UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

DANIEL HOFFMANN

ULIXES MOBILE, PROTÓTIPO DE UM APLICATIVO ALIMENTADOR DE UM *FIELD SERVICE MANAGEMENT SYSTEM*

CURITIBA
2013

DANIEL HOFFMANN

ULIXES MOBILE, PROTÓTIPO DE UM APLICATIVO ALIMENTADOR DE UM *FIELD SERVICE MANAGEMENT SYSTEM*

Trabalho de Conclusão de Curso apresentada ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito para a conclusão do curso de Especialização em Java e Sistemas Móveis.

Orientador: Prof. Msc. Luiz Augusto Pelisson.

CURITIBA

2013

Ficha Catalográfica

DANIEL HOFFMANN

ULIXES MOBILE, PROTÓTIPO DE UM APLICATIVO ALIMENTADOR DE UM *FIELD SERVICE MANAGEMENT SYSTEM*

Este trabalho foi julgado e aprovado para a conclusão do curso de Especialização em Java e Sistemas Móveis da Universidade Tecnológica Federal do Paraná.

Curitiba, 08 de Agosto de 2013.

Prof.ª Dr. Marília A. Amaral
Coordenadora do Curso de Pós Graduação em Java e Sistemas Móveis

BANCA EXAMINADORA

Prof. Msc. Luiz Augusto Pelisson
Universidade Tecnológica Federal do Paraná
Orientador

Prof . Msc. Leandro Batista de Almeida
Universidade Tecnológica Federal do Paraná

Prof.ª Dr.ª Marília A. Amaral
Universidade Tecnológica Federal do Paraná

RESUMO

HOFFMANN, D. **Ulixes Mobile, Protótipo de um Aplicativo Alimentador de um Field Service Management System.** 2013. Trabalho de Conclusão de Curso (Especialização em Java e Sistemas Móveis) – Departamento Acadêmico de Informática (DAINF), UTFPR, Curitiba.

Em 2011 foi proposto como trabalho de conclusão de curso, baseado em um amplo levantamento de requisitos, um aplicativo para atender mais completamente as necessidades complexas do processo de gerenciamento de serviço de campo das empresas que necessitam de tais serviços. Contudo, durante o desenvolvimento de tal sistema o aplicativo *mobile* que é utilizado pelos técnicos para interagir com o sistema foi deixado em segundo plano visto que o objetivo principal naquele tempo era desenvolver o sistema de gerenciamento de chamados. Levando isto em consideração, este trabalho tem por propósito desenvolver um protótipo de um aplicativo para o sistema operacional *Android* que atenderá de forma mais completa os requisitos dos técnicos de campo.

Palavras-Chave

Sistemas de Gerenciamento de Força de Campo, Android, Aplicativos Móveis.

ABSTRACT

HOFFMANN, D. **Ulixes Mobile, Protótipo de um Aplicativo Alimentador de um Field Service Management System.** 2013. Trabalho de Conclusão de Curso (Especialização em Java e Sistemas Móveis) – Departamento Acadêmico de Informática (DAINF), UTFPR, Curitiba.

In 2011 was proposed as Completion of Course Work, based on an extensive survey of requirements, an application to more fully meet the needs of complex processes of management field services for companies who need such services. However, during the development of such system the mobile application that is used by technicians was left in the background as the main objective at the time was to develop the ticket management system. Taking this into account, this work has the purpose to develop a prototype of an application for the Android operation system that meet more completelly the requirements of the field technicians.

Key-words

Field Service Management System, Android, Mobile Applications.

LISTA DE SIGLAS

ACID	- *Atomic, Consistent, Isolated and Durable*
ADT	- *Android Developer Tools*
ANSI-C	- *American National Standards Institute C*
API	- *Application Programming Interface*
BLOB	- *Binary Large Object*
CLI	- *Command Line Interface*
DAO	- *Data Access Objects*
FSM	- *Field Service Management*
FSMS	- *Field Service Management System*
GPS	- *Global Positioning System*
HTTP	- *Hypertext Transfer Protocol*
JDK	- *Java Development Toolkit*
JVM	- *Java Virtual Machine*
MVC	- *Model View Control*
OHA	- *Open Handset Alliance*
RIA	- *Rich Internet Applications*
SGBD	- Sistema de Gerenciamento de Banco de Dados
SHA-1	- *Security Hash Alghorithm 1*
SQL	- *Structured Query Language*
SOAP	- *Simple Object Access Protocol*
URL	- *Uniform Resouce Locator*

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O intuito do projeto aqui descrito é desenvolver um aplicativo para dispositivo móvel *Android* que será utilizado para a interação entre o técnico de campo e o Sistema de Gerenciamento de Força de Campo *Campo (Field Service Management System) Ulixes* desenvolvido como trabalho de conclusão de curso em 2011 para a obtenção do Diploma de Tecnólogo em Sistemas para Internet pela Universidade Tecnológica Federal do Paraná.

O referido sistema armazenava os dados de solicitações técnicas de clientes, permitindo que estas fossem recuperadas posteriormente, além de controlar o fluxo de atendimento de tais solicitações.

O técnico de campo, funcionário de uma empreiteira prestadora de serviços à empresa usuária do sistema, atualiza o status dos tickets atendimento através de um aplicativo móvel, alimentando novamente o sistema *Ulixes* com informações relativas aos atendimentos.

1.1 JUSTIFICATIVA

Segundo Hoffmann e Moraes (2011):

> *"*O fluxo das atividades que compõem os serviços de campo de uma empresa reúne uma complexidade suficiente para que seja mais vantajoso permitir o seu gerenciamento por um sistema. É necessário guardar todos os dados envolvidos de forma segura, para que o atendimento possa ser prestado ao cliente no local e prazo adequados, e para que cada técnico de campo saiba o que deve ser feito. A empresa precisa também manter um histórico das solicitações de serviço — os *tickets* — tanto para remunerar adequadamente os responsáveis quanto por razões administrativas".

O sistema *Ulixes* foi concebido com esse propósito, e, apesar de convergir para o estado da arte no que se refere a tecnologias de Sistemas de Gerenciamento de Força de Campo e disponibilizar diversas funcionalidades, era incompleto por não ter um bom aplicativo para celular como interface com o usuário, no caso os técnicos de campo. Desta forma, para que todo o potencial do sistema possa ser desfrutado pelo usuário, melhorias e novas funcionalidades são necessárias ao aplicativo móvel.

1.2 OBJETIVOS

A motivação para a concepção do projeto de melhoria do Sistema *Ulixes Mobile* pode ser descrita em um conjunto de objetivos específicos sintetizados por um objetivo geral.

1.2.1 **Objetivo Geral**

Melhorar a experiência de utilização do aplicativo Ulixes Mobile pelos técnicos de campo, através da criação de um novo protótipo para este aplicativo.

1.2.2 **Objetivos Específicos**

Melhorar a interface do técnico de campo com o Sistema de Gerenciamento de Força de Campo *Ulixes*, de forma a aprimorar a usabilidade e a qualidade dos dados do sistema.

Implementar funções de geolocalização no sistema móvel, melhorando a usabilidade do sistema.

Modificar a forma da entrada dos tickets no sistema, substituindo o módulo de recebimento de tickets por SMS (*Short Message Service*) por um sistema *Web* (*Servlets* ou *Web Services*).

1.3 ESCOPO

Levando-se em consideração os objetivos levantados, a aplicação móvel deverá ser capaz de modificar os status dos *tickets* enviados para um determinado técnico de campo. Além disso, o aplicativo deverá ser capaz de traçar rotas entre os diversos locais de atendimento definidos para a jornada de trabalho do técnico.

De forma resumida, podem ser definidos como integrantes do escopo do projeto:

- Os envio e recebimento de *tickets* através de *webservices*;
- O controle de acesso aos recursos da aplicação;
- A definição de rotas através de todos os pontos de atendimento;

O *Ulixes Mobile*, e seus desenvolvedores também não são de modo algum responsáveis pelas soluções providas pelos técnicos, quando estes estão prestando suporte ao cliente; o sistema não oferece ferramentas de diagnóstico de problemas ou repositório de informações técnicas – seu papel é exclusivamente o de gerenciar dados de atendimento.

1.4 PÚBLICO ALVO

O público alvo do *Ulixes Mobile* são os técnicos de campo e seus supervisores, colaboradores de empresas que utilizarão tal sistema.

Estes usuários já têm alguma experiência com os processos pertinentes ao tratamento de chamados de clientes. Ainda assim, pretende-se criar uma interface intuitiva que facilite a migração para o *UlixesMobile*, proporcionando uma curva de aprendizado mais eficiente e possibilitando o uso da aplicação mesmo por usuários leigos.

1.5 RESTRIÇÕES E RISCOS

Para o bom funcionamento do sistema *Ulixes Mobile*, é necessário ao usuário um celular com sistema operacional *Android* 2.2 ou superior, um plano de dados 3G (*3rd Generation*) habilitado neste celular e a prévia instalação do sistema *Ulixes Mobile* neste dispositivo.

Um risco avaliado para o sistema está relacionado às restrições que possam existir no celular do técnico de campo. O sistema *Ulixes Mobile* foi extensamente testado em um celular Motorola *Milestone* utilizando uma quantidade de recursos muito pequena durante o uso, entretanto pode ter um desempenho pouco satisfatório em máquinas demasiadamente obsoletas.

1.6 METODOLOGIA.

O desenvolvimento da nova versão sistema *Ulixes Mobile* teve como base as deficiências constatadas durante os testes executados na primeira versão do aplicativo móvel desenvolvido em conjunto com o sistema *Ulixes*.

Além disso, todos os requisitos levantados em 2011 durante o desenvolvimento do sistema *Ulixes* foram levados em consideração para a idealização da nova versão do aplicativo móvel.

Utilizando-se os preceitos do desenvolvimento em espiral, a aplicação e a documentação foram feitas em ciclos, com refinamentos na documentação e na aplicação a cada ciclo. Como no desenvolvimento do sistema *Ulixes*, o desenvolvimento ágil não foi considerado uma opção, visto que se deseja produzir uma boa documentação para o projeto.

Devido às restrições de tempo, o número de ciclos de refinamento foi reduzido para o desenvolvimento do aplicativo *Ulixes Mobile*, contudo, isto não impactou na qualidade do software.

1.7 ESTRUTURA DO DOCUMENTO

Este subtópico destina-se a falar de como o documento é estruturado. O capítulo 1 tem como objetivo apresentar o projeto de modo geral, mostrando suas motivações e esboçando a maneira como o projeto foi desenvolvido.

O capítulo 2 descreve os softwares de aplicação, retornando ao conceito do qual o sistema *Ulixes* e por consequência o sistema *Ulixes Mobile* derivam.

O capítulo 3 aborda a arquitetura do sistema. O tópico começa detalhando os requisitos identificados. Logo após, descreve as tecnologias empregadas, justificando a escolha das mesmas para o projeto. Neste capítulo também são apresentados os requistos funcionais e não funcionais que levaram ao desenvolvimento da aplicação e os diagramas UML (*Unified Modeling Language*) que derivaram de tais requisitos. No final do capítulo são apresentadas as soluções desenvolvidas.

Por fim, o capítulo 4 apresenta as conclusões derivadas do projeto e que futuros projetos podem ser feitos com o que foi aprendido durante o desenvolvimento desse aplicativo. Esse capítulo é seguido pelas referências consultadas e pelo apêndice.

2 ARQUITETURA DO SISTEMA

O capítulo em questão procura descrever de que modo o sistema *Ulixes Mobile* foi estruturado. Ele cita os requisitos que orientaram as decisões de projeto, aborda as camadas que compõem a aplicação, fala das soluções de interface adotadas, e apresenta as tecnologias selecionadas para a implementação do sistema e as razões pelas quais foram escolhidas.

