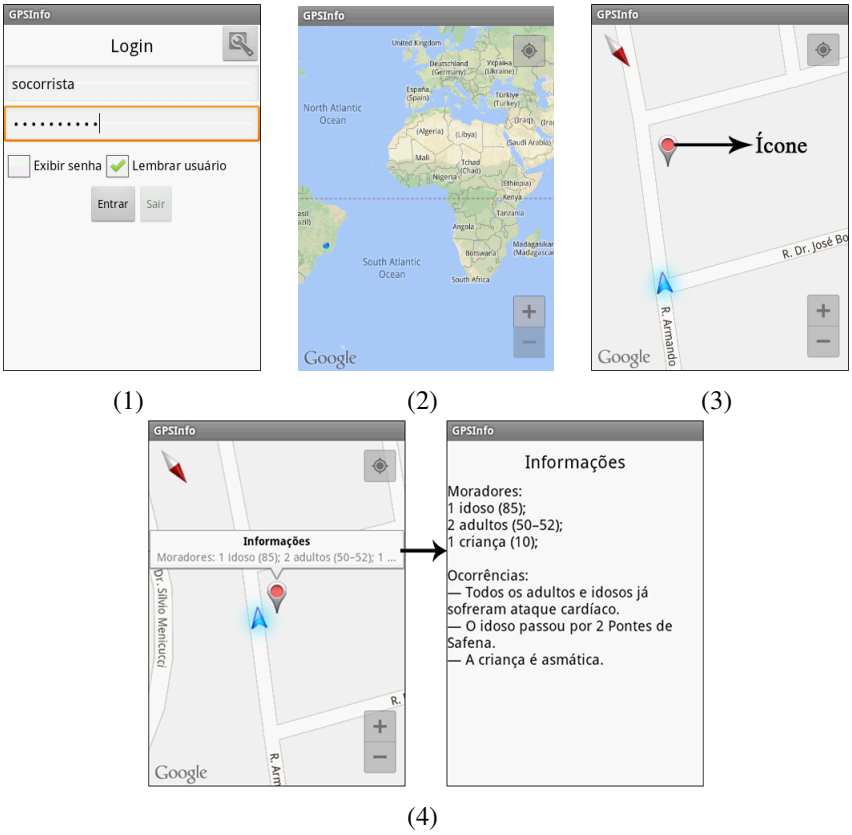


Figura 6 Telas da aplicação

4.1 Arquitetura geral

Conforme Donahoo e Calvert (2009), cada comunicação de um sistema do tipo cliente-servidor é análoga a uma ligação telefônica, que é iniciada por uma parte (a pessoa que faz a ligação telefônica), enquanto a outra parte responde ao contato (a pessoa que atende a ligação e começa a falar). Ainda, como pode ser visto na Figura 7, de acordo com os mesmos autores os termos cliente e servidor desempenham respectivamente os papéis: iniciar uma comunicação; e aguardar passivamente e responder aos clientes que entrarem em contato. Por fim, Donahoo e Calvert (2009) afirmam que os *softwares* cliente e servidor, juntos, compõem uma aplicação.

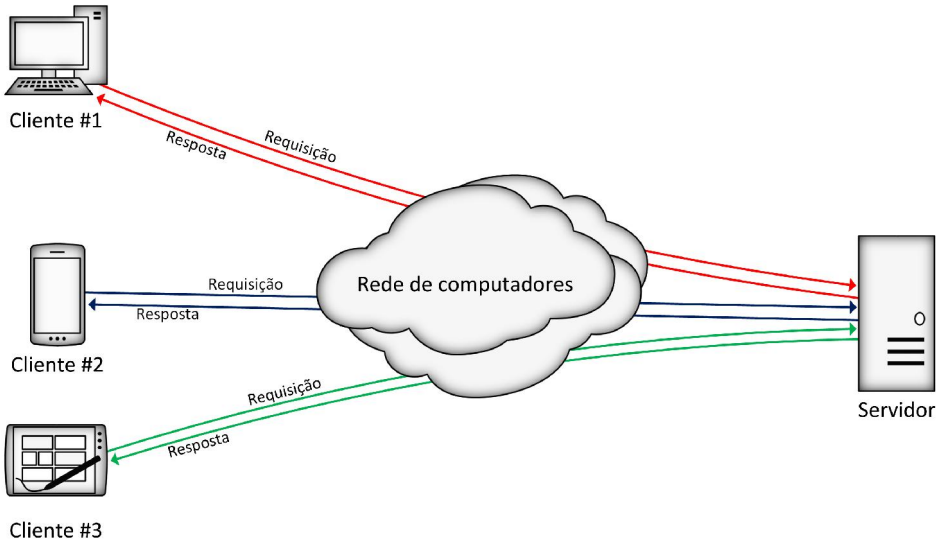


Figura 7 Arquitetura cliente-servidor

Considerando-se a natureza do problema de fornecer informações georreferenciadas para vários usuários, concluiu-se que a arquitetura cliente-servidor era mais apropriada para o desenvolvimento da aplicação proposta pelo fato de, segundo Farnaghi, Mansourian e Toomanian (2009), ser uma estrutura distribuída

que divide tarefas entre os prestadores de serviços (servidores) e quem os solicita (clientes). Segundo o mesmo autor, tal arquitetura é a mais utilizada nas aplicações em que vários usuários acessam informações provenientes de uma base de dados em comum. O autor ainda afirma que, geralmente, o servidor é um computador de alto desempenho que compartilha seus recursos com os clientes e, em contra partida, o cliente não compartilha nenhum de seus recursos, mas solicita ao servidor algum conteúdo ou serviço.

Na Figura 8 expõe-se a arquitetura da aplicação proposta, na qual a direção das setas indica a direção do fluxo de dados.

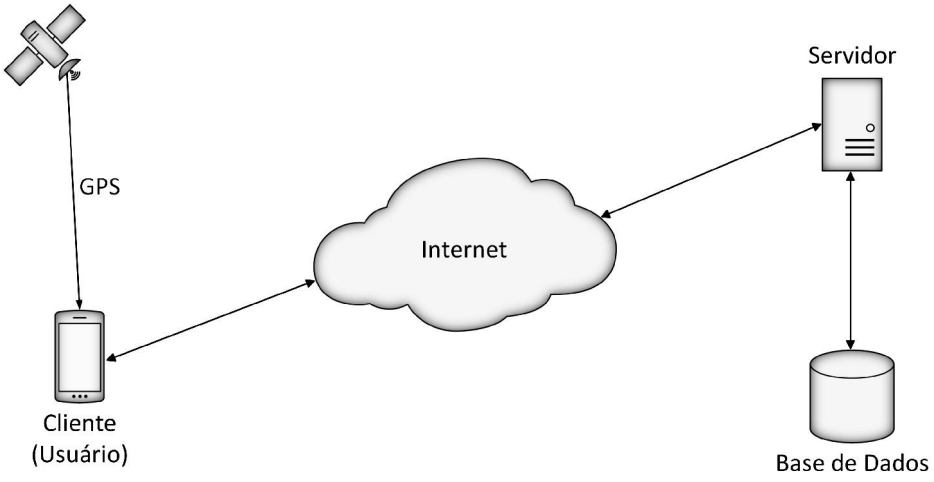


Figura 8 Arquitetura geral da aplicação

4.2 Conexão

Devido à natureza da aplicação aqui proposta, de fornecer para o usuário informações em tempo real, a comunicação entre os *softwares* cliente e servidor é intensa. Sendo assim, optou-se pela conexão *socket* para se estabelecer e gerenciar a comunicação entre o cliente e o servidor por ser uma conexão persistente—

ou seja, uma única conexão é estabelecida para o envio/recebimento de todas as requisições/informações—e por ser, portanto, adequada para o transporte intenso de dados. Além disso, a conexão *socket* permite que o servidor se comunique com o cliente sem que este último tenha feito solicitação alguma, como por exemplo, para notificar os usuários quando alguma informação tiver sido atualizada no banco de dados.

A comunicação ponto-a-ponto em uma rede de computadores é definida pelo modelo OSI[25]. Esse modelo é subdividido em sete camadas (Figura 9), que tem responsabilidades específicas para assegurar o transporte de dados através da rede e tornar tal transporte transparente para o usuário. O *socket* se localiza na 5ª camada (*Session*[26]), responsável por estabelecer, gerenciar e finalizar a comunicação entre dois programas (cliente e servidor).

[25] *Open Systems Interconnection.*
[26] Sessão.

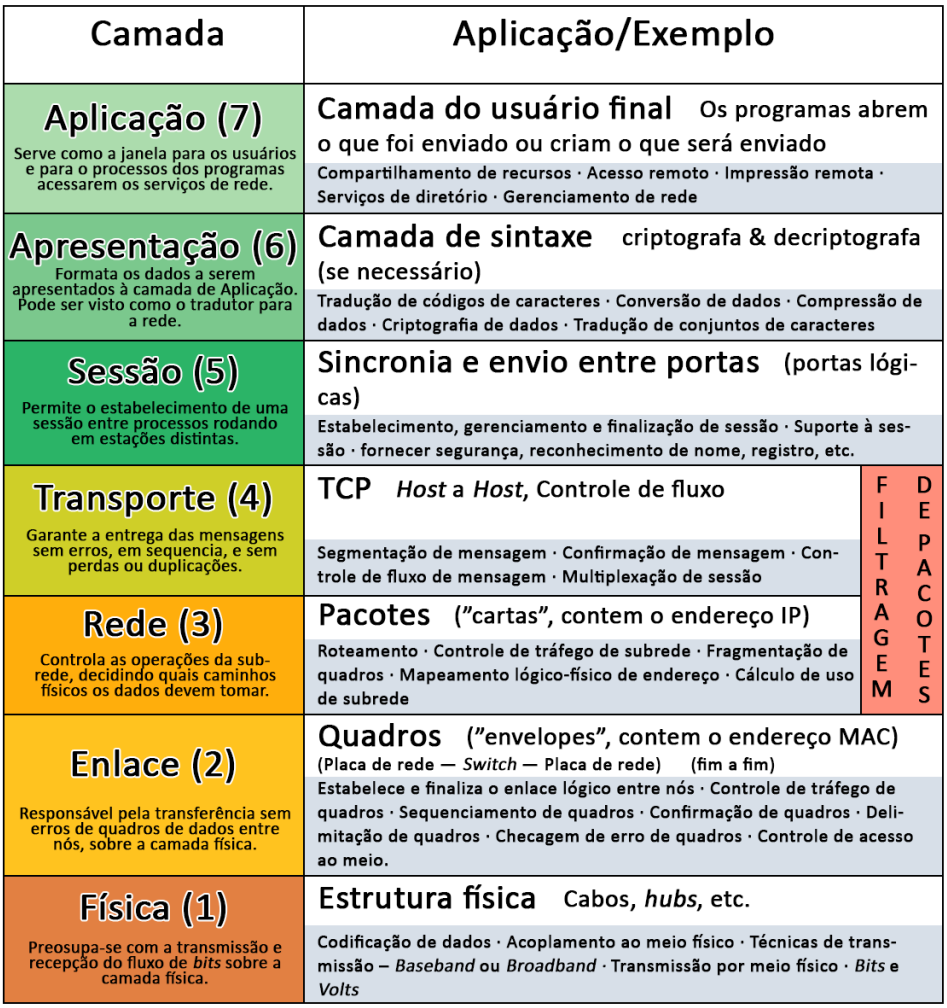


Figura 9 Modelo OSI (*Open Systems Interconnection*)
Fonte: Adaptado e traduzido de Cisco Systems Inc (2012)

4.3 Diagrama de caso de uso

O diagrama de caso de uso é um mapeamento superficial da aplicação que mostra, de modo geral, como o usuário poderá interagir com a mesma. Aqui, tal interação se dá a partir do momento em que o usuário configura o aplicativo no

dispositivo móvel para se conectar ao servidor e realiza o *login*, que é necessário para se ter acesso ao mapa de navegação. A partir daí, o usuário tem acesso às informações georreferenciadas em tempo real. Isto é, só depois de ingressar no mapa de navegação é possível visualizar tais informações, como pode-se observar na Figura 10. Por fim, após o *login* bem sucedido, é possível que o usuário realize o *logout* e saia da aplicação a qualquer instante durante o seu uso.

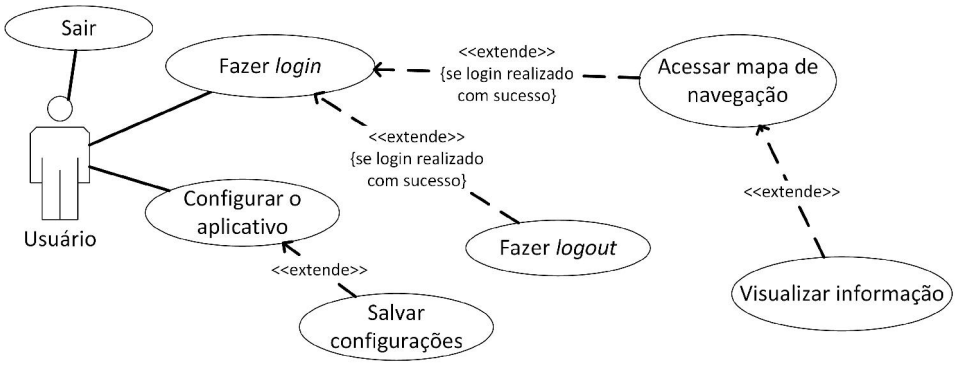


Figura 10 Diagrama de caso de uso

4.4 Diagrama de atividade

O diagrama de atividade é um complemento ao diagrama de caso de uso e fornece quais ações o usuário pode tomar para interagir com a aplicação, o que melhora o entendimento sobre o uso da mesma. Na Figura 11 é possível ver, com mais detalhes, como o usuário interage com a aplicação e qual a consequência de cada interação.

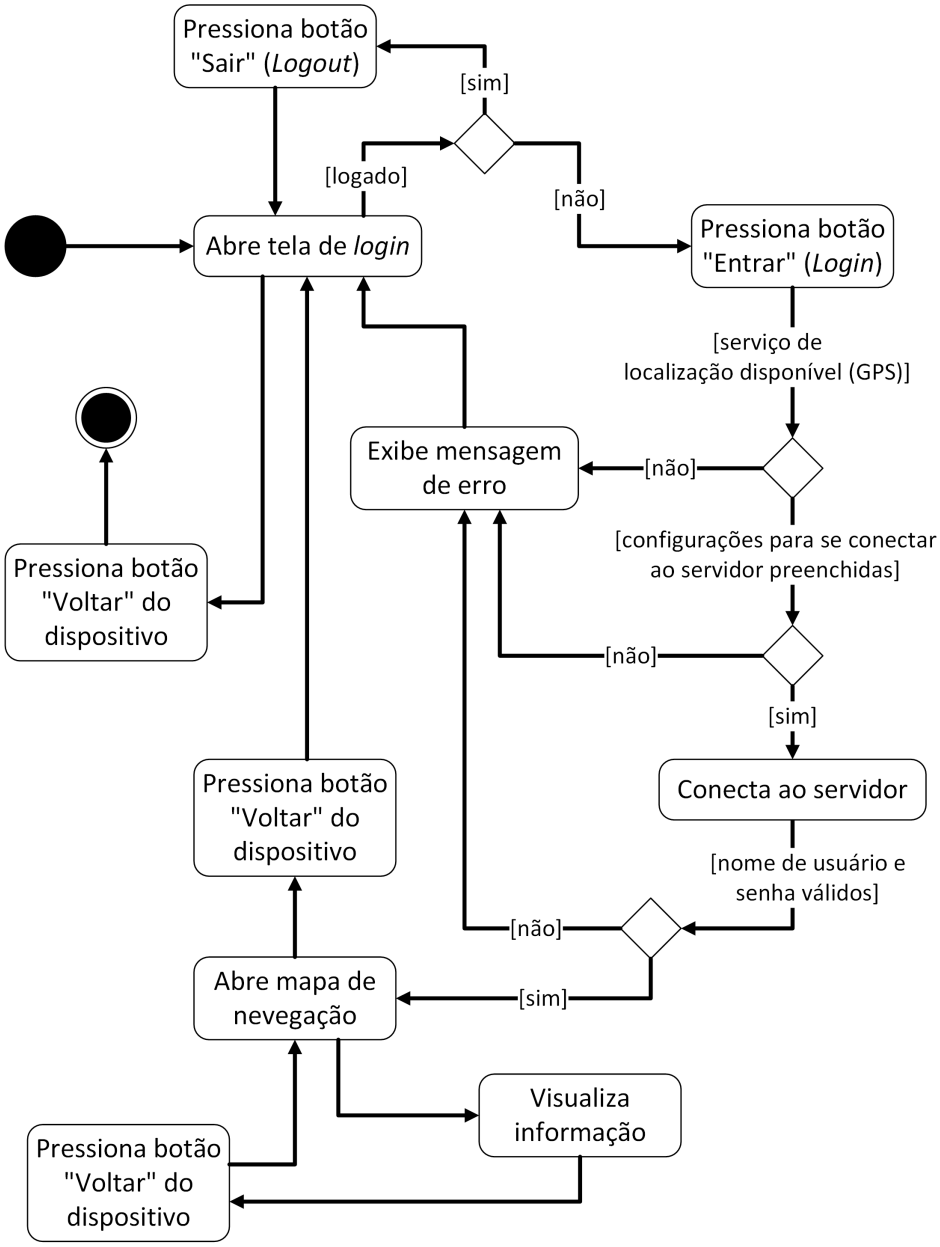


Figura 11 Diagrama de atividade

4.5 Controle de permissão

O objetivo da aplicação é fornecer informações georreferenciadas aos usuários, acerca da localização geográfica de cada um dos mesmos, em tempo real. Entretanto, no caso de uma empresa ou corporação, onde há hierarquia, geralmente há a necessidade de se controlar o acesso às informações. Com base nessa restrição, o controle de permissão foi desenvolvido.

A ideia principal desse controle consiste apenas em associar uma permissão a cada informação da tabela de informações do banco de dados, tal como a cada usuário da tabela de usuários. Isso possibilita a criação de vários grupos de usuários, categorizando-os e restringindo o acesso de cada usuário somente às informações que lhe cabem. A permissão é um número inteiro que, quanto menor, amplia o acesso às informações (maior nível de permissão—Figura 12, na qual n é maior que 1) e, quanto maior, restringe o acesso às mesmas (menor nível de permissão). Assim, os usuários podem obter do servidor somente as informações que possuem a permissão maior ou igual à sua.

Como também pode ser visualizado na figura, o modelo do controle de permissão é hierárquico, ou seja, os usuários com nível de permissão baixo não tem acesso às informações que os usuários com nível de permissão alto tem. A diferenciação das cores das setas se deve à restrição de acesso às informações, conforme condição hierárquica.

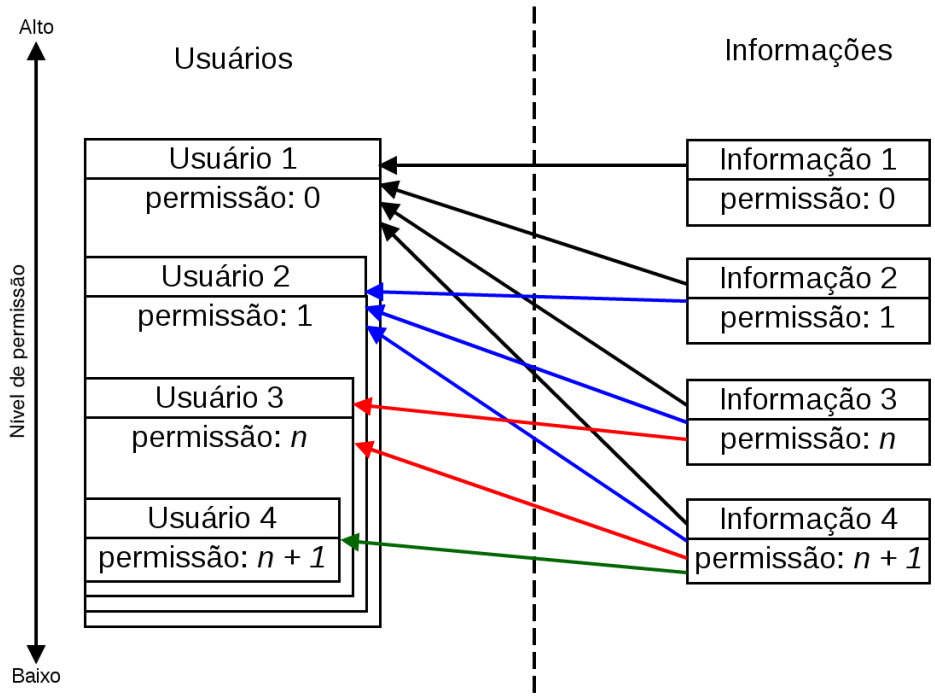


Figura 12 Informações que serão recuperadas pelos usuários de acordo com o nível de permissão

A desvantagem desse modelo está no fato de não ser possível que se tenha usuários com mesmo nível de permissão acessando informações diferentes, ou que se tenha um usuário com nível de permissão intermediário acessando informações exclusivas (que não podem ser acessadas por nenhum outro usuário).

4.6 Protocolo de comunicação

Com o intuito de fazer com que a aplicação desempenhe a sua função principal, que é fornecer aos usuários as informações georreferenciadas pertinentes às suas atividades em campo, considerou-se, inicialmente, o protocolo HTTP para a realização das requisições e recebimento das respostas. No entanto, tal proto-

colo possui cabeçalhos que aumentariam a transferência de dados—o que não é desejável—e o mesmo não permite que o servidor se comunique com os clientes sem que estes últimos façam alguma requisição—o que também aumentaria a transferência de dados e limitaria a comunicação entre as partes. Sendo assim, desenvolveu-se para este trabalho um protocolo de comunicação a fim de fazer com que os *softwares* cliente e servidor pudessem se comunicar de maneira efetiva.

Segundo Kozierok (2005), um protocolo define uma linguagem e um conjunto de regras e procedimentos, que permitem que dispositivos e sistemas comuniquem entre si. Além disso, conforme Donahoo e Calvert (2009), o protocolo assegura que todos os dispositivos em uma rede estejam de acordo sobre como várias ações devem ser executadas em todo o processo de comunicação e, ainda, dita como a informação está estruturada e como a mesma será interpretada.

A Figura 13, na qual x está entre zero e n, explicita o modo como a comunicação entre os *softwares* cliente e servidor é feita. Como pode ser visualizado, toda mensagem, independente de ser uma requisição ou uma resposta, contém um identificador (“[id da requisição]”) e um código (“[código da mensagem]”). No entanto, o corpo da mensagem é opcional.

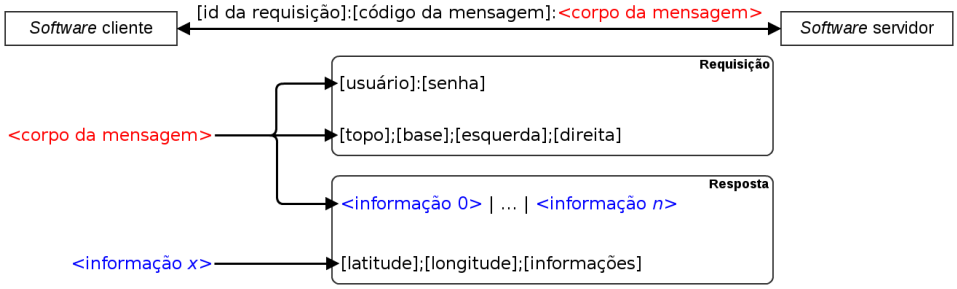


Figura 13 Modelo de requisição e resposta

Ainda, a mensagem é inteiramente transferida em modo texto e os campos da mesma (“[id da requisição]”, “[código da mensagem]” e “<corpo da mensagem>”) separam-se pelo caractere ‘**:**’ (dois-pontos).

O identificador da mensagem (“[id da requisição]”) é um número inteiro, positivo, que se inicia em 0 (zero) e que é incrementado a cada requisição feita pelo *software* cliente, permitindo que esse último, ao receber uma resposta do *software* servidor, saiba a qual requisição essa última se refere.

As figuras 14 e 15, na qual *n* é maior que 1, esclarecem, respectivamente e de modo complementar à Figura 13, a estrutura de uma requisição, efetuada pelo *software* cliente, e a estrutura de um resposta, enviada pelo *software* servidor.

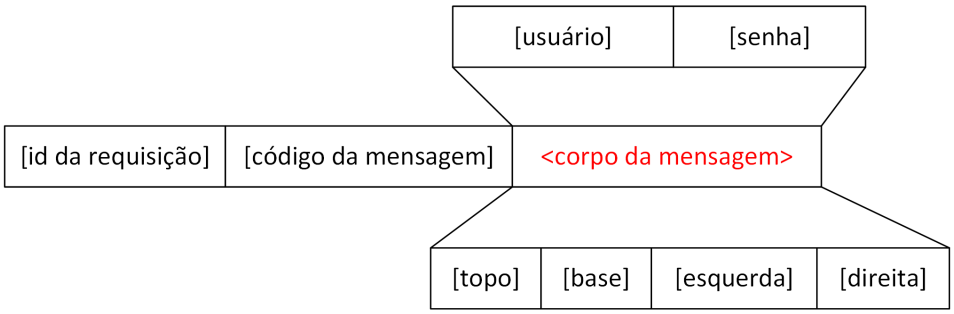


Figura 14 Estrutura da requisição

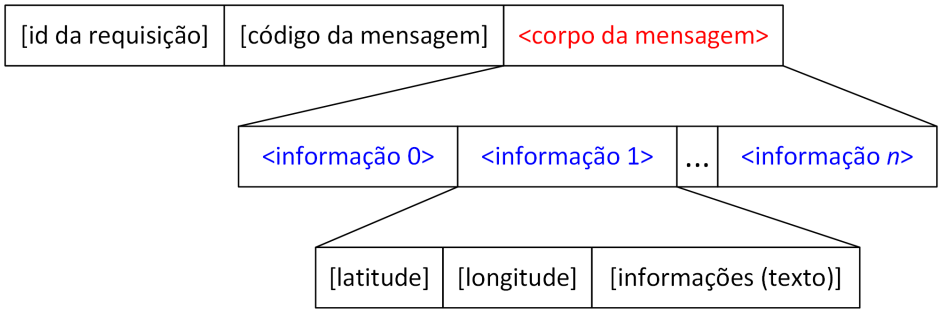


Figura 15 Estrutura da resposta

O início do protocolo de comunicação rege o processo de *login* da aplicação (Figura 16, na qual o número 300 indica requisição de *login*, o número 200 indica *login* efetuado com sucesso e *x* é um número inteiro maior ou igual a zero). Basicamente, esta parte do protocolo consiste no envio do nome de usuário e da senha (separados pelo caractere '**:**'—dois-pontos) pelo *software* cliente, e no recebimento de um número inteiro como resposta (identificado por "[código da mensagem]" nas figuras 13 e 15, e "[código da resposta]" na Figura 16). Dependendo do número inteiro enviado como resposta pelo *software* servidor, o *software* cliente pode abrir o mapa e iniciar a comunicação (*login* efetuado com sucesso), ou exibir na tela do dispositivo móvel a mensagem de erro correspondente à resposta recebida.

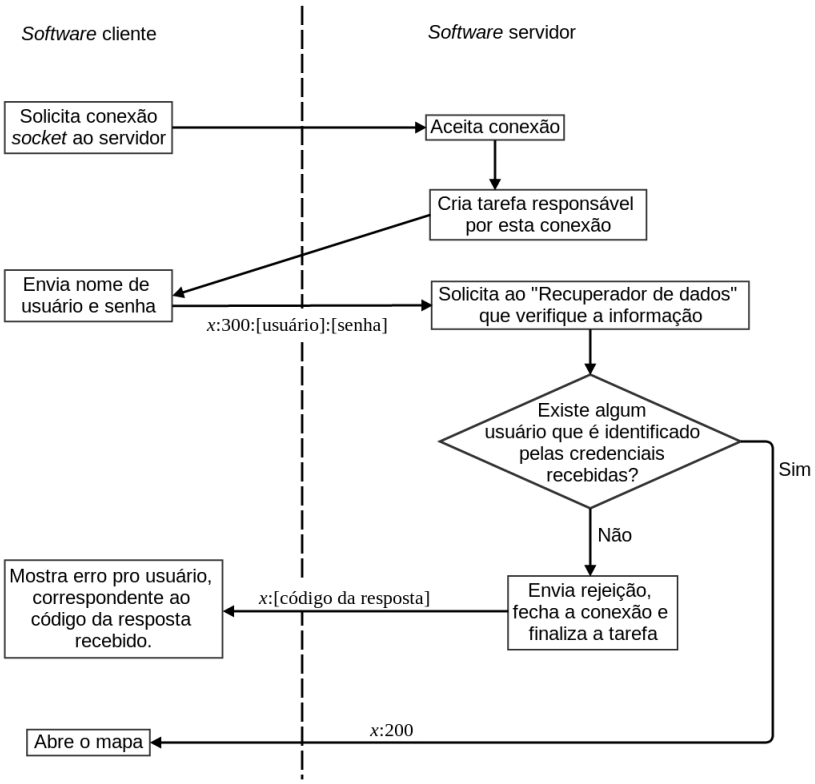


Figura 16 Início do protocolo de comunicação (processo de *login*)

Após o *login* bem sucedido do usuário, iniciam-se as requisições dinâmicas. Essa parte do protocolo consiste na solicitação de informações pelo *software* cliente, no envio das informações pelo *software* servidor e no envio do *ACK*[27] pelo *software* cliente, confirmando o recebimento das mesmas (Figura 17, na qual o número 500 indica requisição de informação, o número 600 indica resposta com informações, o número 111 indica *ACK* e *x* é um número inteiro maior ou igual a zero).