2.1 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Requisitos funcionais são aqueles que descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, é aquilo que descreve o que tem que ser feito pelo sistema. Já os requisitos não funcionais são aqueles que expressam como deve ser feito, em geral se relacionam com padrões de qualidade como confiabilidade, desempenho e robustez, também são muito importantes, pois definem como o *software* deverá atender suas especificações funcionais.

O processo de levantamento de requisitos foi realizado em duas etapas no ano de 2011 durante o desenvolvimento da primeira versão do sistema *Ulixes*. A primeira consistiu de uma análise do sistema de FSM (*Field Service Management*) utilizado correntemente pela companhia cujo problema este projeto se propõe a solucionar.

A segunda se deu através da realização de uma entrevista, na qual o responsável pelo suporte em primeiro nível do sistema atualmente em uso na empresa assumiu o papel de cliente. O Apêndice A contém as perguntas e respostas da entrevista, e os subtópicos a seguir reúnem a essência do que foi apurado durante o levantamento de requisitos como um todo.

Para o desenvolvimento da nova versão do sistema *Ulixes* não se considerou necessário fazer um novo levantamento de requisitos, visto que os dados coletados em 2011 ainda continuam válidos e relevantes para o desenvolvimento do sistema em questão.

2.1.1 **Requisitos Funcionais**

Tomando como base a análise de requisitos efetuada em 2011 e utilizando apenas os requisitos pertinentes ao aplicativo móvel, podemos enumerar os seguintes requisitos funcionais:

- *Web service* para fazer a inserção de novos *tickets*;
- *Web service* para fazer a alteração posterior dos dados de *tickets* inseridos (particularmente o status, de forma que outros sistemas externos possam mudar o status da solicitação conforme os processos de negócio da empresa);
- O sistema deve ainda implementar uma solução para que o técnico possa alterar o status do *ticket* utilizando para tal um dispositivo móvel.
- O aplicativo deve permitir a obtenção de rotas entre os destinos de atendimento;
- Por fim, o aplicativo deve permitir a consulta do histórico de atendimentos prestados pelo técnico de campo;

2.1.2 **Requisitos não Funcionais**

Utilizando os dados da entrevista efetuada em 2011, foram definidos para o Aplicativo *Ulixes Mobile* os seguintes requisitos não funcionais:

- O aplicativo deve ser compatível com *smartphones* de baixo custo;
- O *Ulixes Mobile* deve ser compatível com as versões do *Android* superiores a versão 2.1;

2.2 ESTRUTURA DO SISTEMA

A aplicação *Ulixes Mobile* foi idealizada para funcionar da forma mais simples possível, utilizando *WebServices* para obter e enviar os tickets para o Sistema *Ulixes*. Uma classe foi criada com o intuito de concentrar essas funcionalidades de acesso a *WebServices*, e, desta forma, sendo a única parte do sistema que lida com os objetos SOAP.

O banco de dados embarcado *SQLite* também tem uma grande importância para o projeto *Ulixes Mobile*, visto que todos os dados relativos aos *tickets* são armazenadas

neste. Todo o processo de acesso a este banco de dados é feito por uma classe DAO (*Data Access Object*).

A interação do usuário com o sistema é feita através de classes de interface, tendo sua lógica processada em classes de background, sendo assim, podemos dizer que o sistema foi implementado utilizando uma interpretação flexível do padrão arquitetural MVC (*Model View Control*).

2.2.1 **Softwares de Aplicação**

Segundo Hoffmann e Moraes (2011):

> "Os softwares desenvolvem papel crucial na melhoria dos processos empresariais, reduzindo custos e melhorando a qualidade do serviço prestado por tais empresas. Esses *softwares*, voltados à resolução de problemas específicos de um determinado negócio, são chamados de *softwares* de aplicação".

Pressman (2006, p. 6) define um *sofware* de aplicação como sendo:

> "[...] programas isolados que resolvem uma necessidade especifica do negócio. Aplicações nessa área processam dados comerciais ou técnicos de um modo que facilita as operações ou gestão/tomada de decisões técnicas do negócio. Além das aplicações convencionais de processamento de dados, o software de aplicação é usado para controlar funções Fo negócio em tempo real (por exemplo, processamento de transações no ponto-de-venda, controle de processo de fabricação em tempo real)."

Também segundo Pressman (2006), "[...] todo software é iniciado por alguma necessidade do negócio [...]", partindo desde simples correções em aplicações existentes até a criação de novos sistemas, produtos ou serviços.

Grandes empresas costumam ter diversos sistemas customizados para responderem as suas necessidades de utilização, o que torna a manutenção desses extremamente cara, visto que as correções têm que ser executadas por equipes da própria empresa ou por outras empresas terceiras contratadas especificamente para isso, ao contrário dos *softwares* de prateleira, que são mantidos por suas empresas criadoras. O *Ulixes*, apesar de não ser um *software* de prateleira, se propõe a ter um custo de

manutenção extremamente baixo, se deixar de possibilitar uma boa capacidade de customização as suas empresas usuárias.

2.2.2 **Tecnologias Utilizadas**

Para a execução do projeto *Ulixes Mobile* foi necessária a utilização de várias tecnologias, entre elas se pode destacar a utilização do sistema operacional *Android* como plataforma para a execução do sistema, a utilização do framework *KSoap* para o acesso a *WebService* e o banco de dados embarcado *SQLite* para a persistência dos dados. Cada uma dessas três tecnologias será explicada em maiores detalhes nos subtópicos que se seguem.

2.2.2.1 Android

O *Android* foi escolhido como plataforma de desenvolvimento do *Ulixes Mobile* por ser a plataforma mais popular para sistemas móveis. Além disso, sua vasta gama de *smartphones* possibilita aos usuários várias opções de baixo custo, que em nosso caso, seriam ideais para o uso pelos técnicos de campo.

Como é descrito por Mario Zechner (2011, p. 02), o

> "*Android* foi notado pela primeira vez em 2005, quando o *Google* adquiriu uma pequena empresa chamada *Android Inc*. Isso alimentou especulações de que o *Google* queria entrar no espaço móvel. Em 2008, o lançamento da versão 1.0 do *Android* pôs fim a todas as especulações e o *Android* se tornou o novo desafiante no mercado móvel. Desde então, vem batalhando com as plataformas já estabelecidas, como *iOS* (então chamado de *iPhone OS*) e *BlackBerry*, e suas chances de ganhar parecem muito boas."

O mesmo Mario Zechner (2011, p. 02) descreve que

> "... devido ao Android ser de código aberto, os fabricantes de celulares têm uma baixa barreira de entrada quando se utiliza a nova plataforma. Eles podem produzir dispositivos para todos os segmentos de preços, modificando o próprio *Android* para acomodar o poder de processamento de um dispositivo específico. Desta forma, o *Android* não se limita aos dispositivos

> de alto custo, podendo ser implantado também em dispositivos de baixo custo, atingindo assim um público mais amplo".

A formação "... da *Open Handset Alliance* (OHA) em 2007 foi crucial para o sucesso do *Android"* (ZECHNER, 2011). A OHA inclui diversas companhias, entre as quais *HTC*, *Qualcomm*, *Motorola* e *NVIDIA*, que colaboram com o desenvolvimento de padrões para dispositivos móveis.

O *Android* é baseado em um *kernel Linux* versão 2.6 e as suas aplicações são escritas em *Java*, e são executadas em uma máquina virtual de nome *Dalvik VM*. O *Android "...* é um sistema mutiprocessos que suporta vários aplicativos simultâneos, aceita múltiplas formas de entrada, é altamente interativo, e é flexível o suficiente para suportar uma ampla gama de dispositivos, agora e no futuro. A interface ao usuário é ao mesmo tempo rica e fácil de usar" (MEDNIEKS et Al.).

Segundo Zechner (2011, p. 07) as principais funcionalidades do *Android* são:

- Um *framework* de aplicação que proporciona um rico conjunto de *APIs* (*Application Programming Interface*) para criar vários tipos de aplicações. Também permite a reutilização e substituição de componentes fornecidos pela plataforma e por aplicativos terceiros;
- A máquina virtual *Dalvik*, que é responsável pela execução das aplicações *Android*;
- Um conjunto de bibliotecas gráficas para programação 2D e 3D;
- Suporte a mídias comuns de áudio e vídeo;
- API para acessar periféricos, como a câmera, GPS (*Global Positioning System*), bússola, acelerômetro, tela sensível ao toque, *trackball* e teclado. Note que nem todos os dispositivos *Android* têm estes periféricos, ou seja, a já citada fragmentação de *hardware*;

A arquitetura do *Android* é composta pó uma pilha de componentes, e cada componente é construído sobre os componentes da camada abaixo. A figura 1 nos dá uma visão global dos principais componentes do *Android*.

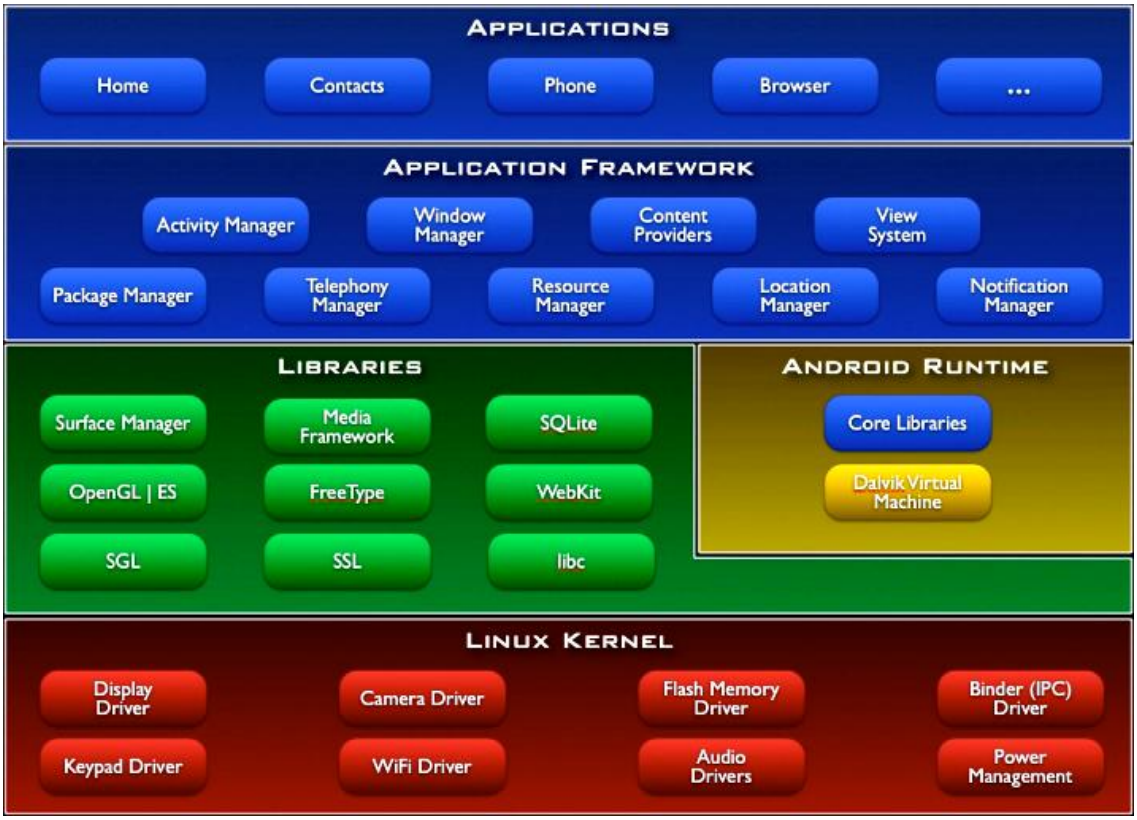

Figura 1 – Visão global arquitetura *Android*.

Fonte: Zechner (2011, p. 08).

2.2.2.2 kSOAP

"*kSOAP* é uma biblioteca para clientes web service para ambientes *Java* com recursos reduzidos como *Applets* ou aplicações J2ME" (*KSOAP 2*, 2013). No caso do *Ulixes Mobile*, a versão dois desta biblioteca foi utilizada em uma aplicação *Android*.