[27] Abreviação para *ACKnowledge*. É uma mensagem de confirmação, que indica que o *software* cliente recebeu as informações com sucesso.

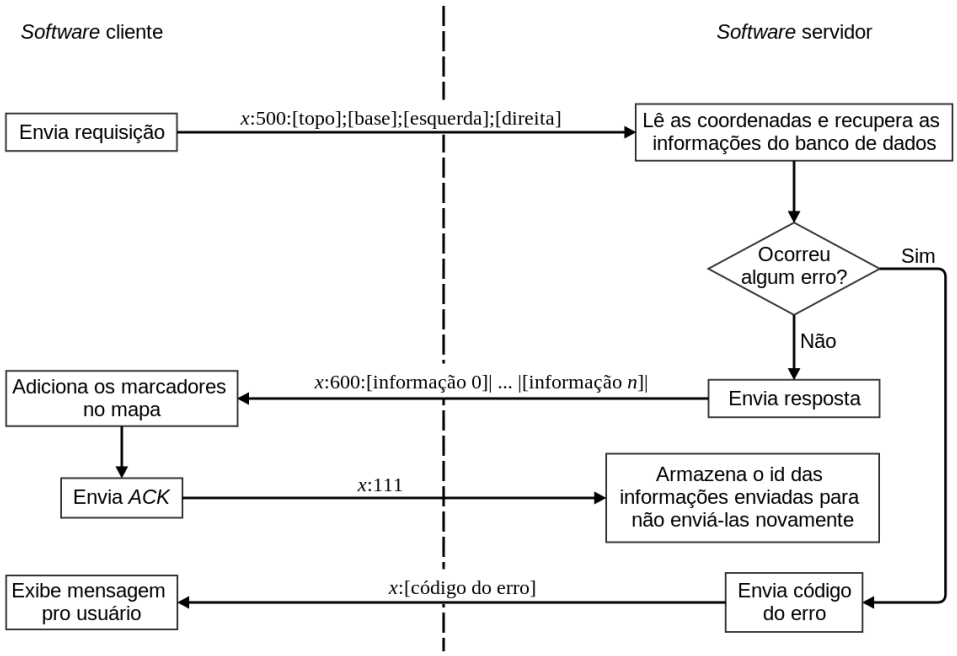


Figura 17 Restante do protocolo (requisições dinâmicas)

Como pode ser visto nas figuras 13, 14 e 17, o *software* cliente realiza uma requisição, enviando as coordenadas da área do mapa a ser visualizada na tela do dispositivo móvel (na ordem: latitude[28] da borda superior, latitude da borda inferior, longitude[28] da borda esquerda e longitude da borda direita—Figura 18)[29] separadas pelo caractere '**;**' (ponto-e-vírgula). De maneira análoga, o *software* servidor responde às requisições enviando as informações solicitadas separadas pelo caractere 'l' (*pipe*). Cada informação é composta pela latitude da sua coordenada geográfica, pela longitude da mesma coordenada, e pelo texto (a informação propriamente dita), separados pelo caractere '**;**' (ponto-e-vírgula) (figuras 13 e 15).

[28] Ver apêndice A.

[29] Representadas por "[topo]", "[base]", "[esquerda]" e "[direita]" respectivamente.

Figura 18 Área do mapa visualizada pelo usuário na tela do dispositivo

A Figura 19, na qual *x* é um número inteiro maior ou igual a zero, mostra de forma concisa o que ocorre nessa parte do protocolo, responsável pela solicitação e obtenção das informações.

O Logo M Estilizado é uma marca registrada da Motorola Trademark Holdings, LLC.

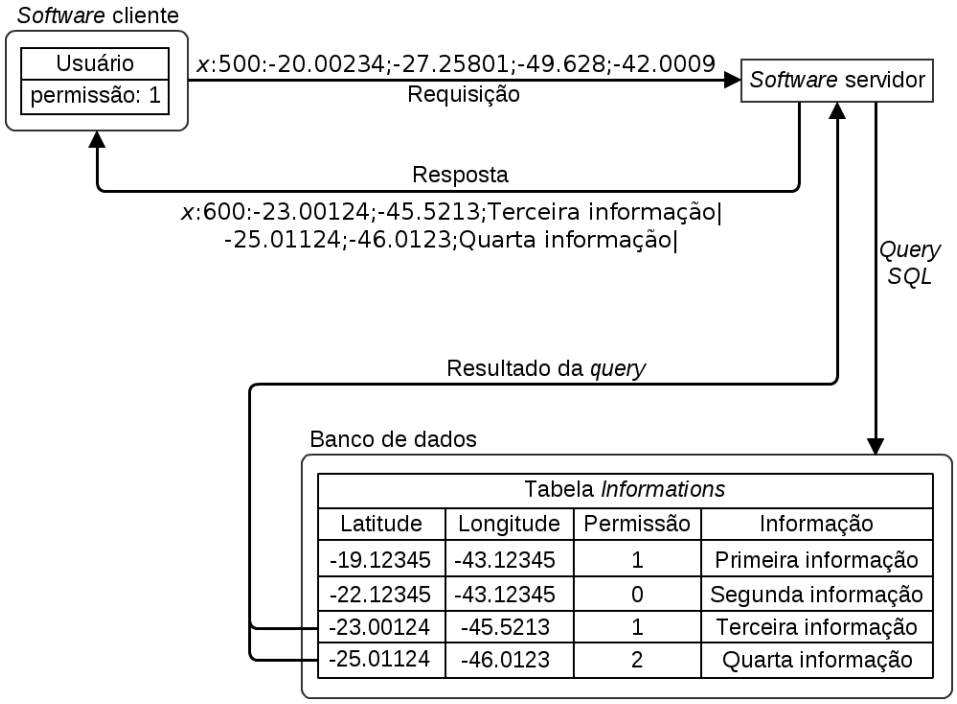


Figura 19 Exemplo de solicitação e recebimento de informações

As requisições repetem-se continuamente enquanto os *softwares* cliente e servidor estiverem conectados entre si. Precisamente, o programa cliente fará uma nova requisição ao servidor toda vez que a área do mapa a ser visualizada pelo usuário for atualizada (motivo das requisições serem dinâmicas). Além disso, para reduzir o tráfego de dados, o *software* servidor mantem uma lista dos identificadores[30] das informações enviadas para o usuário, que foram confirmadas (através do *ACK*) pelo *software* cliente. Assim, o *software* servidor não enviará tais informações novamente.

[30] Ver seção 4.7.

4.7 Diagrama de banco de dados

Como pode ser visto na Figura 20, procurou-se manter simples a modelagem do banco de dados. Apesar da base de dados possuir apenas duas tabelas que não tem qualquer relação entre si, essa modelagem atende perfeitamente o objetivo da aplicação de fornecer informações para os usuários em tempo real, com a possibilidade de restringir o acesso a tais informações de acordo com a permissão de cada usuário.

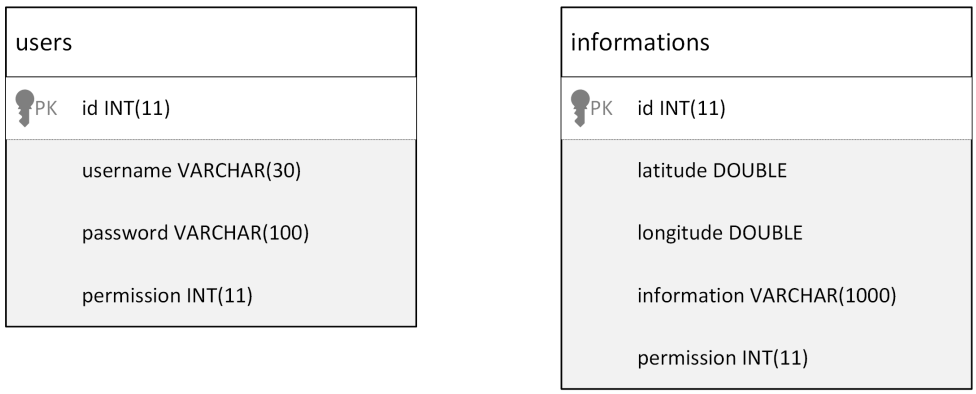


Figura 20 Diagrama de banco de dados

Tanto a tabela “informations” quanto a tabela “users” possuem um identificador (campo “id”) do tipo inteiro (INT), que armazena números de até onze dígitos. Esses identificadores são as chaves primárias de ambas as tabelas e são auto-incrementados sempre que novos dados são adicionados às mesmas, o que faz com que cada registro em tais tabelas possua um identificador único.

Ainda, a tabela “users” possui: o campo “username”, do tipo VARCHAR, para se armazenar nomes de usuário de até trinta caracteres; o campo “password”, também do tipo VARCHAR, onde o *hash*[31] da senha, de até 100 caracteres, é

[31] Função utilizada para gerar sequências alfanuméricas de tamanho fixo, a partir de entradas de texto de tamanho variável.

armazenado; e o campo “permission”, do tipo INT, onde é armazenado um número inteiro de até onze dígitos que representa o nível de permissão do usuário. Já a tabela “informations” possui: o campo “latitude”, do tipo DOUBLE, para se armazenar um número decimal que representa a latitude associada ao registro; o campo “longitude”, também do tipo DOUBLE, onde armazena-se um número decimal que representa a longitude associada ao registro; o campo “information”, do tipo VARCHAR, onde o texto da informação em si, de até mil caracteres, é armazenado; e o campo “permission”, análogo ao campo homônimo da tabela “users”.

4.8 Aplicativo móvel (*software* cliente)

4.8.1 Mapa de navegação

Para a exibição do mapa de navegação no aplicativo a ser executado no dispositivo móvel, duas opções foram consideradas: o Google Maps e o OpenStreetMap. O OpenStreetMap é um “mapa gratuito e editável” considerado “a solução cartográfica, disponível publicamente, mais extensa do mundo”. Seu sistema é colaborativo, alimentado pelos próprios usuários, ou seja, qualquer pessoa pode modificar o mapa. O Google Maps por outro lado, é um serviço de cartografia gratuito e bem estabelecido no mercado, oferecido pelo Google, e que não é colaborativo, o que lhe atribui maior confiabilidade (ALLIANCE FOR YOUTH MOVEMENTS, 2011, tradução nossa).

Para a aplicação aqui proposta, o Google Maps foi escolhido como o serviço responsável pela exibição do mapa de navegação para o usuário por oferecer maior confiabilidade, possuir uma biblioteca que funciona com o Android de modo fluente, dispor de uma documentação ampla e ser utilizado por muitos desenvolvedores em todo o mundo. Além disso, a natureza colaborativa do OpenStreetMap,

que faz com que o mesmo não ofereça qualquer garantia de fidelidade ou exatidão em seus mapas (ALLIANCE FOR YOUTH MOVEMENTS, 2011), foi outro ponto que contou a favor para a escolha do Google Maps.

Para que tivesse controle sobre a confiabilidade dos aplicativos publicados para a plataforma Android, o Google adotou o sistema de manifesto. Dessa forma, qualquer aplicativo, para funcionar no Android, deve conter um arquivo chamado "AndroidManifest.xml", que é responsável por apresentar ao sistema operacional informações essenciais sobre o programa. Assim, o sistema operacional tem controle sobre quais recursos do dispositivo móvel o aplicativo pode ou não acessar (GOOGLE INC., 2013b).

Basicamente, para se utilizar o Google Maps é necessário que uma chave seja gerada no Google APIs Console[32] e, para isso, é necessário que o desenvolvedor tenha uma conta no Google. Com a chave gerada associada à conta do desenvolvedor, o Google tem como rastrear o aplicativo em sistemas, como o Google Play Store™, e analisar como o aplicativo utiliza os recursos do Google, como os servidores do Google Maps (GOOGLE INC., 2013a). Em seguida, tal chave deve ser adicionada ao arquivo "AndroidManifest.xml". Desse modo, o Google tem controle de quais aplicativos estão devidamente cadastrados para utilizar os seus serviços. O processos de criação e utilização da chave e instalação e uso da biblioteca Google Play Services, para se ter acesso ao Google Maps no dispositivo móvel, estão detalhados, em inglês, nos anexos A e B.

Para que a segunda versão da *API* do Google Maps funcionasse com dispositivos móveis rodando versões do Android anteriores à 3.0, utilizou-se a classe "SupportMapFragment" para a exibição do mapa de navegação. Esse suporte a versões anteriores se fez necessário para que o aplicativo cliente pudesse ser tes-

[32] *https://code.google.com/apis/console/*

Google Play Store é uma marca comercial do Google Inc.

tado em um dispositivo móvel rodando a versão 2.3.7 do Android, disponível para este estudo. Embora a classe "SupportMapFragment" tenha sido usada para que o aplicativo fosse retrocompatível, a versão da *API* do Google Maps utilizada depende do OpenGL® 2.0 ES™[33], que é uma *API* gráfica padrão para a exibição de gráficos tridimensionais acelerados (KHRONOS GROUP, 2013). Desse modo, é possível que o mapa, eventualmente, seja exibido com algumas distorções.

4.8.2 Arquitetura do cliente

Como pode ser visto na Figura 21, na qual a direção das setas indica a direção da comunicação entre os componentes, a simplicidade da arquitetura do *software* cliente é evidente. Há apenas o processo principal do aplicativo, responsável pela execução da interface gráfica, e uma *thread*, responsável pela comunicação persistente assíncrona com o servidor.

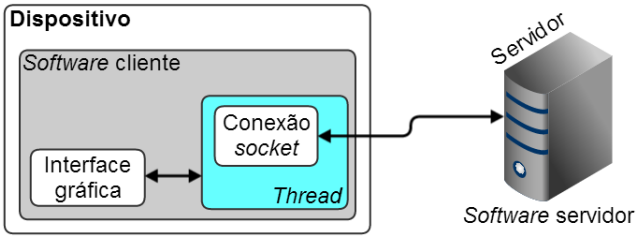


Figura 21 Arquitetura da aplicação com ênfase no dispositivo móvel

4.8.3 Diagrama de classe

O aplicativo cliente, como pode ser visto na Figura 22, é composto por quatro telas com as quais o usuário interage, identificadas por "MainActivity", "ConfigActivity", "MapActivity" e "InfoActivity". A classe "MainActivity" é a

[33] *OpenGL for Embedded Systems.*
OpenGL é uma marca registrada da Silicon Graphics, Inc.
OpenGL ES é uma marca comercial da Silicon Graphics, Inc.

tela principal do aplicativo, na qual o usuário entra com o nome de usuário e a senha para acessar o mapa de navegação. A classe “ConfigActivity” é a tela na qual o usuário tem que entrar com os dados do servidor (endereço IP[34] ou URL[35], e a porta na qual o servidor está aguardando as conexões) para que o aplicativo possa se conectar ao mesmo e obter as informações. A classe “MapActivity” é a tela do mapa de navegação, na qual o usuário vê um símbolo azul indicando a própria localização, além dos símbolos indicando onde há informações que podem ser visualizadas. E por último, a classe “InfoActivity” é a tela na qual o usuário tem acesso ao texto da informação propriamente dita, que é obtida do servidor.

[34] *Internet Protocol*.
[35] *Uniform Resource Locator.*

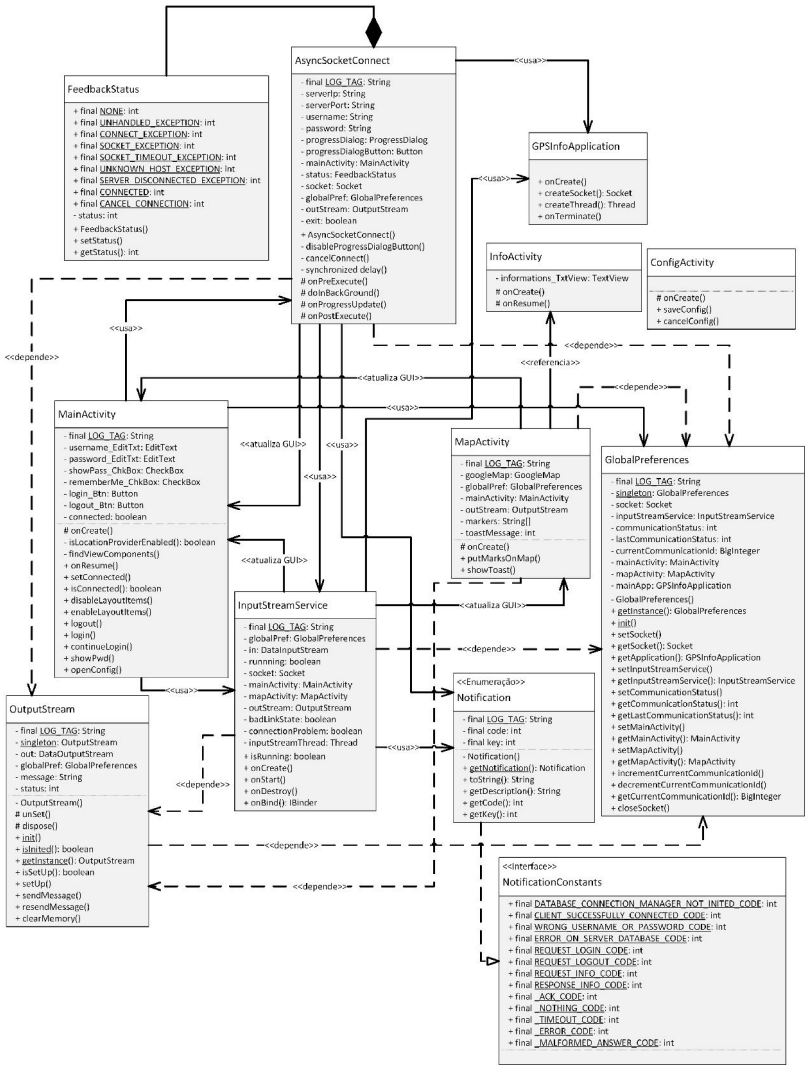


Figura 22 Diagrama de classe do *software* cliente

Como também pode ser verificado na figura, existem duas classes essenciais ao funcionamento do aplicativo das quais algumas das outras classes dependem. São elas: “GlobalPreferences” e “OutputStream”. A classe “GlobalPreferences” oferece às demais classes conhecimento sobre o estado do aplicativo. Desse modo, as demais classes verificam o estado do programa antes de executarem suas

funções. A classe "OutputStream" é responsável pelo envio das mensagens ao servidor, ou seja, sem essa classe funcionando adequadamente o aplicativo não se comunica com o servidor, o que compromete o objetivo geral da aplicação.

Outra classe também essencial ao funcionamento do aplicativo, mas sem que nenhuma outra classe dependa dela, é a "AsyncSocketConnect". Tal classe é unicamente responsável pela realização do *login* do usuário junto ao servidor. Essa classe estende a classe nativa do Android "AsyncTask", que permite que uma determinada ação seja executada de modo a não impedir o funcionamento apropriado da interface gráfica do usuário. Além disso, admite à interface gráfica exibir *feedback*[36] para o usuário, em tempo real, a respeito da ação a ser executada (o processo de *login*).

A classe "InputStreamService", análoga à classe "OutputStream" (portanto de mesma importância), é responsável pelo recebimento das mensagens do servidor. Essa classe roda como um serviço do Android, em uma *thread* dedicada, pronta para receber qualquer mensagem do servidor a qualquer momento.

Por fim, a classe "GPSInfoApplication", que estende a classe "Application" do Android, foi desenvolvida apenas para que a conexão *socket* e a *thread* do serviço "InputStreamService" fossem criadas no contexto global do aplicativo, de modo que as mesmas não são finalizados pelo Android enquanto o programa estiver em execução. Já a classe "Notification", que implementa a classe "NotificationConstants", foi desenvolvida única e exclusivamente para deixar o *software* mais organizado, centralizando nessa classe a constantes a serem utilizadas pela aplicação.

[36] "Reação a alguma coisa. = RESPOSTA, RETORNO" (PRIBERAM INFORMÁTICA, S.A., 2010a).

4.9 *Software* servidor

4.9.1 Arquitetura do servidor

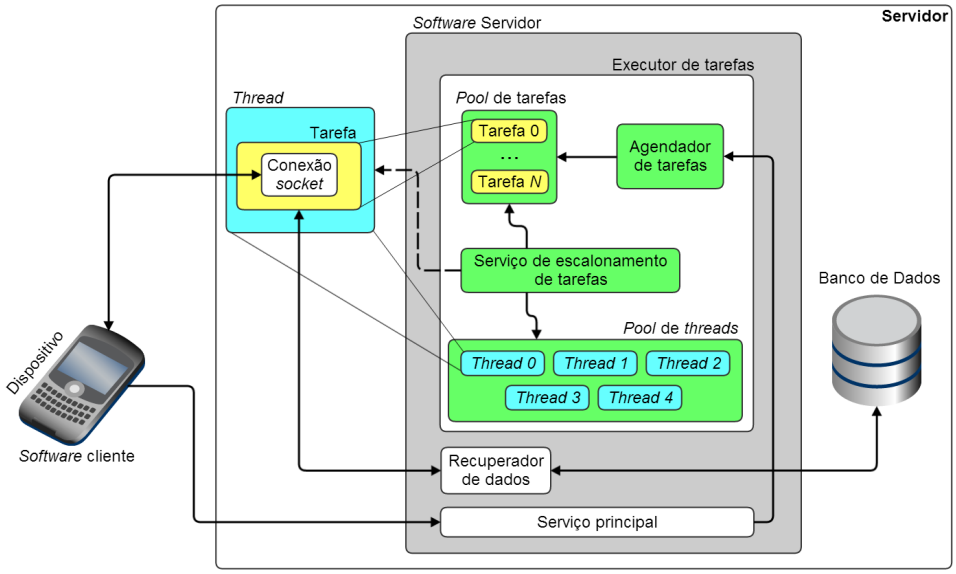


Figura 23 Arquitetura da aplicação com ênfase no servidor

Como pode ser visto na Figura 23—na qual a direção das setas indica a direção de comunicação entre os componentes, a seta pontilhada indica a função do escalonador de tarefas e *N* representa o número de clientes conectados ao servidor—a complexidade da arquitetura do *software* servidor é aparente. Quando o *software* servidor é iniciado, o "Executor de tarefas" (junto ao seus *pools* de tarefas e de *threads*, e ao "Serviço de escalonamento de tarefas") também é iniciado, seguido do "Serviço principal" (*thread* dedicada a monitorar a porta pela qual o *software* servidor receberá as conexões dos clientes). No momento em que algum dispositivo se conecta, o "Serviço principal" aceita a conexão, solicita ao "Executor de tarefas" o agendamento de uma nova tarefa (que ficará responsável

pela conexão aceita) e volta a monitorar a porta. A partir daí, o "Serviço de escalonamento de tarefas" passa a escalonar (por escalonamento *Round-robin*[37]) as tarefas do "*Pool* de tarefas" entre as *threads* do "*Pool* de *threads*" disponíveis. Ao ser escalonada, a tarefa é executada na *thread* a que foi atribuída por até meio segundo. Após esse tempo (ou após o término da execução, o que vier antes) a tarefa desocupa a *thread* e volta ao "*Pool* de tarefas" para ser escalonada futuramente. A *thread* recentemente disponibilizada retorna ao "*Pool* de *threads*", pronta para executar outra tarefa.

Durante a sua execução em uma *thread*, a tarefa está preparada para receber uma requisição através da conexão *socket*[38]. Quando isso acontece, a tarefa solicita ao "Recuperador de dados" (que por sua vez consulta o banco de dados) as informações requisitadas, e então envia a resposta através da mesma conexão.

Nos testes realizados durante o desenvolvimento da aplicação, tanto o *software* servidor quanto o banco de dados estavam hospedados na mesma máquina (identificada na Figura 23 por "Servidor"), mas poderiam estar em máquinas diferentes.

4.9.2 Diagrama de classe

Diferente do programa cliente, o *software* servidor não possui interface gráfica; apenas uma interface de linha de comando limitada a um único comando: "sair" ("quit" caso o idioma selecionado seja o inglês). Essa parte da aplicação é executada no servidor como um serviço, que aguarda passivamente à conexões dos clientes e os responde quando solicitado.

[37] Tipo de escalonamento preemptivo que "interrompe a execução de uma tarefa depois de um certo tempo, de modo que o escalonador possa executar outra tarefa e retomar a anterior mais tarde." (SALTZER; KAASHOEK, 2009, p. 355–356, tradução nossa).

[38] Ver subseção 4.2.

Como pode ser visto na Figura 24, a classe “DatabaseConnectionManager” é essencial ao funcionamento do *software* servidor e da aplicação como um todo, pelo fato de ser a classe que controla o acesso ao banco de dados. Outra classe igualmente importante é a “ClientHandler”, responsável por gerenciar a comunicação com o cliente. A cada novo cliente conectado ao servidor, uma nova instância dessa classe é criada como uma tarefa e adicionada à lista de tarefas[39].

[39] Ver subseção 4.1.

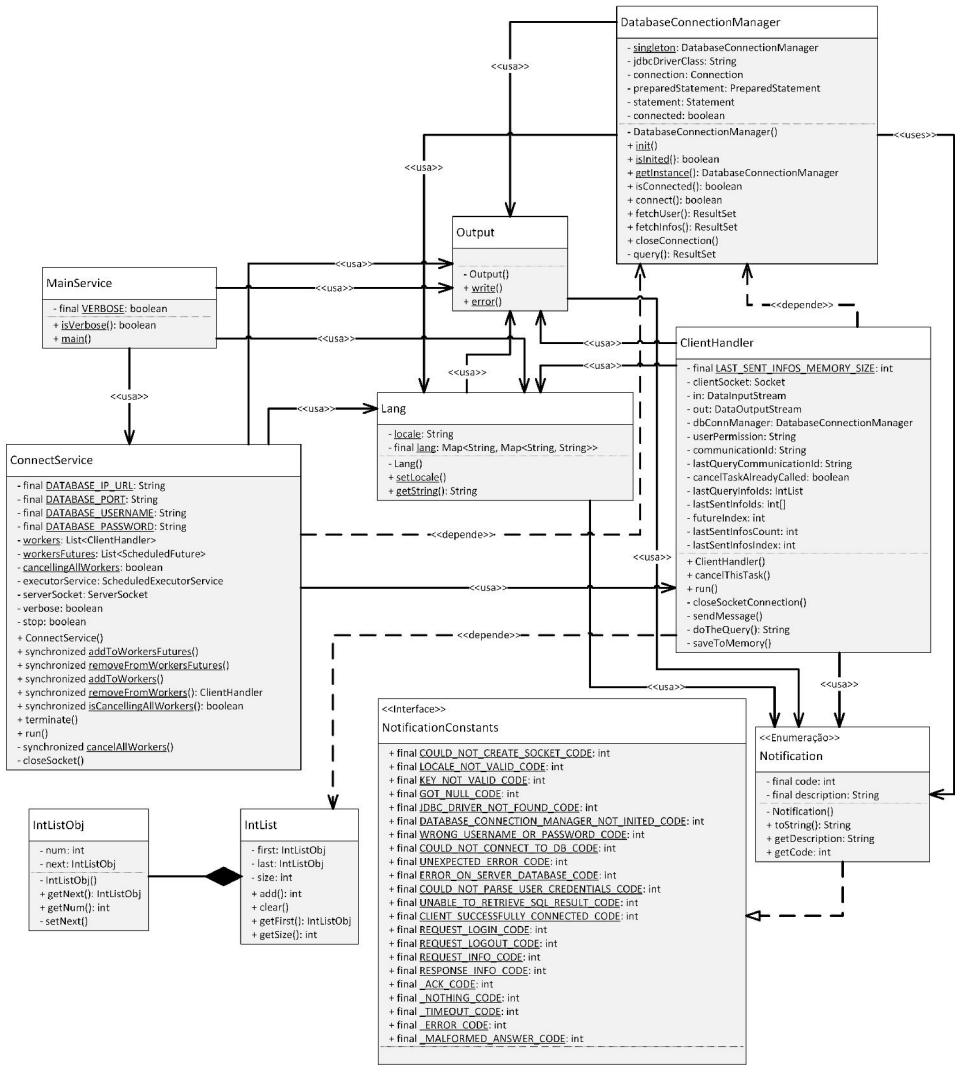


Figura 24 Diagrama de classe do *software* servidor

A classe “MainService” é o processo principal do programa, executada assim que o último é aberto. Logo após ser executada, tal classe inicia a execução da classe “ConnectService” em uma *thread* dedicada. Essa última classe aguarda passivamente pela conexão dos clientes e cria a instância da classe “ClientHandler”, responsável pela comunicação com o cliente.