O *kSOAP 2* é "...uma reformulação completado *kSOAP*, levando em conta as lições apreendidas com a versão 1 do *kSOAP*..." (*KSOAP 2*, 2013).

O kSoap "...faz a conversão de dados de mensagens SOAP em objetos Java, de forma que o desenvolvedor trabalhe com essa tecnologia de forma transparente" (YUAN, 2004, p. 310 apud SANTOS, 2012).

No *Ulixes Mobile*, a utilização do *kSOAP 2* foi necessária para os casos de uso enviar tickets e receber tickets.

2.2.2.3 SQLite

Segundo o site oficial do projeto o "... *SQLite* é uma biblioteca de software que implementa um banco de dados *SQL (Structured Query Language)* autossuficiente, sem servidor, sem configurações e transacional. O *SQLite* é o banco de dados mais amplamente utilizado no mundo..." sendo seu código fonte "...de domínio público." (SQLITE HOME PAGE, 2013).

*"*Em contraste com outros bancos de dados, o SQLite remove de forma agressiva recursos que não são necessários, diminuindo muito seu custo para o sistema.*"* (MEDNIEKS et Al.).

Algumas das funcionalidades descritas no site do *SQLite* (SQLITE HOME PAGE, 2013) são as seguintes:

- As transações são atômicas, consistentes, isoladas e duráveis (ACID, acrônimo em inglês para atomic, consistent, isolated and durable), mesmo depois de falhas no sistema ou no suprimento de energia;
- Nenhuma configuração ou administração é necessária;
- O banco de dados completo é armazenado em um único arquivo multiplataforma;
- Suporta bancos de dados de terabytes e strings e BLOBs (*Binary Large Object*) de *gigabytes*;
- Mais rápido do que bancos de dados cliente/servidor para a maioria das operações comuns;
- *API* simples e fácil de usar;
- Escrito em *ANSI-C* (*American National Standards Institute* C);
- Código bem comentado com 100% do mesmo alcançado por testes;
- Disponível como um único arquivo *ANSI-C* que pode ser facilmente colocado em outro projeto;
- Autossuficiente: sem dependências externas;
- Multi-plataforma: *Unix* (*Linux, Mac OS-X, Android e iOS*) e Windows (*Win32, WinCE e WinRT*) são suportados. Fácil de portar para outros sistemas;
- Os códigos fontes são de domínio público.

- Vem com uma interface de comando independente (*CLI*) do cliente que pode ser usada para administrar o banco de dados *SQLite*;

Outras características que tornam o *SQLite* ideal para a utilização em dispositivos móveis, segundo o site do projeto, são o seu pequeno tamanho em formato binário (compilado), o uso eficiente de memória e a sua alta confiabilidade.

2.3 DIAGRAMAS DE CASOS DE USO

Diagramas de Casos de Uso são utilizados para descrever as funcionalidades de um sistema. Para um melhor entendimento das funcionalidades do aplicativo *Ulixes Mobile*, temos a seguir a representação em forma de diagrama das funcionalidades do sistema.

2.3.1 **Diagramas de Casos de Uso *Ulixes Mobile***

O diagrama da Figura 2 representa os casos de uso do sistema *Ulixes Mobile* que estarão disponíveis para os técnicos de campo.

Na imagem podem-se verificar os casos de uso que compõe o sistema *Ulixes Mobile*, sendo eles Verificar Tickets, Obter Rota, Obter Tickets, Enviar Tickets, Modificar Configurações e Obter Histórico.

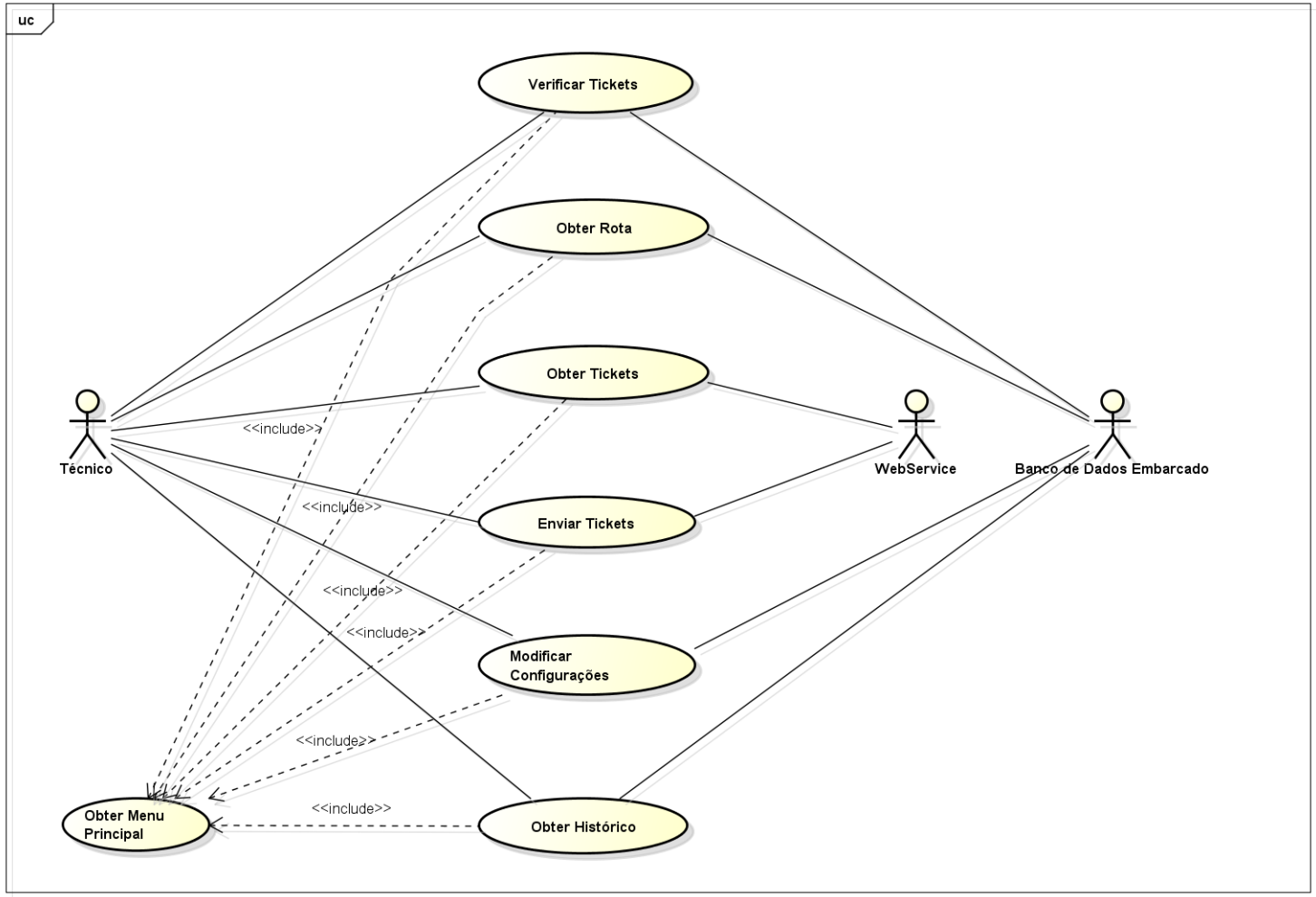

Figura 2 – Diagrama de Casos de Uso *Ulixes Mobile*.

Fonte: Autoria própria.

2.4 DIAGRAMAS DE CLASSES

Diagramas de Classes são representações de estrutura e dos relacionamentos entre as classes que servem de modelos para objetos. Esta modelagem serve de base para a construção de diagramas de sequência, de comunicação e de estados, pois define todas as classes que o sistema precisa possuir para o seu bom funcionamento. Nessa representação, ficam conhecidos todos os atributos e métodos existentes nas classes e todos os relacionamentos e associações existentes entre tais classes. A seguir têm-se os diagramas de classes do sistema *Ulixes Mobile*, sendo que as classes constantes nesses diagramas foram obtidas através de levantamento realizado nos diagramas de casos de uso. Além disso, já são especificados os métodos e atributos que integrarão cada classe do projeto.

2.4.1 **Diagrama de Classes *Ulixes Mobile***

O diagrama de classes do *Ulixes Mobile* tem seu núcleo na classe *UlixesApp* (classe global que estende de *Application* é serve para manter um estado de aplicação global), esta classe ativa a *SplashScreen* que por sua vez ativa a classe *MainActivity*, que é a principal classe de interface com o usuário do sistema.

No diagrama também podemos verificar as outras classes de interface com o usuário, sendo elas a *TelaConfig, TelaTicket, TicketActivity* e *UlixesMapActivity*, além é claro da já listada *MainActivity*.

Uma das características importantes do diagrama de classes e que as relações entre estas podem ser inferidas do diagrama, mesmo com um diagrama simples com este, como se pode observar na figura 3.

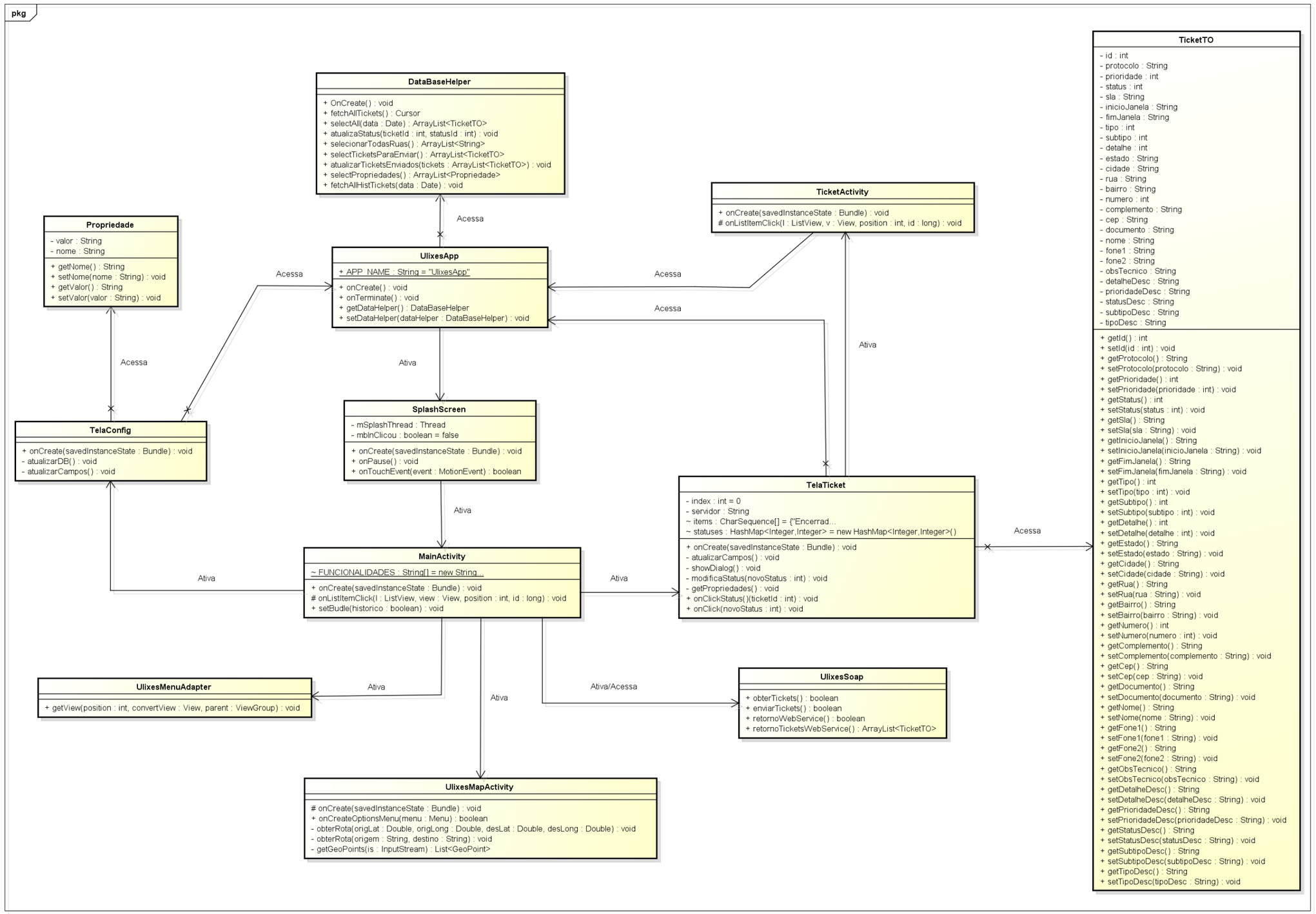

Figura 3 – Diagrama de Classes *Ulixes Mobile*.