A classe “Output” tem como papel exibir na interface de linha de comando informações relacionadas à execução da aplicação, para fins de depuração. Ainda, tal classe interage com a classe “Lang”, responsável por fornecer o suporte a mais de um idioma ao *software* servidor. Por fim, a classe “Notification” possui as constantes a serem utilizadas na aplicação, ou seja, corresponde à classe homônima do *software* cliente.

4.10 Exemplo prático de uso da aplicação proposta

Com o objetivo de facilitar a compreensão do funcionamento da aplicação desenvolvida, foi gerado um exemplo prático a fim de representar uma situação mais próxima possível do real. O caso hipotético proposto considera a utilização da aplicação por socorristas que trabalham no serviço de remoção (ambulância) de um hospital. Sendo assim, parte-se do princípio que o hospital tem acesso a um computador atuando como servidor, já com o Java e o MySQL devidamente instalados[40], com o *software* servidor da aplicação em execução (Figura 25) e com acesso a *internet*. Além disso, a base de dados utilizada pelo hospital é unificada, ou seja, alimentada de acordo com as entradas e saídas no mesmo e em outros hospitais do município.

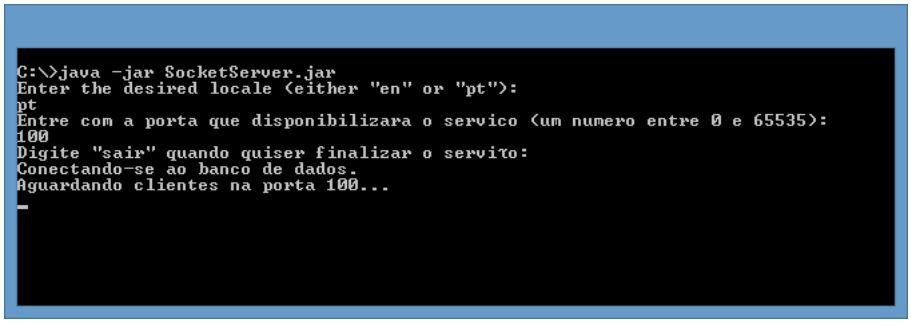


Figura 25 *Software* servidor em execução, aguardando conexões dos clientes

[40] Ver subseção 3.2.

Para que o aplicativo, que roda no dispositivo móvel, funcione adequadamente com o servidor, é preciso configurar o primeiro para que as informações sejam acessadas. Como pode ser visto na Figura 26, é necessário preencher o IP ou endereço web do servidor e a porta na qual o *software* servidor aguarda pelas conexões dos clientes (no caso desse exemplo a porta número 100). O campo "Tempo limite para conectar" indica o tempo máximo que o aplicativo aguardará pela resposta do servidor e, quando esse tempo expira, o aplicativo pergunta ao usuário o que fazer em seguida (tentar conectar novamente ou cancelar).

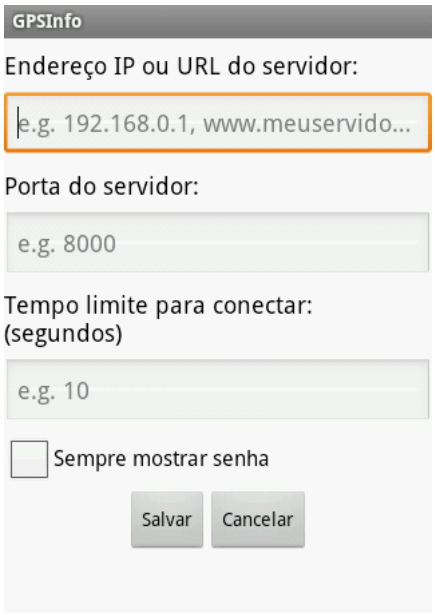


Figura 26 Tela de configuração do aplicativo

Na situação hipotética, o referido hospital recebe uma ligação de um homem, sozinho em casa, que está com fortes dores no peito e na nuca. A atendente o comunica que uma equipe irá ao local e convoca dois socorristas para o serviço. Um dos socorristas, que estava próximo à atendente, prontamente abre o aplicativo

em seu dispositivo móvel e realiza o *login*[41] para não ter que fazê-lo quando estiver dentro da ambulância. O processo de *login* está representado na Figura 27, na qual (1) e (2) ilustram a inserção de nome de usuário e senha; (3), (4), (5) e (6), o processo de autenticação da aplicação; e (7), a exibição do mapa após a confirmação do *login*.

Figura 27 Processo de *login*

Para visualizar o local de partida, onde os socorristas convocados se encontram, o portador do dispositivo móvel pressiona o botão de localização, identificado na figura acima pelo rótulo (a). A partir disso, os socorristas entram na ambulância e vão em direção ao local do chamado para atender o paciente (Figura

[41] "Processo de ligação a uma rede protegida que permite ao usuário acessar um sistema informático mediante a introdução da sua identificação e senha." (PRIBERAM INFORMÁTICA, S.A., 2010c).

28). Na figura, a seta azul indica a direção e sentido do movimento da ambulância, tal como a sua localização geográfica. A ordem cronológica das imagens que ilustram o percurso até a casa do paciente está de acordo com o rótulo das mesmas.

À medida que os socorristas percorrem o trajeto até o local do chamado, ícones aparecem na tela do dispositivo—(a), (b) e (c), indicando, em tempo real, de quais locais há informação disponível no servidor. Isso ocorre pelo fato do destino não ser previamente definido no aplicativo.

Sendo assim, ao chegaram no local de destino—ícone de rótulo (c), os socorristas percebem que existem informações disponíveis e, a partir disso, pressionam o ícone (que funciona como um botão), o que faz com que um balão que dá acesso às informações apareça (Figura 29). Por fim, ao pressionar esse balão, que também funciona como um botão, os socorristas tem acesso às informações relacionadas àquele local, o que faz com que os mesmos tenham uma base de conhecimento para auxiliá-los a decidir quais procedimentos realizar.

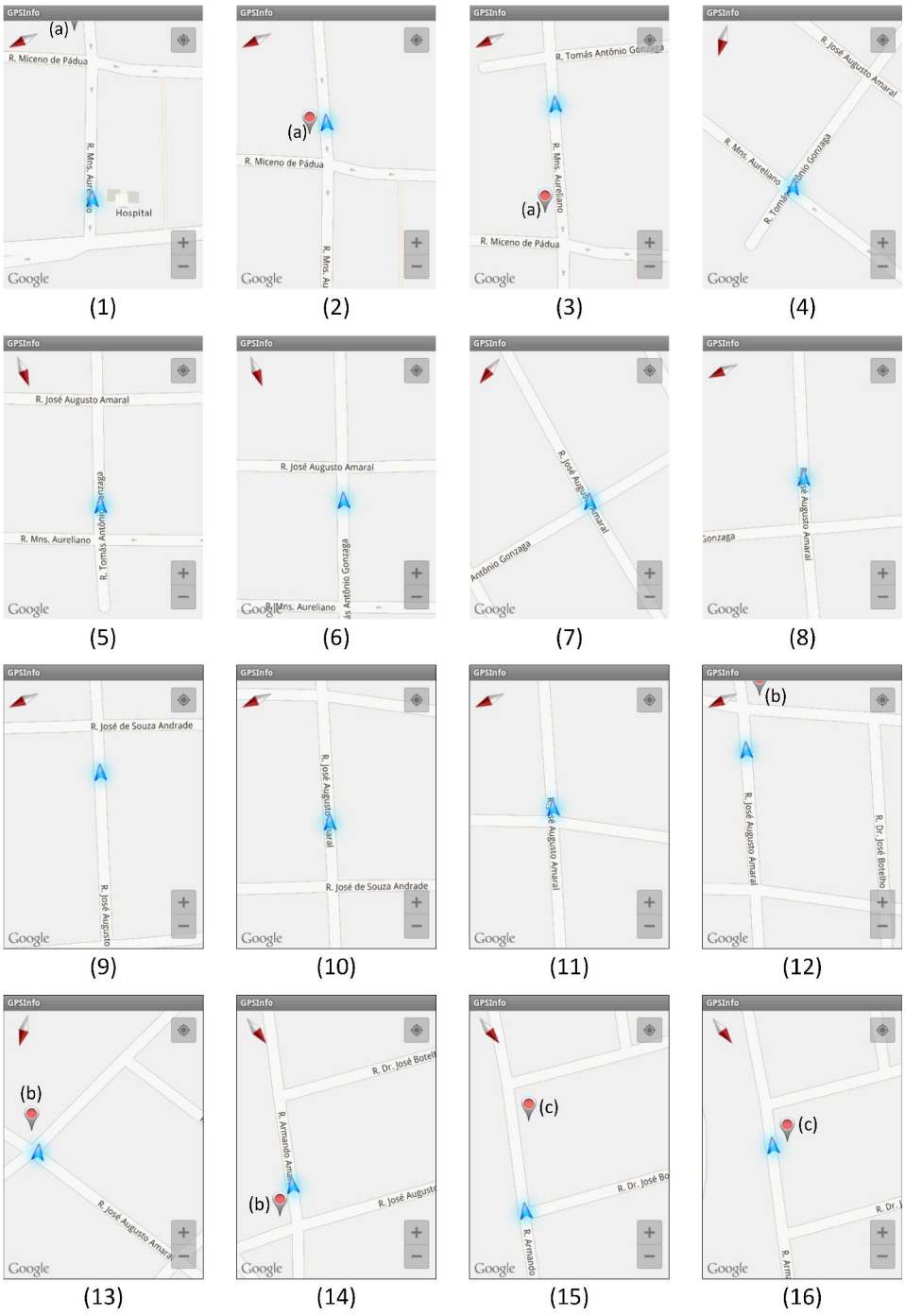


Figura 28 Percurso até a casa do paciente

Figura 29 Balão de acesso às informações e as informações em si

5 CONCLUSÃO

Este trabalho se propôs a apresentar o protótipo de uma aplicação, baseada em mapas e em geolocalização, para a visualização de informações georreferenciadas em tempo real, a ser utilizada como uma ferramenta de auxílio em trabalhos de campo e/ou tomadas de decisão. Com base no desenvolvimento e nos resultados apresentados e discutidos, conclui-se que a aplicação desenvolvida mostra-se viável para o auxílio em trabalhos de campo e/ou tomadas de decisão.

5.1 Características da aplicação

A aplicação foi desenvolvida considerando-se as características pré-atentivas[42,43] da percepção humana de modo que, aproveitando-se do matiz e da densidade, por exemplo, a atenção do usuário é direcionada aos ícones que sobrepõem o mapa de navegação e que indicam em quais locais há informação disponível para ser visualizada (Figura 30).

Tais características pré-atentivas tornam a identificação das informações na tela do dispositivo móvel mais rápida, o que interfere significativamente em diferentes situações de uso prático nas quais a velocidade de percepção desses dados é essencial, como, por exemplo, em eventos emergenciais associados a órgãos de segurança e saúde.

[42] Características que são detectadas rapidamente pelo sistema visual humano.
[43] Ver seção 2.

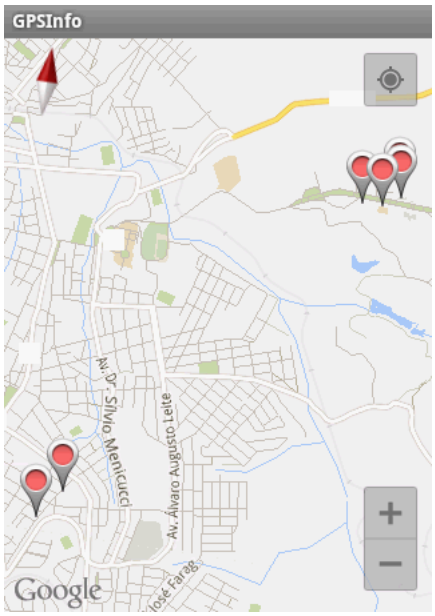


Figura 30 Ícones que indicam a presença de informações a serem visualizadas

Considerou-se, também, a interatividade que é viabilizada tanto pela plataforma Android quanto pela *API* do Google Maps, o que faz com que a aplicação desenvolvida esteja em consonância com a definição de visualização da informação segundo Chen (2004) e Card, Mackinlay e Shneiderman (1999)[43], ou seja, uma ferramenta para dar suporte à exploração visual e interativa dos dados pelo usuário de forma a amplificar a cognição do mesmo através da visualização e compreensão desses dados. Além disso, ter-se projetado a aplicação como móvel, aliado ao poder de processamento dos dispositivos móveis atuais, que possibilitam a realização de tarefas complexas, permite que a mesma seja utilizada como uma ferramenta de auxílio à tomada de decisão em qualquer lugar e a qualquer momento, conforme melhor convier para o usuário, como afirmam Burigat e Chittaro (2008) e Payne et al. (2009). Ainda, o fato dos dispositivos móveis afetarem significativamente a natureza e a disponibilidade de dados georreferenciados, associado ao

fato de 85% de toda informação gerada mundialmente possuir um componente espacial (endereço, código postal, coordenada geográfica, etc), conforme os autores supracitados, torna oportuno o fornecimento de informações para o usuário que possam auxiliá-lo em trabalhos de campo. Sendo assim, a aplicação atinge o objetivo de ser uma ferramenta de auxílio ao trabalho de campo e/ou tomada de decisão, dado que a visualização da informação permite maior rigor e exatidão na tomada de decisões, minimiza o esforço cognitivo necessário para a tomada das mesmas e fornece conhecimento para justificá-las (BETTMAN; LUCE; PAYNE, 1998; DULL; TEGARDEN, 1999; TARANTINO, 2000).

5.2 Desenvolvimento

Durante o seu desenvolvimento, a utilização da ferramenta Eclipse em conjunto com o Android SDK e o ADT[44] se mostrou eficiente, uma vez que a depuração de erros, durante a execução do aplicativo no dispositivo móvel, foi facilitada. Além disso, a utilização da linguagem de programação Java para o desenvolvimento de toda a aplicação favoreceu o processo de programação, dado que, além de ser a linguagem de programação padrão para o desenvolvimento de aplicativos para a plataforma Android, a comunicação entre os programas cliente e servidor pôde ser feita de forma simplificada.

Embora o Android tenha sido escolhido como a plataforma alvo a ser utilizada nos dispositivos móveis, o ciclo de vida dos processos em execução no sistema operacional é complexo e singular, o que conduziu a dificuldades na criação e no gerenciamento de *threads* no dispositivo.

[44] *Android Development Tools.*

5.3 Limitações

A aplicação apresenta algumas limitações quanto à sua utilização, como por exemplo a dependência da presença de *internet* no dispositivo móvel, uma vez que, sem acesso à *internet*, a API do Google Maps não exibe o mapa de navegação para o usuário. Além disso, a dependência da API do Google Maps dos aplicativos Google Maps e Google Play Services limita a utilização da aplicação, dado que sem esses aplicativos instalados no dispositivo móvel a aplicação não funciona corretamente, e afeta a velocidade da sua execução, uma vez que tais aplicativos consomem muitos recursos, como memória RAM[45] e processamento. Ainda, a segunda versão da API do Google Maps—utilizada neste trabalho, visto que o uso da primeira versão não é mais permitido pelo Google—não opera de modo fluente nas versões do Android anteriores à 3.0, posto que, nessas versões, o mapa é renderizado na tela com alguns erros. Além disso, a aplicação foi desenvolvida para as versões do Android iguais ou superiores à versão 2.3.3.

5.4 Comparação com outras aplicações

Ao comparar a aplicação com ferramentas similares da mesma área, é possível apresentar as vantagens reais da primeira frente às tecnologias existentes para a visualização de informações, como pode ser visto no Quadro 1. O sistema CIVIL[46] foi projetado para o planejamento de soluções de situações de emergência e a sua utilidade é clara, porém o mesmo não é móvel, ou seja, é necessário que se tenha acesso a um computador de mesa. Além disso, o mesmo não é sensível ao contexto, o que atrasa o processo de planejamento, dado que o usuário tem que encontrar o local correto no mapa antes de iniciar qualquer análise da situação. O

[45] *Random Access Memory*.
[46] Ver seção 2.

MoViSys, em contra partida, é um aplicativo móvel e sensível ao contexto, que exibe para o usuário informações ao seu redor em tempo real. Contudo, embora o MoViSys funcione de maneira similar à da aplicação desenvolvida neste estudo, obtendo informações do banco de dados à medida que a posição geográfica do usuário é atualizada, o mesmo não permite a exibição de informações na forma de texto. Ainda, a ferramenta proposta apresenta uma vantagem adicional em relação às aplicações anteriores, que consiste na possibilidade de atualização de informações, em tempo real, por parte do servidor no dispositivo móvel do usuário sem que este tenha feito qualquer requisição, o que é viabilizado pela comunicação entre o *software* cliente e o servidor através de uma conexão persistente. Por fim, a existência de um *software* servidor, ausente nos sistemas aqui comparados, implica em uma menor perda de desempenho da aplicação quando utilizada simultaneamente por um elevado número de usuários.

	Baseado em mapas	Móvel	Sensível ao contexto	Recupera informações de um banco de dados	Arquitetura cliente-servidor	Informações em forma de texto
CIVIL	X					
MoViSys	X	X	X	X		
GPSInfo	X	X	X	X	X	X

Quadro 1 Comparativo de funcionalidades

5.5 Considerações do exemplo prático

O cenário proposto gera um questionamento: por que usar a aplicação móvel deste trabalho ao invés do sistema de comunicação via rádio, atualmente em uso, na situação do referido exemplo prático? Inicialmente, é importante frisar que o rádio, apesar de cumprir as funções para as quais é designado, está defasado frente ao avanço da tecnologia e, portanto, é conveniente que seu uso seja revisto.

A primeira vantagem da aplicação frente ao método atual é que esta permite que os dados sejam acessados por várias equipes de socorristas simultanea-

mente e com menor possibilidade de ocorrência de mal entendidos, uma vez que o rádio está sujeito a interferência externa, o que pode comprometer a qualidade do som e o entendimento das informações.

Neste contexto, é possível que haja uma integração entre dois ou mais órgãos, o que, na situação hipotética, pode ser exemplificado pelo acesso ao banco de dados unificado dos hospitais pela polícia, de modo que, no caso de crimes que envolvam alguma forma de dano físico, os policiais tenham acesso às informações do crime simultaneamente aos socorristas. Além do mais, os socorristas não precisam se preocupar em lembrar das informações, uma vez que as tem disponíveis à mão, o que, dependendo da situação, é essencial ao atendimento. Isso fica claro no exemplo prático, no qual o próprio paciente foi quem acionou o serviço e não havia uma segunda pessoa para repassar as informações importantes para uma melhor prestação de socorro. Em determinados casos, essa ferramenta pode ser decisiva para o salvamento de uma vida.

No entanto, caso os socorristas se movam para uma área sem acesso à *internet*, o acesso às informações é consequentemente interrompido, visto que a aplicação depende da *internet* para cumprir seu papel. Sendo assim, o sistema de comunicação via rádio pode ser mantido como *backup*, sendo utilizado quando o sistema principal, a aplicação, falhar.

A aplicação pode, ainda, ser empregada em outras situações como trabalhos de campo nas áreas agrícola e florestal, manutenção de serviços, auditorias em empresas, etc. De modo geral, segundo Payne et al. (2009), a utilização de aplicações móveis como ferramentas de tomada de decisão é vantajosa em qualquer organização que empregue força de trabalho móvel, se beneficiando da coleta, análise e apresentação de informações *in loco* conforme suas necessidades.

5.6 Sugestões para trabalhos futuros

Pelo fato da comunicação entre os *softwares* cliente e servidor ser inteiramente em texto puro, portanto passível de interceptação, sugere-se a implementação de um módulo de segurança para criptografar as mensagens trocadas pelos mesmos. Além disso, propõe-se também: (1) implementação de um módulo com interface amigável responsável pelo cadastro de informações no banco de dados; (2) teste, validação e avaliação do sistema junto a usuários reais a fim de aprimorá-lo quanto à sua utilização, desempenho, usabilidade, entre outros; e (3) desenvolvimento de um ambiente colaborativo onde os próprios usuários estejam aptos a inserir ou alterar informações através de seus dispositivos móveis.

REFERÊNCIAS

AHAS, R.; AASA, A.; ROOSE, A.; MARK, U.; SILM, S. Evaluating passive mobile positioning data for tourism surveys: An estonian case study. **Tourism Management**, v. 29, n. 3, p. 469–486, 2008.

ALLIANCE FOR YOUTH MOVEMENTS. **Google Maps or OpenStreetMap**: Which Mapping Tool is Best for Your Project? 2011. Acesso em: 16 de julho de 2013. Disponível em: <http://www.movements.org/blog/entry/google-maps-or-openstreetmap/>.

ANGUISH, S.; BUCK, E.; YACKTMAN, D. **Cocoa Programming**. 1st. ed. [S.l.]: Sams Publishing, 2002. 1272 p.

BAI, X.; WHITE, D.; SUNDARAM, D. Visual intelligence density: definition, measurement, and implementation. In: **Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction**. New York, NY, USA: ACM, 2009. (CHINZ '09), p. 93–100.

BERNHARDSEN, T. **Geographic Information Systems**: An Introduction. 3rd. ed. New York: John Wiley & Sons, Inc., 2002. 448 p.

BETTMAN, J. R.; LUCE, M. F.; PAYNE, J. W. Constructive consumer choice processes. **Journal of Consumer Research**, v. 25, n. 3, p. 187–217, 1998.

BROWN, E. **Ubuntu for Tablets Joins Canonical's Convergence Crusade**. 2013. Acesso em: 19 de julho de 2013. Disponível em: <http://www.linux.com/news/embedded-mobile/mobile-linux/702049-ubuntu-for-tablets-joins-canonicals-convergence-crusade>.

BURIGAT, S.; CHITTARO, L. Geographic data visualization on mobile devices for user's navigation and decision support activites. In: **Spatial Data on the Web**. [S.l.]: Springer, 2007. p. 261–284.

_____. Interactive visual analysis of geographic data on mobile devices based on dynamic queries. **Journal of Visual Languages & Computing**, Elsevier, v. 19, n. 1, p. 99–122, 2008.

_____. On the effectiveness of overview+detail visualization on mobile devices. **Personal Ubiquitous Computing**, Springer-Verlag, London, UK, UK, v. 17, n. 2, p. 371–385, 2013.

CARD, S.; MACKINLAY, J.; SHNEIDERMAN, B. (Ed.). **Readings in Information Visualization**: Using Vision to Think. California, USA: Morgan Kaufmann Publishers, 1999. 606 p.

CHANG, C.-S.; CHEN, T.-S.; HSU, W.-H. The study on integrating webquest with mobile learning for environmental education. **Computers & Education**, v. 57, n. 1, p. 1228–1239, 2011.

CHEN, C. **Information Visualization**: Beyond the Horizon. 2nd. ed. London: Springer, 2004. 89–142 p.

CHEN, M.-C.; CHEN, J.-L.; CHANG, T.-W. Android/osgi-based vehicular network management system. **Computer Communications**, v. 34, n. 2, p. 169–183, 2011.

CHITTARO, L. Visualizing information on mobile devices. **Computer**, v. 39, n. 3, p. 40–45, 2006.

CISCO SYSTEMS INC. **OSI (Open Source Interconnection) 7 Layer Model**. 2012. Acesso em: 16 de julho de 2013. Disponível em: <https://learningnetwork.cisco.com/docs/DOC-15624>.

DALA-ALI, B. M.; LLOYD, M. A. A.; AL-ABED, Y. The uses of the iphone for surgeons. **The surgeon**: journal of the Royal Colleges of Surgeons of Edinburgh and Ireland, v. 9, n. 1, p. 44–48, fev. 2011.

DARADKEH, M. K. Y. **Information visualisation to support informed decision-making under uncertainty and risk**. 2012. 194 f. Tese (Doctor of Philosophy) – Faculty of Environment, Society and Design, Lincoln University, Christchurch, New Zealand, 2012.

DONAHOO, M. J.; CALVERT, K. L. **TCP/IP Sockets in C**: Practical Guide for Programmers. 2nd. ed. [S.l.]: Morgan Kaufmann, 2009.

DULL, R. B.; TEGARDEN, D. P. A comparison of three visual representations of complex multidimensional accounting information. **Journal of Information Systems**, v. 13, n. 2, p. 117–131, 1999.

EL-RABBANY, A. **Introduction to GPS**: The Global Positioning System, Second Edition. [S.l.]: Artech House Publishers, 2006.

FARNAGHI, M.; MANSOURIAN, A.; TOOMANIAN, A. Integration of rendering technologies and visualization techniques to improve 3d mobile gis

applications. In: **ISPRS Workshop on quality, scale and analysis aspects of city models**. Lund, Sweden: [s.n.], 2009.

FRIENDLY, M. Milestones in the history of thematic cartography, statistical graphics, and data visualization. Acesso em: 23 de junho de 2013. 2009. Disponível em: <http://euclid.psych.yorku.ca/SCS/Gallery/milestone/milestone.pdf>.

GARGENTA, M. **Learning Android**. [S.l.]: O'Reilly Media, 2011.

GOOGLE INC. **Google Maps Android API v2**. 2013a. Acesso em: 20 de julho de 2013. Disponível em: <https://developers.google.com/maps/documentation/android/start>.

_____. **The AndroidManifest.xml File**. 2013b. Acesso em: 20 de julho de 2013. Disponível em: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.

GOUVEIA, J. T. R. de. **VisualAUThor - Uma plataforma de visualização de relações de colaboração**. 2013.

GRÖLLER, E. Insight into data through visualization. In: MUTZEL, P.; JÜNGER, M.; LEIPERT, S. (Ed.). **Graph Drawing**. [S.l.]: Springer Berlin Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2265). p. 352–366.

HARRIS, R. **Introduction to Decision Making**. 1998. Acesso em: 25 de julho de 2013. Disponível em: <http://www.virtualsalt.com/crebook5.htm>.

HEALEY, C. G.; ENNS, J. Attention and visual memory in visualization and computer graphics. **Visualization and Computer Graphics, IEEE Transactions on**, v. 18, n. 7, p. 1170–1188, 2012.

HUTCHINS, E. **Cognition in the Wild**. Cambridge, MA: MIT Press, 1995.

HUTOROWICZ, H. de.; ADLER, B. F. Maps of primitive peoples. **Bulletin of the American Geographical Society**, American Geographical Society, v. 43, n. 9, p. 669–679, 1911.

IDC CORPORATE USA. **Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year, According to IDC**. 2013. Acesso em: 30 de abril 2013. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS23946013>.

INTERNATIONAL CARTOGRAPHIC ASSOCIATION. **Directory 2011–2015**. 2011. Acesso em: 09 de julho de 2013. Disponível em: <http://icaci.org/files /documents/reference_docs/2011-2015_directory.pdf>.

JUN, E.; LANDRY, S.; SALVENDY, G. A visual information processing model to characterize interactive visualization environments. **International Journal of Human-Computer Interaction**, v. 27, n. 4, p. 348–363, 2011.

JUNG, C. F. **Metodologia Aplicada a Projetos de Pesquisa: Sistemas de Informação & Ciência da Computação**. Rio de Janeiro – RJ: Axcel Books do Brasil Editora, 2004.

KHRONOS GROUP. **OpenGL ES – The Standard for Embedded Accelerated 3D Graphics**. 2013. Acesso em: 20 de julho de 2013. Disponível em: <http:/ /www.khronos.org/opengles/>.

KOZIEROK, C. M. **The TCP/IP Guide**: A Comprehensive, Illustrated Internet Protocols Reference. [S.l.]: No Starch Press, 2005.

LEWIS, G. Cartography, history of. In: SMELSER, E. in C. J.; BALTES, P. B. (Ed.). **International Encyclopedia of the Social & Behavioral Sciences**. Oxford: Pergamon, 2001. p. 1501–1509.

MASHABLE, INC. **Google Maps**. 2006. Acesso em: 10 de julho de 2013. Disponível em: <http://mashable.com/category/google-maps/>.

MONARES, A.; OCHOA, S. F.; PINO, J. A.; HERSKOVIC, V.; RODRIGUEZ-COVILI, J.; NEYEM, A. Mobile computing in urban emergency situations: Improving the support to firefighters in the field. **Expert Systems with Applications**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 2, p. 1255–1267, 2011.

MORESI, E. **Metodologia de Pesquisa**. Brasília, DF: Universidade Católica de Brasília – Pró-reitoria de Pós-Graduação – Programa de Pós-Graduação Stricto Sensu em Gestão do Conhecimento e Tecnologia da Informação, 2003.

NYMAN, R. **Firefox OS devices officially released!** 2013. Acesso em: 19 de julho de 2013. Disponível em: <https://hacks.mozilla.org/2013/07 /firefox-os-devices-officially-released/>.

ORACLE CORPORATION. **Java Timeline**. 2012. Acesso em: 13 de julho de 2013. Disponível em: <http://oracle.com.edgesuite.net/timeline/java/>.

PAYNE, L. D.; ZLATKOV, D. T.; JERNIGAN, J. M.; SCARBROUGH, J. M. A location-aware mobile system for on-site mapping and geographic data management. In: **Proceedings of the 10th ACM conference on SIG-information technology education**. New York, NY, USA: ACM, 2009. (SIGITE '09), p. 166–172.

PRIBERAM INFORMÁTICA, S.A. *"feedback"*. In: **Dicionário Priberam da Língua Portuguesa**. 2010a. Acesso em: 19 de julho de 2013. Disponível em: <http://www.priberam.pt/dlpo/dlpo.aspx?pal=feedback>.