Fonte: Autoria própria.

2.5 DIAGRAMAS DE SEQÜÊNCIA

Diagramas de sequência têm por finalidade apresentar de forma mais detalhada as funcionalidades do sistema, detalhando a sequencia das mensagens trocadas entre os objetos que compõe o sistema.

Para o sistema *Ulixes Mobile* definimos seis casos de uso principais e um caso de uso auxiliar que é incluído em todos outros casos de uso, sendo ele o caso de uso Obter Menu Principal.

A seguir temos a representação gráfica na forma de casos de uso de todos os sete casos de uso definidos para o sistema.

2.5.1 **Diagrama de Sequência Obter Menu Principal**

A figura 4 apresenta o caso de uso Obter Menu Principal, que é o ponto de entrada para a aplicação *Ulixes Mobile*.

Ao abrir a aplicação, a classe *UlixesApp* é a primeira a ser instanciada, esse objeto é responsável por instanciar a classe *DataBaseHelper*, que tem como função servir de interface para o banco de dados embarcado *SQLite*. Neste momento de inicialização o banco de dados é verificado, e, sendo o primeiro acesso, este é criado por esta classe auxiliar.

Uma vez que esta verificação é completada, a classe *MainActivity* é instanciada e a tela principal da aplicação é mostrada ao usuário. Nesta tela, o usuário pode selecionar qual funcionalidade do sistema deseja selecionar, sendo as funcionalidades disponíveis: Verificar Tickets, Obter Rota, Obter Tickets, Enviar Tickets, Modificar Configuração e Obter Histórico.

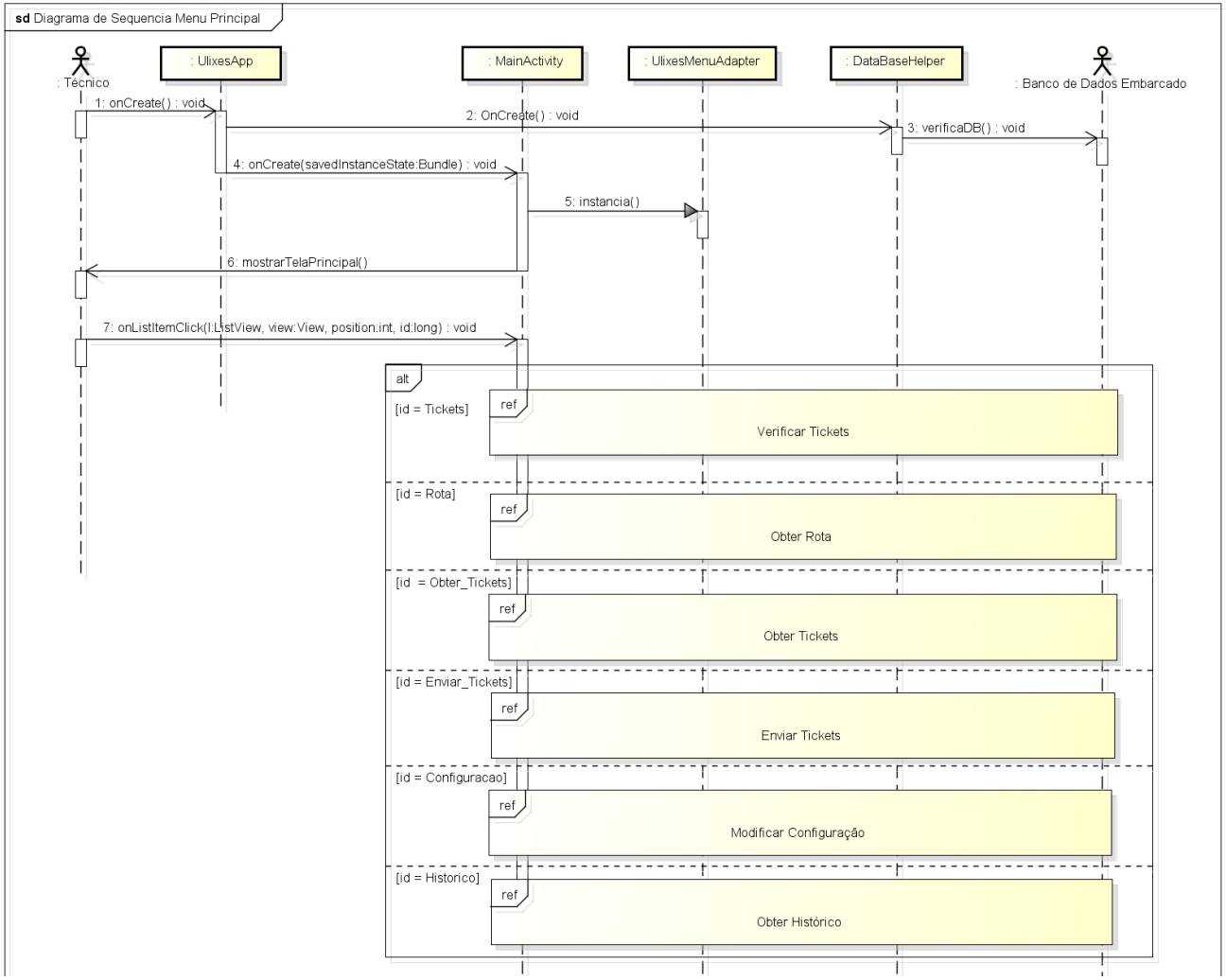

Figura 4 – Diagrama de Sequência Obter Menu Principal.

Fonte: Autoria própria.

2.5.2 **Diagrama de Sequência Verificar Tickets**

O diagrama de sequência Verificar Tickets representa a principal funcionalidade do sistema *Ulixes Mobile*. Essa funcionalidade tem por intuito mostrar ao técnico de campo todos os tickets que estão assinalados para a sua jornada de trabalho diária.

Quando a tela *TicketActivity* é instanciada, uma consulta é realizada no banco de dados embarcado utilizando a classe *DataBaseHelper*, a qual retorna os tickets que tema sua data de atendimento igual a data atual. Esses tickets são apresentados de forma resumida ao usuário, em uma lista (*ListView*).

Como se pode ver na figura 5, uma vez que um ticket é selecionado nesta *ListView*, uma tela com os detalhes do ticket é apresentada ao usuário. Nesta tela, o técnico de campo pode alterar o status do ticket ao clicar no campo status. Ao fazer tal ação, um novo diálogo é apresentado ao usuário, com as seguintes opções: Encerrado, Suspenso Ausência de Cliente, Suspenso Falta de Facilidade e Suspenso Pedido de Reagendamento. Um simples clique sobre uma dessas opções é o suficiente para atualizar o status do ticket.

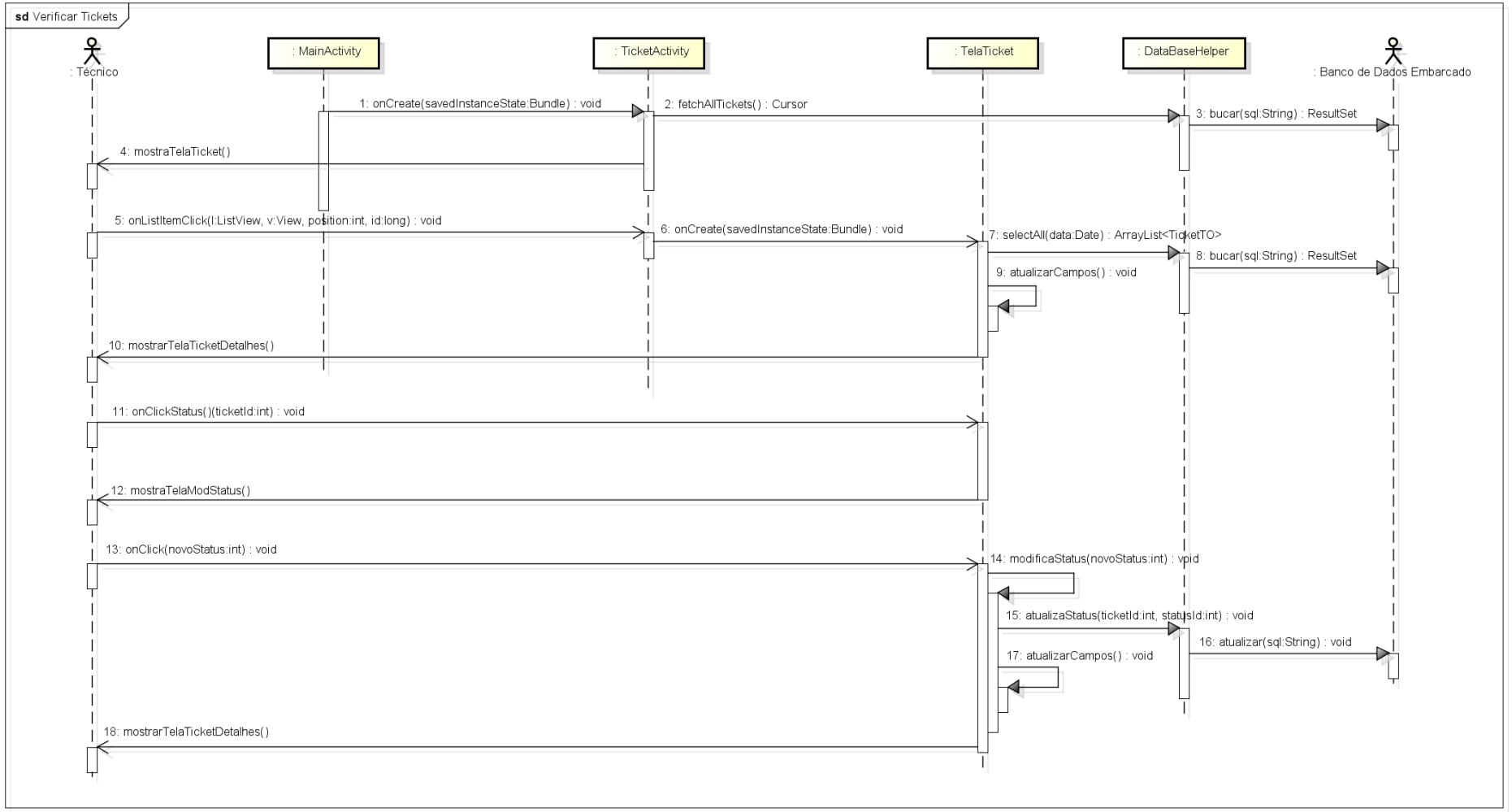

Figura 5 – Diagrama de Sequência Verificar Tickets.

Fonte: Autoria própria.

2.5.3 **Diagrama de Sequência Obter Rota**

Na figura 6 temos a representação na forma de diagrama de sequencia do caso de uso Obter Rota. Essa funcionalidade tem por intuito mostrar ao técnico de campo, utilizando as funcionalidades do *API* de mapas do Google, um mapa com a rota da jornada de trabalho diária do usuário.