_____. *"in loco"*. In: **Dicionário Priberam da Língua Portuguesa**. 2010b. Acesso em: 25 de julho de 2013. Disponível em: <http://www.priberam.pt/dlpo/dlpo.aspx?pal=in%20loco>.

_____. *"login"*. In: **Dicionário Priberam da Língua Portuguesa**. 2010c. Acesso em: 25 de julho de 2013. Disponível em: <http://www.priberam.pt/dlpo/dlpo.aspx?pal=login>.

RUCHTER, M.; KLAR, B.; GEIGER, W. Comparing the effects of mobile computers and traditional approaches in environmental education. **Computers & Education**, Elsevier Science Ltd., Oxford, UK, UK, v. 54, n. 4, p. 1054–1067, 2010.

SALTZER, J. H.; KAASHOEK, M. F. **Principles of Computer System Design**: An Introduction. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

SHNEIDERMAN, B. The eyes have it: a task by data type taxonomy for information visualizations. In: **Visual Languages, 1996. Proceedings., IEEE Symposium on**. [S.l.: s.n.], 1996. p. 336–343.

SILVA, E. L. da; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 3ª. ed. Florianópolis: Laboratório de Ensino a Distância da UFSC, 2001. 121 p.

TARANTINO, L. Advances in information visualisation: recent outcomes. **The Knowledge Engineering Review**, Cambridge University Press, New York, NY, USA, v. 15, n. 4, p. 405–410, 2000.

VAZ, A.; MATOS, P. P. de; AFONSO, A. P.; CARMO, M. B. Movisys – um sistema de visualização para dispositivos móveis. **Actas Interacção 2008**, p. 165–174, 2008.

VOGEL, L. **Google Maps Android API v2 - Tutorial**. 2013. Acesso em: 20 de julho de 2013. Disponível em: <http://www.vogella.com/articles/AndroidGoogleMaps/article.html>.

WARE, C. **Information Visualization**: Perception for Design. 2nd. ed. Amsterdam: Morgan Kaufmann Publishers, 2004.

WOODWARD, D.; HARLEY, J. B. (Ed.). **The History of Cartography: Cartography in prehistoric, ancient, and medieval Europe and the Mediterranean**. Chicago: University of Chicago Press, 1987.

WU, A.; ZHANG, X.; CONVERTINO, G.; CARROLL, J. M. Civil: support geo-collaboration with information visualization. In: **Proceedings of the ACM 2009 international conference on Supporting group work**. New York, NY, USA: ACM, 2009. (GROUP '09), p. 273–276.

ZHENG, L.; LI, M.; WU, C.; YE, H.; JI, R.; DENG, X.; CHE, Y.; FU, C.; GUO, W. Development of a smart mobile farming service system. **Mathematical and Computer Modelling**: An International Journal, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 54, n. 3-4, p. 1194–1203, 2011.

APÊNDICE A – Coordenadas Geográficas

Conforme Bernhardsen (2002), as coordenadas geográficas na superfície da Terra são a latitude, medida em graus a norte ou sul do plano equatorial, e a longitude, também medida em graus a leste ou oeste do meridiano de Greenwich. Segundo o mesmo autor, posições representadas em latitude e longitude são apenas relativas; distâncias e áreas devem ser calculadas através de geometria esférica e do raio da Terra.

O autor também afirma que as linhas formadas por pontos de igual longitude são conhecidas como meridianos, enquanto que as linhas formadas por pontos de igual latitude são conhecidas como paralelos de latitude ou apenas paralelos. Além disso, o autor menciona que essas linhas geralmente são reproduzidas nos mapas em intervalos, como a cada 5 ou 10 graus.

No caso do Google Maps, utilizado no *software* cliente, a distinção da direção das coordenadas (norte, sul, leste ou oeste) é definida pelo sinal das mesmas. Para a latitude, um valor de coordenada positivo indica que a mesma está ao norte do plano equatorial, ao passo que um valor negativo indica a estância da mesma ao sul do mesmo plano. Analogamente, um valor de coordenada positivo para a longitude indica que esta última está a leste do meridiano de Greenwich, enquanto que um valor negativo indica sua estada a oeste do mesmo meridiano.

ANEXO A – Google Maps Android *API* v2

Getting Started

Before you can begin working with the API, you will need to download the API and ensure that you have a Google Maps Android API v2 key. Both the API and the key are freely available.

Overview

Creating a new Android application that uses the Google Maps Android API v2 requires several steps. Many of the steps outlined in this section will only have to be performed once, but some of the information will be a handy reference for future applications. The overall process of adding a map to an Android application is as follows:

1. Download and configure the Google Play services SDK. The Google Maps Android API is distributed as part of this SDK.
2. Obtain an API key. To do this, you will need to register a project in the Google APIs Console, and get a signing certificate for your app.
3. Specify settings in the Application Manifest.
4. Add a map to a new or existing Android project.
5. Publish your application!

You may wish to begin by looking at some sample code, which is included with the Google Play services SDK.

Getting the Google Maps Android API v2

The API is distributed as part of the Google Play services SDK, which you can download with the Android SDK Manager. To use the Google Maps Android API v2 in your app, you will first need to install the Google Play services SDK. To learn how to install the package, see the Google Play services documentation.

As a prerequisite, you need to install the Android SDK. To learn how to do this, see Installing the SDK.

The Google Maps API Key

> **Note:** The Google Maps Android API v2 uses a new system of managing keys. Existing keys from a Google

> Maps Android v1 application, commonly known as MapView, will not work with the v2 API.

To access the Google Maps servers with the Maps API, you have to add a Maps API key to your application. The key is free, you can use it with any of your applications that call the Maps API, and it supports an unlimited number of users. You obtain a Maps API key from the Google APIs Console by providing your application's signing certificate and its package name. Once you have the key, you add it to your application by adding an element to your application's manifest file `AndroidManifest.xml`.

Understanding the process of registering your application and obtaining a key requires some knowledge of Android's publishing process and requirements. In summary, all Android applications must be signed with a digital certificate for which you hold the private key. Because digital certificates are unique, they provide a simple way of uniquely identifying your app. This makes them useful for tracking your application in systems such as Google Play Store, and for tracking your application's use of resources such as the Google Maps servers.

> **Note:** Refer to the Android Guide Signing Your Applications for more information regarding digital certificates.

Maps API keys are linked to specific certificate/package pairs, rather than to users or applications. You only neec one key for each certificate, no matter how many users you have for an application. Applications that use the same certificate can use the same API key. However, the recommended practice is to sign each of your applications with a different certificate and get a different key for each one.

Obtaining a key for your application requires several steps. These steps are outlined here, and described in detail in the following sections.

1. Retrieve information about your application's certificate.
2. Register a project in the Google APIs Console and add the Maps API as a service for the project.
3. Once you have a project set up, you can request one or more keys.
4. Finally, you can add your key to your application and begin development.

Displaying certificate information

The Maps API key is based on a short form of your application's digital certificate, known as its **SHA-1 fingerprint**. The fingerprint is a unique text string generated from the commonly-used SHA-1 hashing algorithm. Because the fingerprint is itself unique, Google Maps uses it as a way to identify your application.

To display the SHA-1 fingerprint for your certificate, first ensure that you have the certificate itself. You may have two certificates:

- **Debug certificate**: The Android SDK tools generate this certificate automatically when you do a "debug" build from the command line, or when you build and run a project from Eclipse without exporting it as a released application. The certificate is only for use with an application that you're testing; you can't publish an app that's signed with a debug certificate. The debug certificate is described in more detail in the section Signing in Debug Mode in the Android Developer Documentation. You can generate an API key from this certificate, but only use the key for testing, never for production.
- **Release certificate**: The Android SDK tools generate this certificate when you do a "release" build with either `ant` program or Eclipse. You can also generate this certificate using the `keytool` program. This

certificate can be used with an app you release to the world. Once you have the correct certificate for your needs, you can display its SHA-1 fingerprint using the `keytool` program.

- For more information about Keytool, see the documentation athttp://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

▸ Displaying the debug certificate fingerprint

▸ Displaying the release certificate fingerprint

Creating an API Project

Once you have your signing certificate fingerprint, create or modify a project for your application in the Google APIs Console and register for the Maps API.

To get a project and register for the API:

1. In a browser, navigate to the Google APIs Console.
   - If you haven't used the Google APIs Console before, you're prompted to create a project that you use to track your usage of the Google Maps Android API. Click **Create Project**; the Console creates a new project called **API Project**. On the next page, this name appears in the upper left hand corner. To rename or otherwise manage the project, click on its name.
   - If you're already using the Google APIs Console, you will immediately see a list of your existing projects and the available services. It's still a good idea to use a new project for Google Maps Android API, so select the project name in the upper left hand corner and then click **Create**.
2. You should see a list of APIs and services in the main window. If you don't, select **Services** from the left navigation bar.
3. In the list of services displayed in the center of the page, scroll down until you see **Google Maps Android API v2**. To the right of the entry, click the switch indicator so that it is **on**.
4. This displays the Google Maps Android API Terms of Service. If you agree to the terms of service, click the checkbox below the terms of service, then click **Accept**. This returns you to the list of APIs and services.

You're now ready to get a Maps API key.

Obtaining an API Key

If your application is registered with the Google Maps Android API v2 service, then you can request an API key. It's possible to register more than one key per project.

To get the key:

1. Navigate to your project in the Google APIs Console.
2. In the left navigation bar, click **API Access**.
3. In the resulting page, click **Create New Android Key...**.
4. In the resulting dialog, enter the SHA-1 fingerprint, then a semicolon, then your application's package name. For example:

```
BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;
com.example.android.mapexample
```

5. The Google APIs Console responds by displaying **Key for Android apps (with certificates)** followed by a forty-character API key, for example:

   ```
   AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0
   ```

6. Copy this key value. You will use it in the next step.

Adding the API Key to your application

The final step is to add the API key to your application. It goes in your application's manifest, contained in the file `AndroidManifest.xml`. From there, the Maps API reads the key value and passes it to the Google Maps server, which then confirms that you have access to Google Maps data.

To add the key to your application:

1. In `AndroidManifest.xml`, add the following element as a child of the `<application>` element, by inserting it just before the closing tag `</application>`:

   ```
   <meta-data
       android:name="com.google.android.maps.v2.API_KEY"
       android:value="API_KEY"/>
   ```

   substituting your API key for *API_KEY*. This element sets the key`com.google.android.maps.v2.API_KEY` to the value *API_KEY* and makes the API key visible to any `MapFragment` in your application.

2. Save `AndroidManifest.xml` and re-build your application.

Specify settings in the Application Manifest

An Android application that uses the Google Maps Android API needs to specify the following settings in its manifest file, `AndroidManifest.xml`:

- Permissions that give the application access to Android system features and to the Google Maps servers.
- Notification that the application requires OpenGL ES version 2. External services can detect this notification and act accordingly. For example, Google Play Store won't display the application on devices that don't have OpenGL ES version 2.
- The Maps API key for the application. The key confirms that you've registered with the Google Maps service via the Google APIs Console.

This section describes each of these settings and how to add them to `AndroidManifest.xml`.

Specifying permissions

Set permissions by adding `<uses-permission>` elements as children of the `<manifest>` element. The syntax is:

```
<uses-permission android:name="permission_name"/>
```

For example, to request the Internet permission, add:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Besides permissions required by other parts of your application, you must add the following permissions in order to use the Google Maps Android API:

- `android.permission.INTERNET` Used by the API to download map tiles from Google Maps servers.
- `android.permission.ACCESS_NETWORK_STATE` Allows the API to check the connection status in order to determine whether data can be downloaded.
- `com.google.android.providers.gsf.permission.READ_GSERVICES` Allows the API to access Google web-based services.
- `android.permission.WRITE_EXTERNAL_STORAGE` Allows the API to cache map tile data in the device's external storage area.

The following permissions are recommended, but can be ignored if your application does not access the user's current location, either programmatically, or by enabling the My Location layer.

- `android.permission.ACCESS_COARSE_LOCATION` Allows the API to use WiFi or mobile cell data (or both) to determine the device's location.
- `android.permission.ACCESS_FINE_LOCATION` Allows the API to use the Global Positioning System (GPS) to determine the device's location to within a very small area.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        />
<uses-permission android:name="com.google.android.providers.gsf.
        permission.READ_GSERVICES"/>
<!-- The following two permissions are not required to use
     Google Maps Android API v2, but are recommended. -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
        />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Requiring OpenGL ES version 2

Because version 2 of the Google Maps Android API requires OpenGL ES version 2, you must add a `<uses-feature>` element as a child of the `<manifest>` element in `AndroidManifest.xml`:

```
<uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

This notifies external services of the requirement. In particular, it has the effect of preventing Google Play Store from displaying your app on devices that don't support OpenGL ES version 2.

Add a Map

After you've added references to the Google Play services SDK, added your key and customized your Android Manifest, you can try adding a map to your application.

The easiest way to test that your application is configured correctly is to add a simple map. You will have to make changes in two files: `main.xml` and `MainActivity.java`. Please note that the code below is only useful for testing your settings in an application targeting Android API 12 or later, This code should not be used in a production application. Examples of how to add more robust code appear throughout this guide and in thesample code.

1. In `main.xml`, add the following fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>
```

2. In `MainActivity.java`, add the following code.

```
package com.example.mapdemo;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

3. Build and run your application. You should see a map. If you don't see a map, confirm that you've completed all of the steps appearing earlier in this document.

Fonte: Google Inc. (2013a).

ANEXO B – Google Maps Android *API* v2: Tutorial

1. Android Basics

The following assumes that you have already basic knowledge in Android development. Please check the **Android development tutorial** for the basics.

2. Google Maps

2.1. MapsView

Google provides via Google play a library for using Google Maps in your application. The following description is based on the Google Maps Android API v2 which provides significant improvements to the older API version.

The library provides the `com.google.android.gms.maps.MapFragment` class and the `MapView`class for displaying the map component.

You need to add additional information to your *`AndroidManifest.xml`* file to use Google Maps.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.locationapi.maps"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="17" />

    <permission
        android:name="com.vogella.android.locationapi.maps.permission.MAPS_R
ECEIVE"
        android:protectionLevel="signature" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <uses-permission android:name="com.vogella.android.locationapi.maps.perm
ission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE
" />
    <uses-permission android:name="com.google.android.providers.gsf.permissi
on.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION
" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.vogella.android.locationapi.maps.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="your_apikey" />
    </application>

</manifest>
```

2.2. MapFragment

The `MapFragment` class extends the `Fragment` class and provides the life-cycle management and the services for displaying a `GoogleMap` widget. `GoogleMap` is the class which shows the map. The`MapFragment` has the `getMap()` method to access this class.

the `LatLng` class can be used to interact with the `GoogleView` class.