O modo de funcionamento desta funcionalidade é bastante similar as outras funcionalidades apresentadas até agora: após a funcionalidade ser selecionada, a classe de interface com o usuário é criada, que por sua vez realiza uma consulta no banco de dados integrado para obter as informações relativas as ruas dos tickets que serão atendidos pelo técnico naquele dia. De posse destes dados, uma consulta é realizada nos serviços *web* do Google Utilizando a *API* 2 do *Google Maps*, que retorna as coordenadas geográficas das ruas que serão traçadas no mapa. Com estas coordenadas, as linhas podem ser traçadas no mapa.

A tela apresentada tem uma interface bem simples, valendo-se das funcionalidades do *Google Maps* para enriquecer a experiência do técnico de campo ao utilizar este caso de uso.

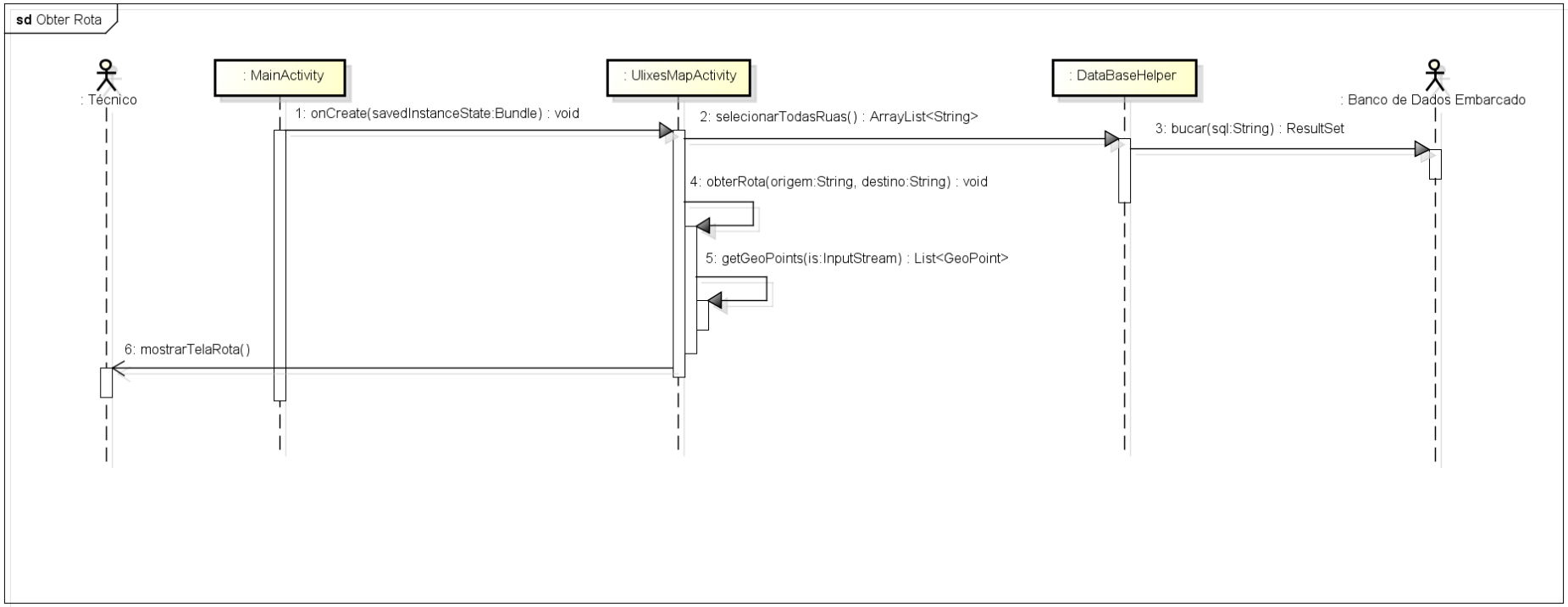

Figura 6 – Diagrama de Sequência Obter Rota.

Fonte: Autoria própria.

2.5.4 **Diagrama de Sequência Obter Tickets**

O caso de uso Obter Tickets foi concebido para que os técnicos de campo pudessem receber seus tickets diários através de uma consulta a um *WebService*. A representação deste como diagrama de sequencia é bem simples, visto que poucas classes estão envolvidas no processo e poucas mensagens são trocadas entre elas.

A funcionalidade é inicializada quando o usuário clica na funcionalidade Obter Tickets no menu principal. Logo após essa ação, o método *obterTickets* da Classe *UlixesSoap* é chamado. Esse método, por sua vez, chama o WebService responsável pelo envio dos tickets, e, se a chamada ocorrer com sucesso, os tickets retornados são salvos no banco de dados embarcado.

Ao final da execução, uma mensagem é retornada para o usuário para indicar se a execução for bem sucedida ou não, como se pode verificar na figura 7.

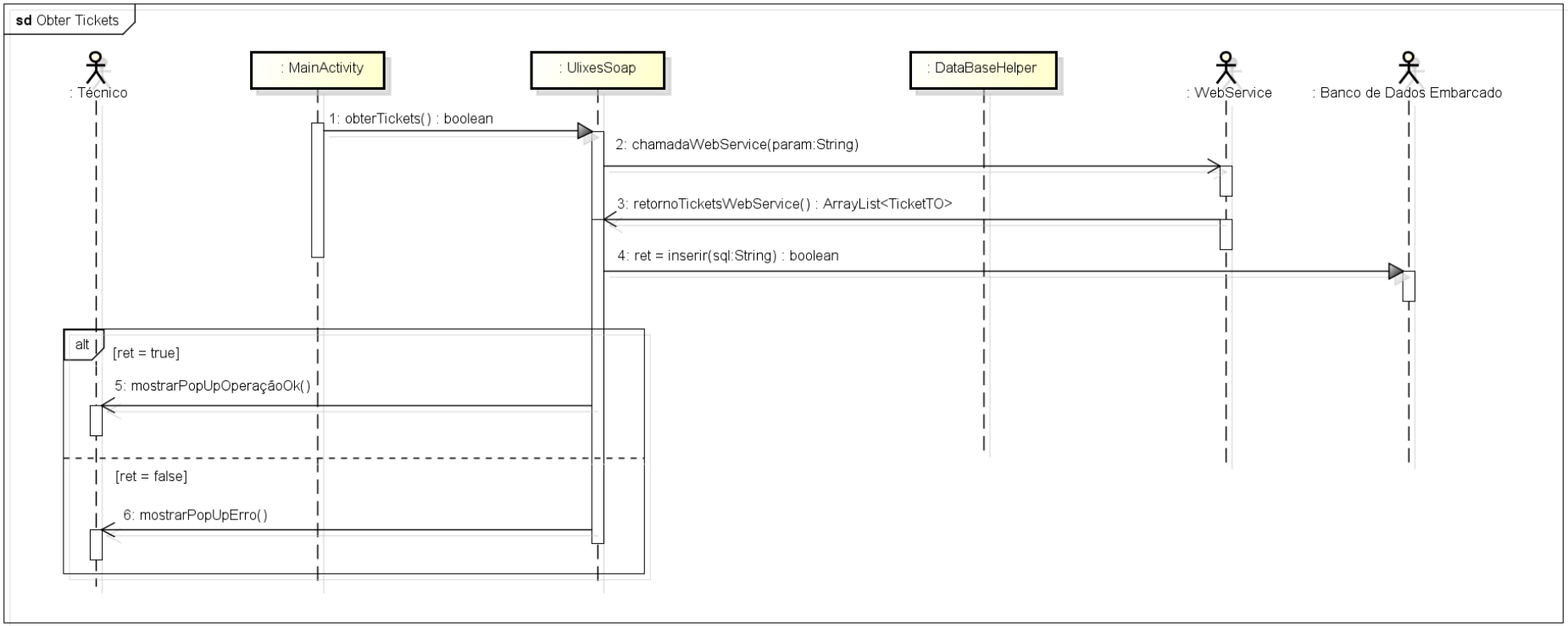

Figura 7 – Diagrama de Sequência Obter Tickets.

Fonte: Autoria própria.

2.5.5 **Diagrama de Sequência Enviar Tickets**

O diagrama de sequencia Enviar Tickets é muito parecido com o diagrama de sequencia Obter Tickets, visto que ambas as funcionalidades fazem uso das mesmas instancias de classes.

Na figura 8 podemos observar que ao ser selecionada tal função, o método *enviarTickets* da classe *UlixesSoap* é chamado, o qual por sua vez chama o método *selectTicketParaEnviar* da classe auxiliar *DataBaseHelper*. O objeto desta classe, por sua vez, faz a consulta ao banco de dados embarcado para obter todos os tickets com status passível de ser enviado para o banco de dados (Encerrado, Suspenso Ausência de Cliente, Suspenso Falta de Facilidade e Suspenso Pedido de Reagendamento).

De posse dos tickets retornados, o objeto instanciado da classe *UlixesSoap* chama o *WebService* no servidor *Ulixes* responsável pelo recebimento do tickets enviado pelo cliente. Ao final do processo, uma mensagem é apresentada ao técnico de campo para indicar se a execução da função ocorreu com sucesso.

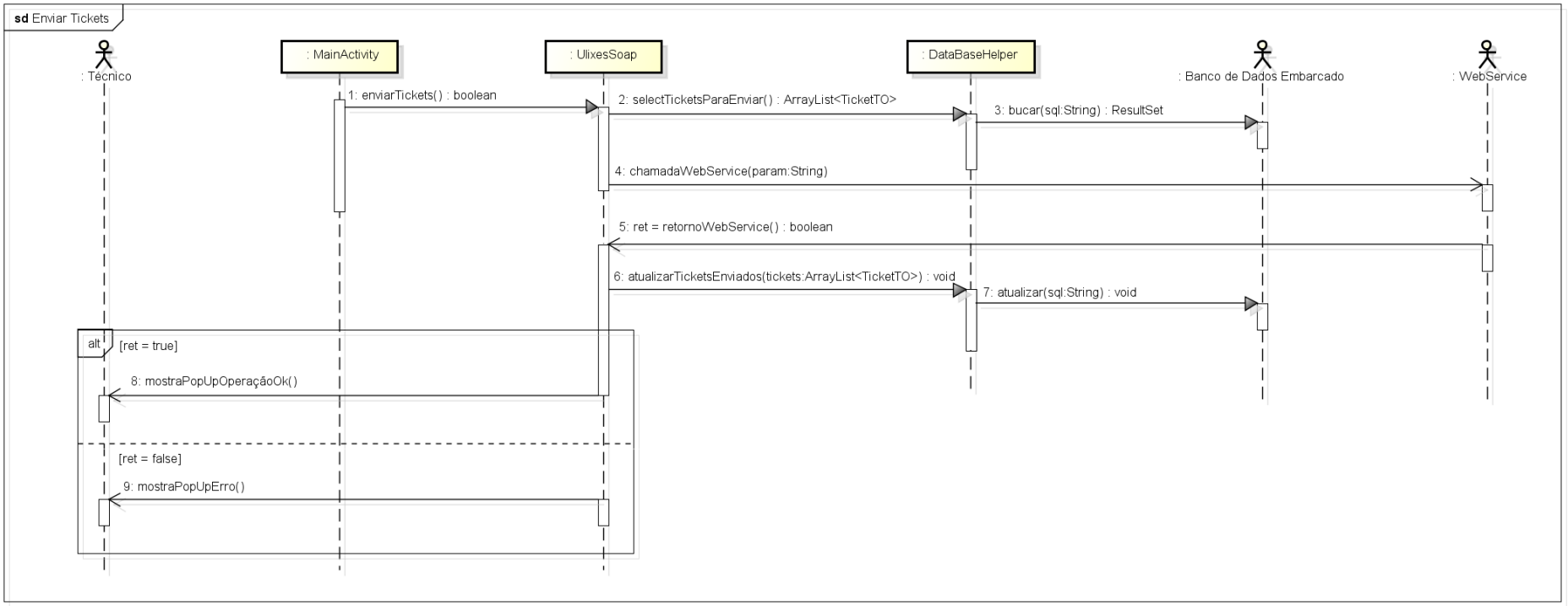

Figura 8 – Diagrama de Sequência Enviar Tickets.

Fonte: Autoria própria.

2.5.6 **Diagrama de Sequência Modificar Configurações**

Na figura 9 temos o diagrama de sequencia do caso de uso Modificar Configurações. Esta funcionalidade tem por intuito permitir que o técnico de campo possa modificar as configurações da *URL* (*Uniform Resource Locator*) para acesso ao *WebService* de obtenção de tickets e a *URL* para acesso ao WebService utilizado para o envio dos tickets ao servidor. Além disso, nesta tela o usuário pode modificar o seu identificador de acesso ao servidor e a sua senha.

No primeiro momento, quando a funcionalidade é selecionada, a *TelaConfig* é instanciada, é, fazendo uso do objeto da classe *DataBaseHelper*, obtém as configurações já previamente salvas no banco de dados. De posse desses dados, a tela de configurações é apresentada ao usuário.

Após a tela ser apresentada, o usuário pode modificar os dados e salvar as novas configurações utilizando o botão atualizar, ou pode cancelar as modificações utilizando o botão homônimo à ação e receber a tela com os dados originais.

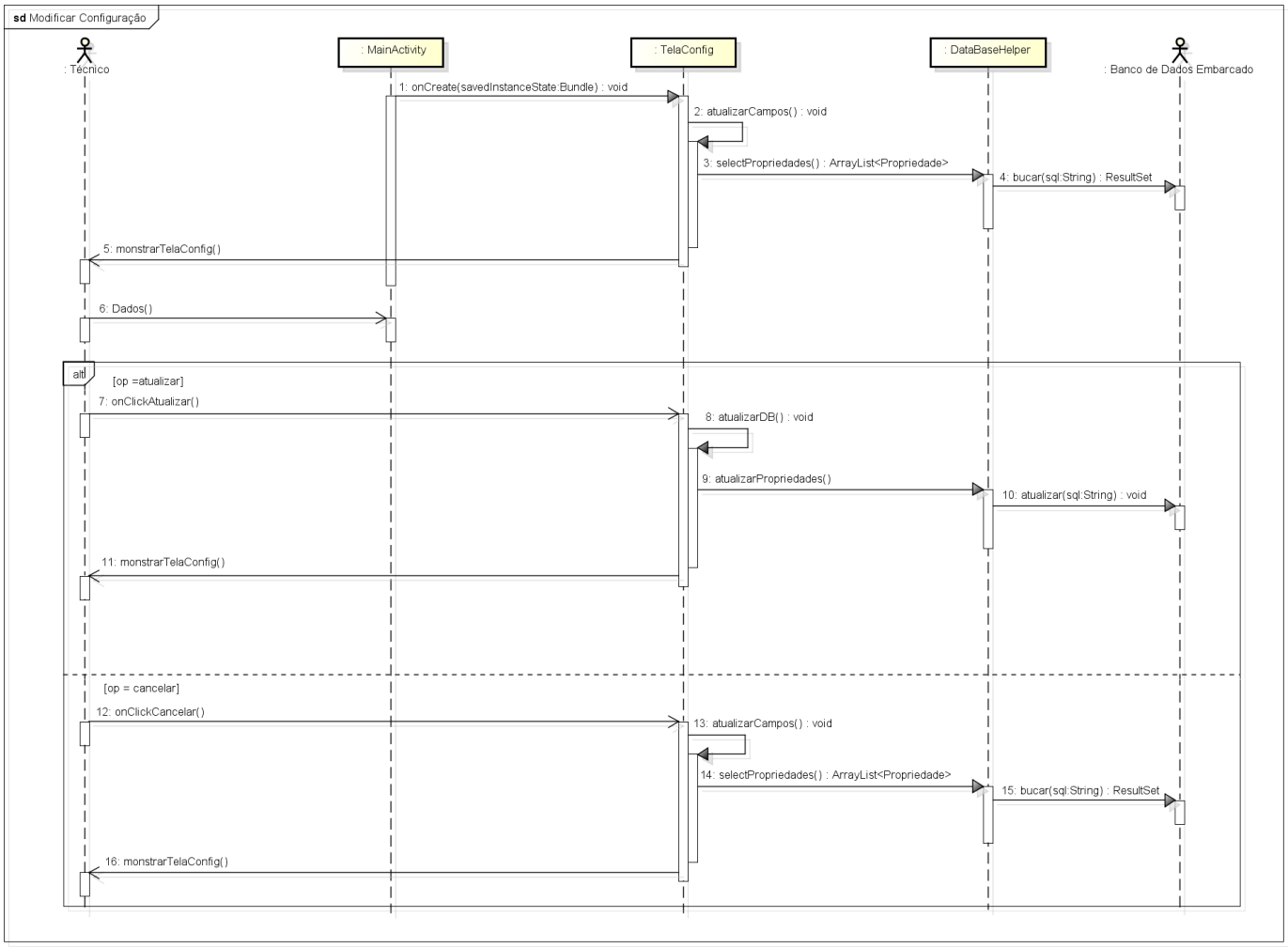

Figura 9 – Diagrama de Sequência Modificar Configurações.

Fonte: Autoria própria.

2.5.7 **Diagrama de Sequência Obter Histórico**

O caso de uso Obter Histórico funciona de forma similar ao caso de Verificar Tickets, inclusive compartilhando as mesmas classes que são utilizadas por esse caso de uso, como se pode verificar na figura 10.

Num primeiro momento, quando a funcionalidade é selecionada no menu principal, o método *setBudle* é chamado no objeto instanciado da classe *MainActivity*, sendo este necessário para se criar um parâmetro de inicialização que indicará ao método *onCreate* da classe *TicketActivity* que esta será aberta em modo Histórico.

Após esse processo, a classe *TicketActivity* é instanciada, a qual chama o método *fetchAllHistTickets* da classe *DataBaseHelper*, que retorna as principais informações de todos os tickets fechados/cancelados no dia anterior. Logo após, a tela ticket é apresentada ao usuário.

Ao se clicar em um ticket na interface, a classe *TelaTicket* é instanciada, os detalhes deste ticket são obtidos através da classe *DataBaseHelper* é apresentados ao usuário utilizando a tela "*TicketDetalhes*".

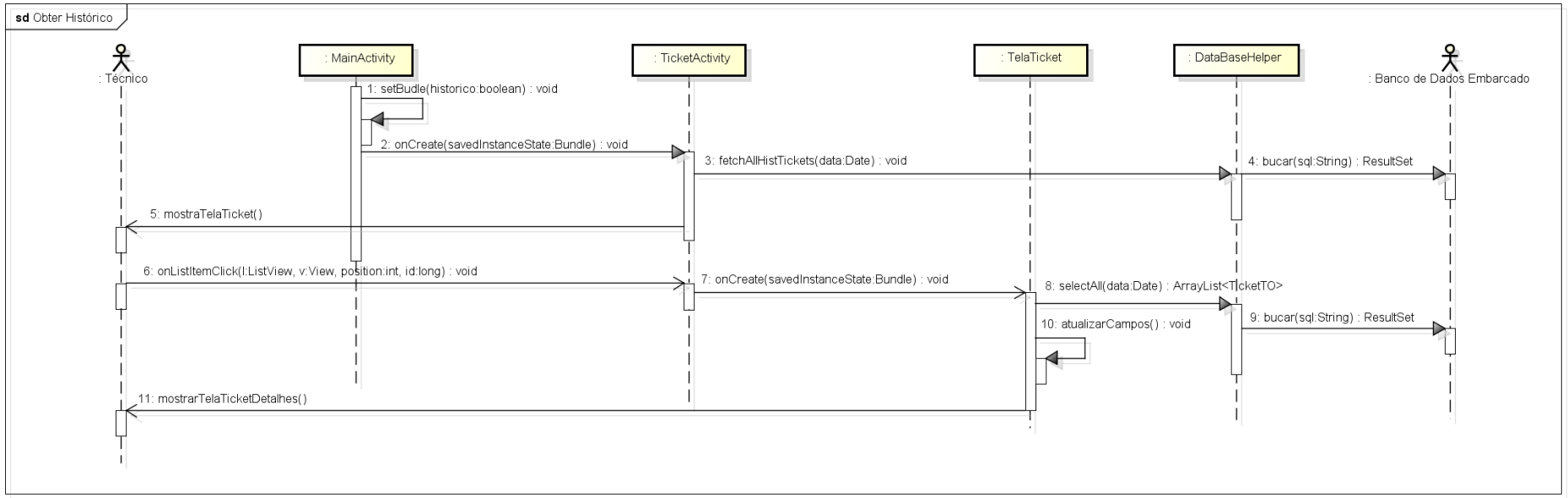

Figura 10 – Diagrama de Sequência Obter Histórico.

Fonte: Autoria própria.

2.6 SOLUÇÕES DESENVOLVIDAS

Após apresentar as especificações iniciais que levaram ao desenvolvimento do sistema *Ulixes Mobile*, e, tendo apresentado o projeto de software que se originou destas especificações, pode-se apresentar agora as soluções que foram criadas com base nestes projetos.

2.6.1 ***Splash Screen***

A *Splash Screen*, apresentada ao início da execução do programa, faz uso das funcionalidades de Threads para apresentar um diálogo de introdução ao sistema, como se pode ver a seguir na figura 11.

Figura 11 – *Ulixes Mobile* Splash Screen.

Fonte: Autoria própria.

2.6.2 **Menu Inicial**

O sistema *Ulixes Mobile* foi idealizado com o intuito de facilitar as operações diárias dos técnicos de campo de empresas de Telecom, contudo, o referido aplicativo

poderia se adequar as operações das mais diversas empresas que necessitam prestar serviço de atendimento a clientes.

Pensando nesse propósito, de facilitar as operações diárias dos técnicos de campo, o menu inicial foi idealizado para ser o mais simples e usual possível, e assim tendo uma ótima curva de aprendizado.

Utilizando uma *ListActivity*, as funcionalidades são apresentadas de forma clara e intuitiva aos usuários, numa disposição linear, que facilita ao técnico de campo a seleção da funcionalidade desejada (figura 12).

Figura 12 – *Ulixes Mobile* Menu Inicial.

Fonte: Autoria própria.

2.6.3 **Verificar Tickets**

A funcionalidade verificar tickets, assim como o menu inicial, estende da classe *ListActivity* para apresentar, de forma resumida, os tickets a serem atendidos pelo técnico de campo em sua jornada diária de trabalho (figura 12).

Nesta tela, são apresentados ao usuário o número do protocolo e o início da janela de atendimento dos tickets. Ao se clicar em qualquer um desses tickets, uma tela contendo todos os detalhes do chamado é apresentada ao usuário.