2.3. Markers

You can create markers on the map via the `Marker` class. This class can be highly customized.

The following code shows an example.

```
public class MainActivity extends Activity {
  static final LatLng HAMBURG = new LatLng(53.558, 9.927);
  static final LatLng KIEL = new LatLng(53.551, 9.993);
  private GoogleMap map;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map))
        .getMap();

    if (map!=null){
      Marker hamburg = map.addMarker(new MarkerOptions().position(HAMBURG)
          .title("Hamburg"));
      Marker kiel = map.addMarker(new MarkerOptions()
          .position(KIEL)
          .title("Kiel")
          .snippet("Kiel is cool")
          .icon(BitmapDescriptorFactory
              .fromResource(R.drawable.ic_launcher)));
    }

  }
```

On the `GoogleMap` you can register a listener for the markers in your map via the`setOnMarkerClickListener(OnMarkerClickListener)` method. The`OnMarkerClickListener` class defines the `onMarkerClicked(Marker)` method which is called if a marker is clicked.

Similar to you also listen to drag events and info window clicks.

2.4. Changing the GoogleView

The `GoogleMap` can be highly customized.

The following example code is taken from the offical Google webpage.

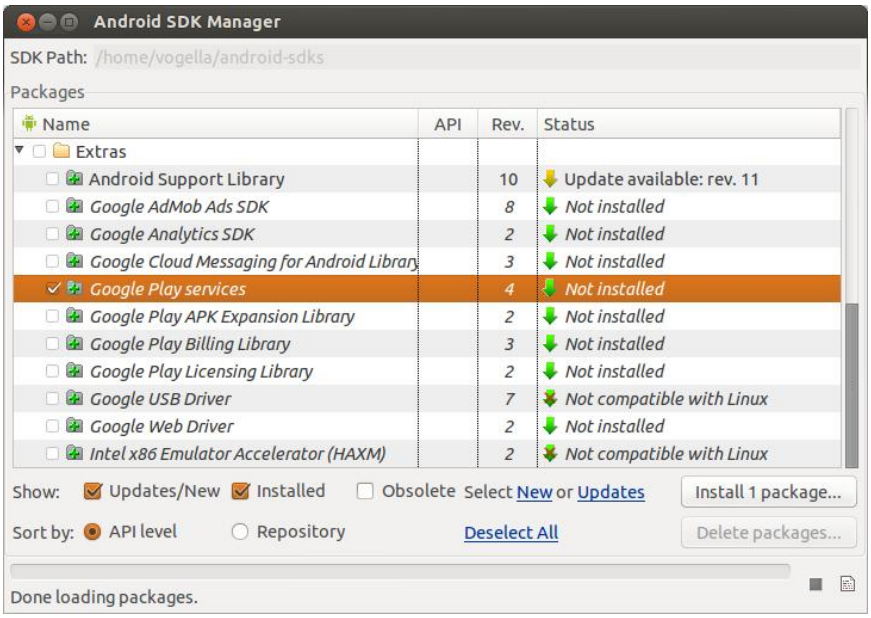
```
static final LatLng HAMBURG = new LatLng(53.558, 9.927);
static final LatLng KIEL = new LatLng(53.551, 9.993);

private GoogleMap map;
... // Obtain the map from a MapFragment or MapView.

//Move the camera instantly to hamburg with a zoom of 15.
map.moveCamera(CameraUpdateFactory.newLatLngZoom(HAMBURG, 15));

// Zoom in, animating the camera.
map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
```

2.5. Android emulator and Google Maps

Unfortunately for using the Google Maps you have to test the application on a real device as the emulator is not supported.

3. Install Google Play services

Open the *Android SDK Manager* and install *Extras → Google Play services*



Import the library which was downloaded into Eclipse
via *File → Import → Android → Existing Android Code into Workspace*.

To use this library define a library dependency in your Android project.

4. Getting the Google Map key

4.1. Overview

To use Google Maps you need to create a valid Google Maps API key. The key is free, you can use it with any of your applications that call the Maps API, and it supports an unlimited number of users.

You get this key via the so-called Google APIS Console. You have to provide your application signature key and the application package name.

This is based on the key with which you sign your Android application during deployment. During development with Eclipse, Eclipse us automatically creates and uses a *debug key*.

4.2. Creating the SHA-1 for your signature key

The Eclipse debug key for signing your application can be found in the`userhome/.android/debug.keystore` file.

To create the SHA-1 for your debug keystore you use the `keytool` command from your JDK installation pointing to the `debug.keystore` file.

```
keytool -list -v -alias androiddebugkey \
-keystore <path_to_debug_keystore>debug.keystore \
-storepass android -keypass android
```

Copy the SHA-1 output, as you need this later.

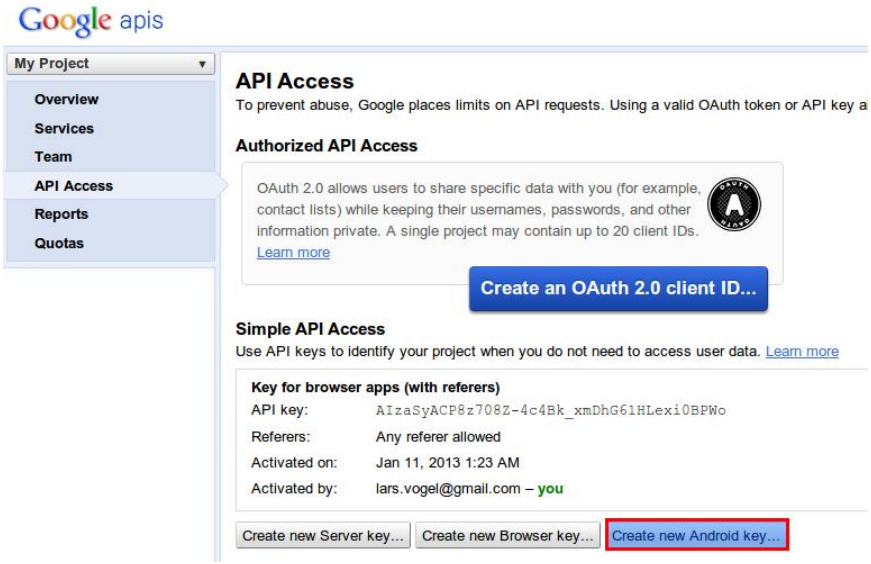
4.3. Register with the Google APIs Console

You have to register in the *Google APIs Console* that you want to use Google Maps for Android. You can reach this console via the following link: **~~Google APIs Console~~**. Select here the *Services* entry.

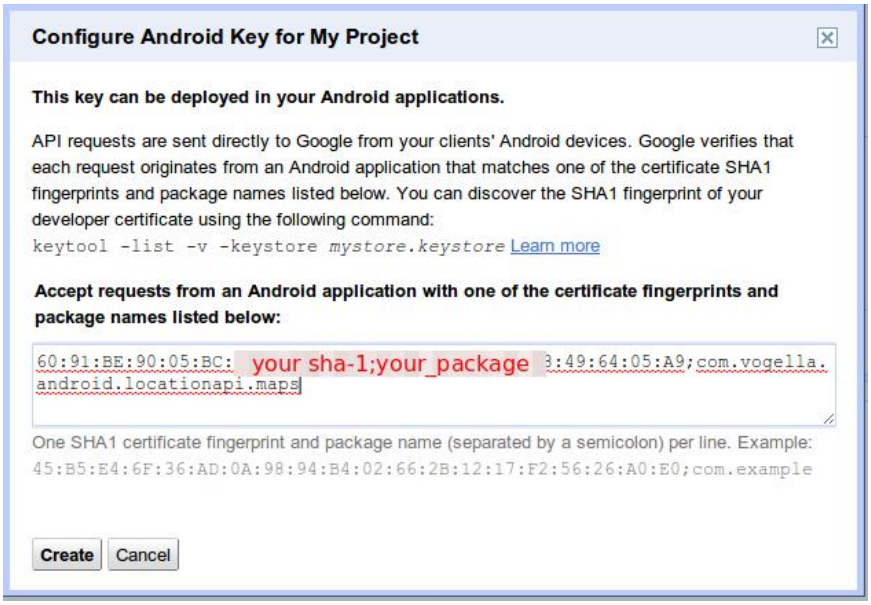
Activate the *Google Maps Android API v2*.

4.4. Create key for your application

You need later to register your application via its package in this console together with the SHA-1 fingerprint of your signature key. For this you select the entry and click on the *API Access* entry. Afterwards click on the *Create new Android key...* entry.



Enter your SHA-1 fingerprint and the package of your application separated by a semicolon. For example you can use the `com.vogella.android.locationapi.maps` package.



The procedure is described in detail in the following link: **~~Getting a Google Maps key~~**.

5. Tutorial: Google Maps

5.1. Create Project

In the following chapter we will build an Android application which shows a `GoogleMap`.

Create a new Android project called *com.vogella.android.locationapi.maps* with an *activity* called *ShowMapActivity*.

Change the *`AndroidManifest.xml`* file to the following code. Add the following permissions to your application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.locationapi.maps"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="17" />

    <permission
        android:name="com.vogella.android.locationapi.maps.permission.MAPS_R
ECEIVE"
        android:protectionLevel="signature" />

    <uses-feature
```

```
        android:glEsVersion="0x00020000"
        android:required="true" />

    <uses-permission android:name="com.vogella.android.locationapi.maps.perm
ission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE
" />
    <uses-permission android:name="com.google.android.providers.gsf.permissi
on.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION
" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.vogella.android.locationapi.maps.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="your_apikey" />
    </application>

</manifest>
```

Get a valid API key for your application and enter this key in the *`AndroidManifest.xml`* file.

5.2. Adjust layout file

In this example we use the `MapFragment`. Change your layout file to the following code.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.MapFragment" />

</RelativeLayout>
```

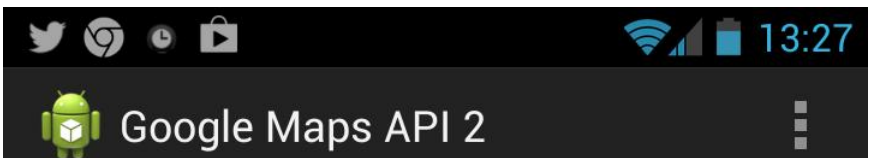
5.3. Activity

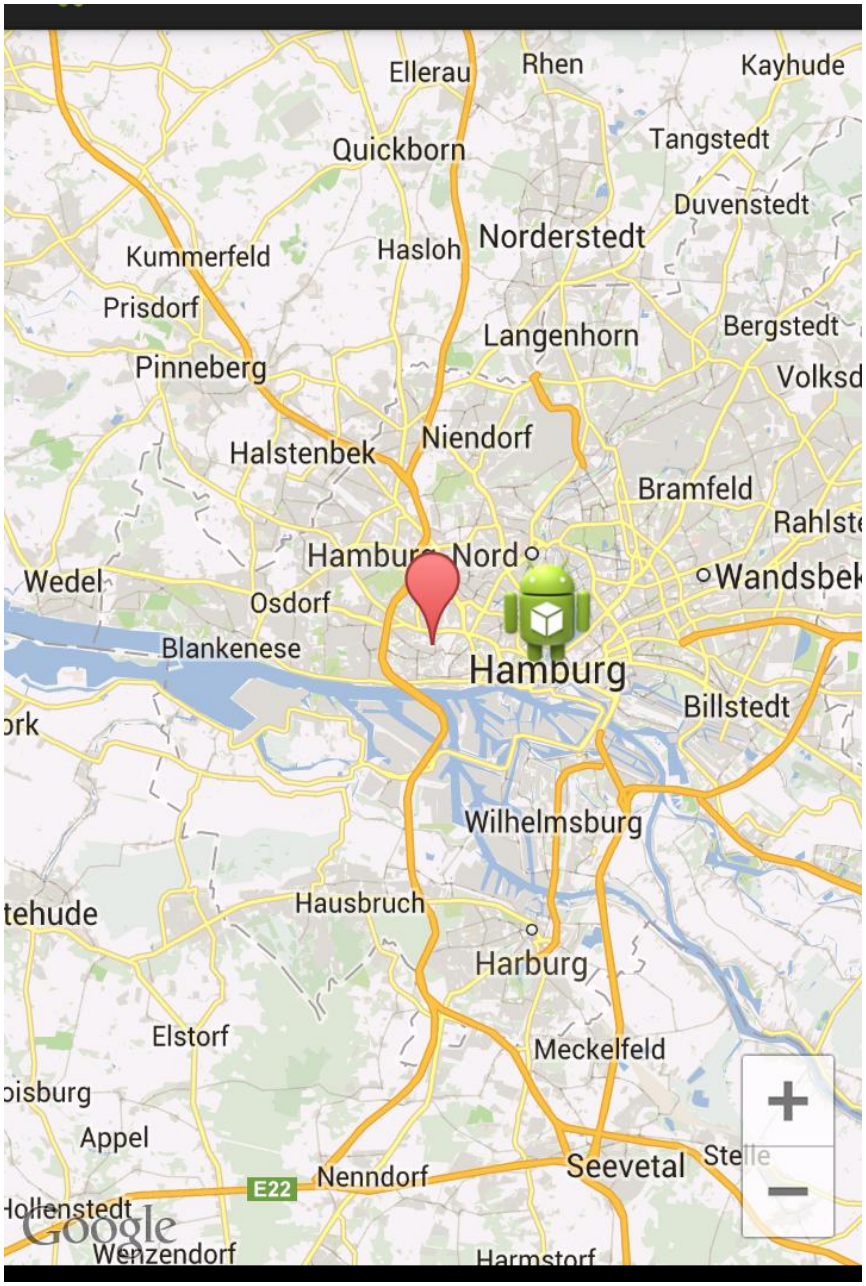
Change your *activity* to the following.

```
package com.vogella.android.locationapi.maps;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

public class MainActivity extends Activity {
  static final LatLng HAMBURG = new LatLng(53.558, 9.927);
  static final LatLng KIEL = new LatLng(53.551, 9.993);
  private GoogleMap map;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map))
        .getMap();
    Marker hamburg = map.addMarker(new MarkerOptions().position(HAMBURG)
        .title("Hamburg"));
    Marker kiel = map.addMarker(new MarkerOptions()
        .position(KIEL)
        .title("Kiel")
        .snippet("Kiel is cool")
        .icon(BitmapDescriptorFactory
            .fromResource(R.drawable.ic_launcher)));

    // Move the camera instantly to hamburg with a zoom of 15.
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(HAMBURG, 15));

    // Zoom in, animating the camera.
    map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
  }

}
```

5.4. Run and Test

Run and test your application. You should be able to zoom in and out and send new geo coordinates to your *activity* via the Emulator.





Fonte: Vogel (2013).