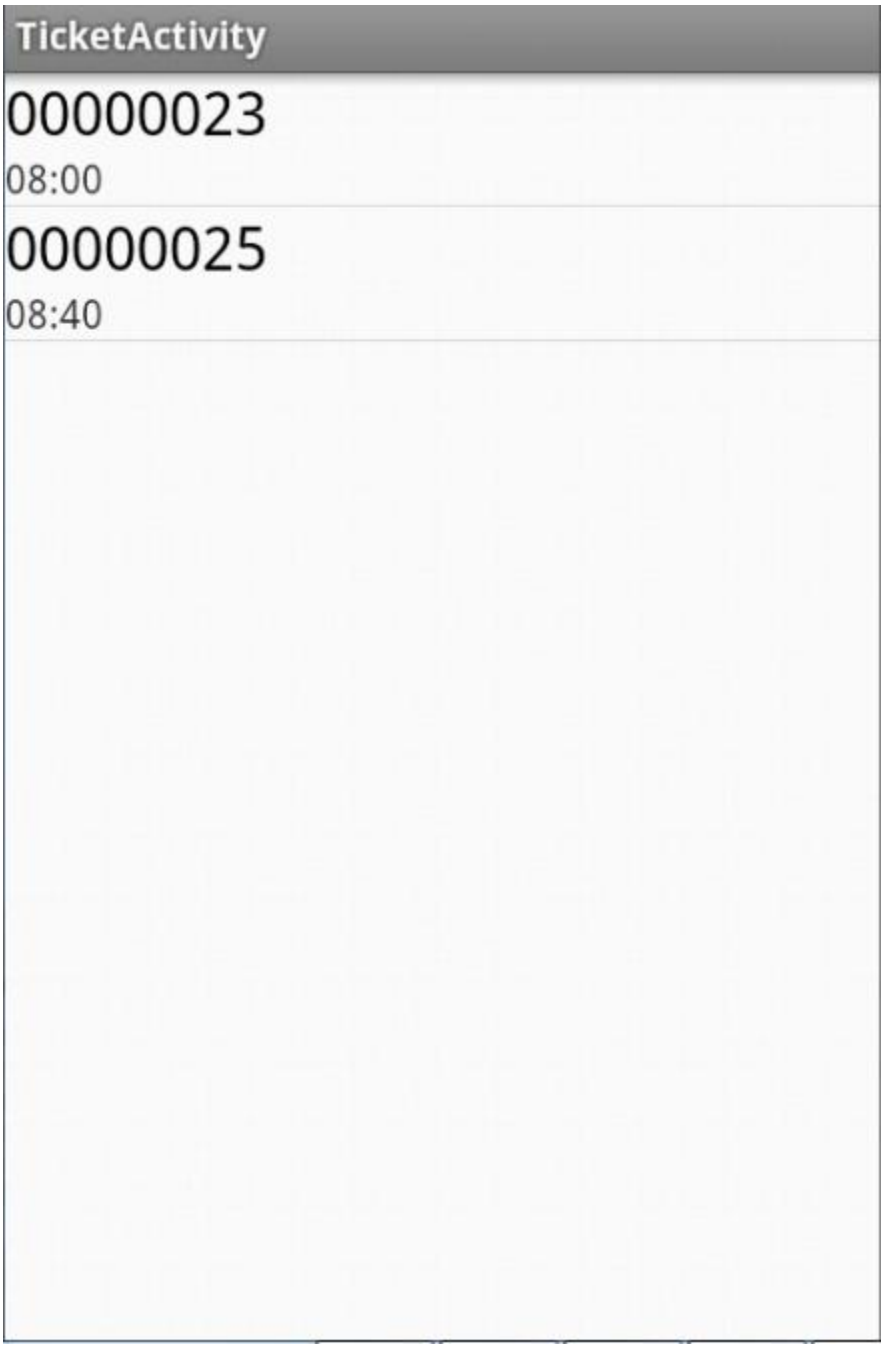

Figura 13 – *Ulixes Mobile* Funcionalidade Verificar Tickets.

Fonte: Autoria própria.

2.6.4 **Obter Rota**

Uma das principais funcionalidades adicionadas ao sistema *Ulixes Mobile* foi a possibilidade de se traçar a rota entre os diversos locais de atendimento nos quais o técnico de campo terá que prestar serviço.

Para que esta tela pudesse ser feita, foi necessário um estudo aprofundado sobre as funcionalidades disponíveis na *API* do *Google Maps* para *Android*.

Uma das grandes dificuldades enfrentadas durante o desenvolvimento do aplicativo foi justamente a adaptação à nova *API* do *Google Maps*, visto que todas as experiências anteriores de desenvolvimento utilizando a referida *API* tinham sido feitas utilizando a primeira versão do *Google Maps*.

O primeiro passo necessário para que se pudesse fazer uso da *Google Maps Android API v2* é liberar a utilização do serviço na conta do *Google* que será utilizada pelo aplicativo, através da url https://code.google.com/apis/console/ (figura 14).

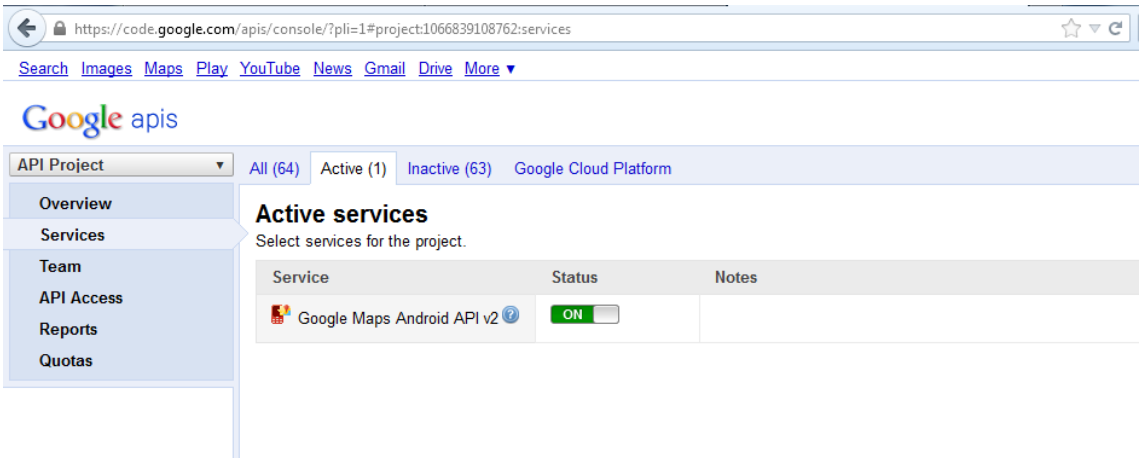

Figura 14 – *Google APIs Console*.

Fonte: Autoria própria.

Após esse procedimento, é necessário gerar uma chave *SHA-1 (Security Hash Alghorithm 1*) para o aplicativo. Usando o aplicativo *keytool*, integrante do *Java Development Toolkit (JDK)*, se obtém a chave *SHA-1* que foi criada durante a instalação do *Android Developer Tools (ADT)*, esta chave é necessária para gerar a chave *SHA-1* da aplicação. O seguinte comando é utilizado para obter a chave do *ADT*:

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

Usando a mesma página do *Google API* console, se pode gerar a chave *SHA-1* que será utilizada para fazer a assinatura da aplicação *Android*. Clicando-se na opção *API Access*, e depois em Create *New Android Key*, o diálogo a seguir é apresentado (figura 15), no qual ao se entrar a chave *SHA-1* do *ADT* e o nome do pacote da aplicação separados por um ponto e virgula, se pode gerar uma nova chave *SHA-1* para assinar a aplicação.

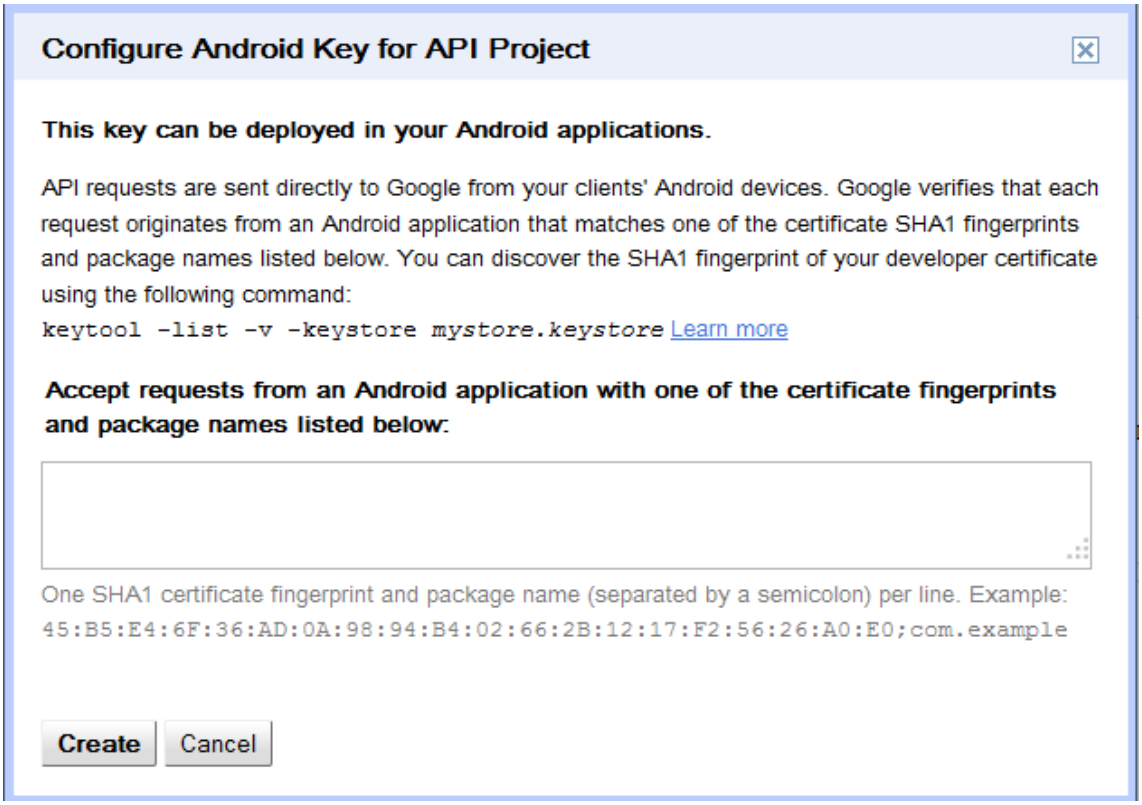

Figura 15 – Gerando uma chave *SHA1* como *o Google API* console.

Fonte: Autoria própria.

Após esse processo, é necessário se criar uma nova entrada no *AndroidManifest* da aplicação, como a descrita a seguir:

```
<meta-data
  android:name="com.google.android.maps.v2.API_KEY"
  android:value="Chave SHA-1 gerada" />
```

Feito isso, o *Google Maps* pode ser usado na aplicação.

Como as outras funcionalidades do *Ulixes Mobile*, o acesso a esse caso de uso é efetuado através da tela principal, ao se clicar na opção Rota. Logo após tal procedimento, uma tela contendo um mapa com rotas traçadas entre todos os pontos de atendimento é apresentada ao usuário, contendo todas as funcionalidades básicas do

Google Maps (*Zoom*, mudança de posição em foco, etc). Essa tela pode ser verificada na figura 16 apresentada a seguir.

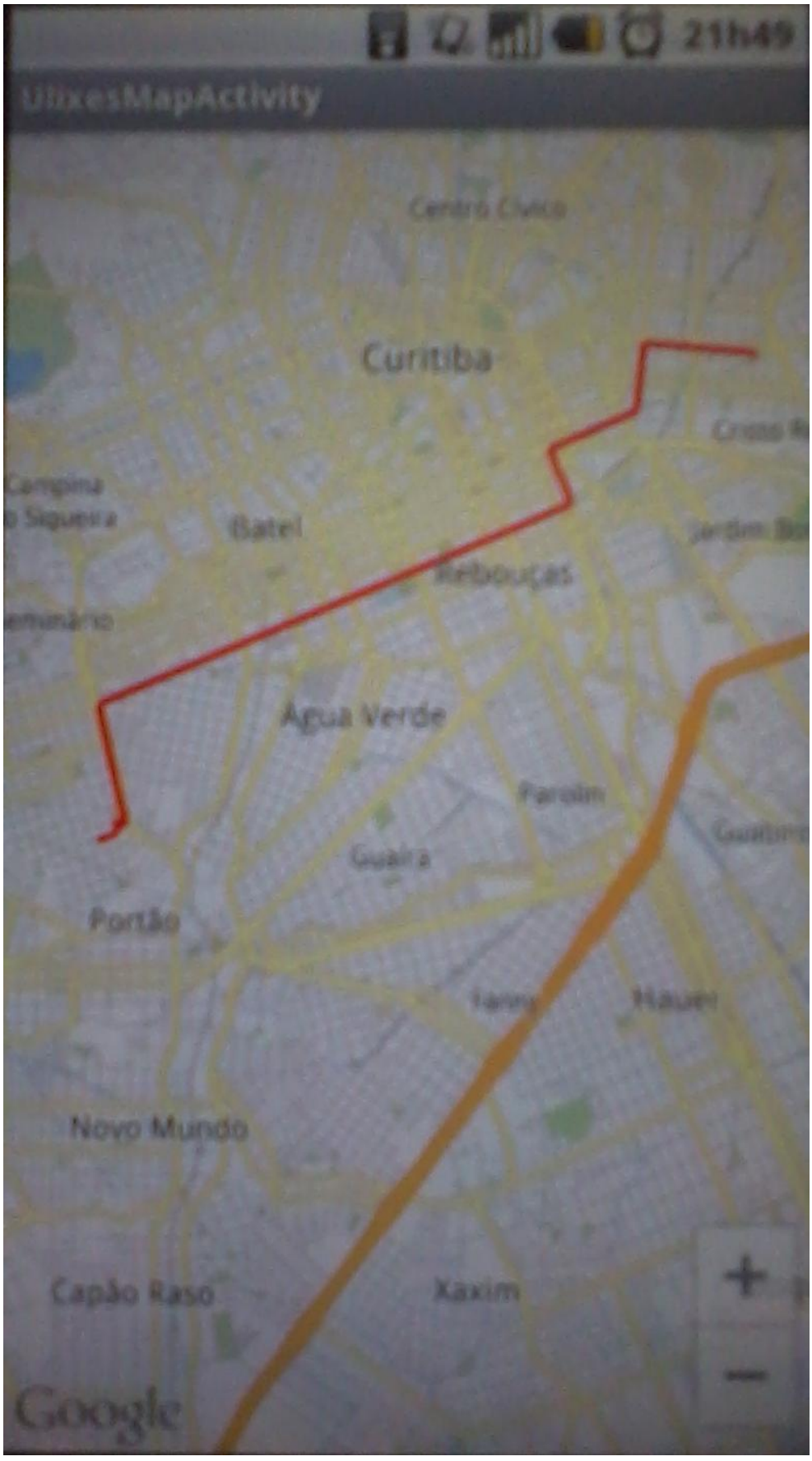

Figura 16 – *Ulixes Mobile* Funcionalidade Obter Rota.

Fonte: Autoria própria.

2.6.5 **Obter Tickets**

A funcionalidade Obter Tickets utiliza-se da biblioteca *KSoap* para conseguir se comunicar com o *WebService* que é utilizado pelo sistema *Ulixes* para enviar os tickets para o aplicativo móvel do técnico de campo. As bibliotecas básicas do *Android* não disponibilizam nenhuma forma de acesso a *WebServices*, devido a isso, a biblioteca *KSoap* se fez tal necessária para que a funcionalidade Obter Tickets pudesse ser implementada.

A ideia principal que levou ao desenvolvimento deste caso de uso é que o técnico de campo possa, ao começo de sua jornada de trabalho, sincronizar as informações relativas aos atendimentos que serão prestados naquele dia, e, desta forma, possa sair para atender aos clientes da empresa de forma autônoma.

Esta funcionalidade não tem nenhuma tela específica, sendo todas as interações deste caso de uso com o usuário apresentadas através da tela principal, como se pode verificar na figura 17.

Figura 17 – *Ulixes Mobile* Funcionalidade Obter Tickets.

Fonte: Autoria própria.

2.6.6 **Enviar Tickets**

Da mesma forma que a funcionalidade Obter Tickets, o caso de uso Enviar Tickets faz uso da biblioteca *KSoap* para acessar o *WebService* que recebe os tickets enviados pelo técnico de campo.

Após fazer uma consulta no banco de dados embarcado, todos os tickets já atendidos pelo técnico de campo, ou seja, todos os tickets que tem seus status iguais a Encerrado, Suspenso Ausência de Cliente, Suspenso Falta de Facilidade e Suspenso Pedido de Reagendamento são selecionados e enviados ao servidor.

Outra semelhança com o caso de uso Obter Tickets é que a funcionalidade Enviar tickets não possui tela própria, interagindo com o usuário através da tela principal, como se pode ver na figura 18.

Figura 18 – *Ulixes Mobile* Funcionalidade Enviar Tickets.

Fonte: Autoria própria.

2.6.7 **Modificar Configurações**

A funcionalidade Modificar Configurações tem por intuito habilitar o usuário a alterar as configurações de conexão com o servidor do sistema *Ulixes*. Assim como as outras funcionalidades do sistema, essa funcionalidade é acessada pela tela principal, ao

se clicar na funcionalidade Configuração, uma tela contendo as opções *URL* Obter Tickets, *URL* Enviar Tickets, Técnico e Senha (figura 19).

Os dados que são apresentados nessa tela são persistidos no banco de dados embarcado, em uma tabela especifica de configurações.

Figura 19 – *Ulixes Mobile* Funcionalidade Modificar Configurações.

Fonte: Autoria própria.

2.6.8 **Obter Histórico**

O objetivo da funcionalidade Obter Histórico é propiciar ao técnico de campo uma forma de se poder verificar os tickets já atendidos previamente. Um fato importante a se mencionar sobre esta funcionalidade é que as telas por ela utilizadas são as mesmas do caso de uso Verificar Tickets, com a diferença de que são selecionados no banco de dados embarcado os tickets atendidos no dia anterior.

Outro ponto importante a se mencionar é que nesta funcionalidade não é permitido se alterar o status dos tickets, visto que são tickets que já foram atendidos no dia anterior. A funcionalidade pode ser vista em maiores detalhes na figura 20.

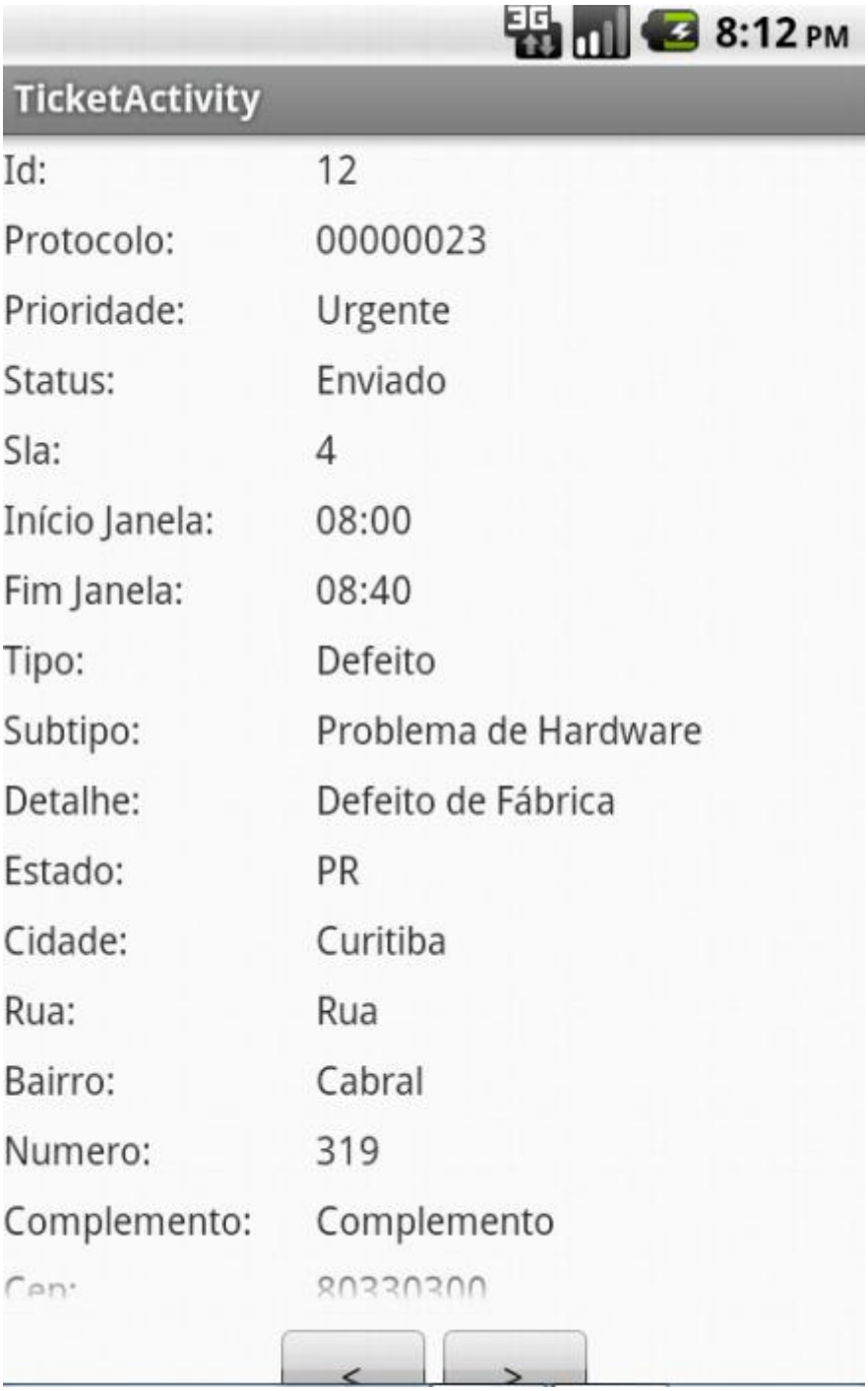

Figura 20 – *Ulixes Mobile* Funcionalidade Obter Histórico.

Fonte: Autoria própria.

3 CONCLUSÕES

O *Ulixes Mobile*, como parte integrante de um sistema de *Field Service Management* (FSM), o sistema *Ulixes*, foi idealizado para atender as necessidades de uma empresa com equipes de campo.

Dado à complexidade das atividades inerentes a prestação de serviços em campo, o *Ulixes Mobile* se propôs a ser uma ferramenta eficiente que pudesse agregar às diversas funcionalidades necessárias a execução das tarefas diárias dos técnicos de campo, facilitando a execução de suas tarefas diárias e reduzindo o tempo necessário para que essas fossem concluídas. O resultado final obtido com o projeto atende de forma completa a esses objetivos.

Ao definir-se o *Android* como plataforma para a execução do aplicativo *Ulixes Mobile*, escolheu-se pelo sistema operacional mais vendido no mundo e no Brasil para *smartphones* (KANTAR WORLDPANEL, 2013). Desta forma, escolheu-se por priorizar o desenvolvimento de um sistema em uma plataforma barata e que fosse acessível aos técnicos de campo ou as empresas para que pudessem fornecer tais dispositivos aos seus funcionários.

Além disso, ao se priorizar a utilização de *APIs* nativas tanto do sistema operacional Android quanto do Google para o desenvolvimento das funcionalidades do sistema *Ulixes Mobile*, houve uma grande economia de tempo e esforço para a conclusão do aplicativo.

Por fim, pode-se dizer que do ponto de vista acadêmico o sistema *Ulixes* serviu para se exercitar diversas tecnologias vistas durante o curso de pós-graduação em Java e Tecnologias Móveis, possibilitando o seu emprego em um sistema móvel completo.

3.1 TRABALHOS FUTUROS

Embora as funcionalidades principais tenham sido implementadas de uma forma satisfatória no sistema *Ulixes Mobile*, a parte gráfica do aplicativo ainda deixa um pouco a desejar. Desta forma, como trabalho futuro, poder-se-ia fazer um esforço para melhorar a interface do aplicativo.

Além disso, versões do aplicativo para outros sistemas operacionais também poderiam ser desenvolvidas, como por exemplo, versões para *iOS* e *Windows Phone*.

Por fim, funcionalidades de relatórios poderiam ser criadas para o aplicativo móvel, de forma a sintetizar as informações dos atendimentos realizados pelos técnicos de campo.

4 REFERÊNCIAS

4.1 BIBLIOGRAFIA

HOFFMANN, Daniel; MORAES, Tatiana. R. G. **Ulixes, um Aplicativo do Tipo Field Service Management System.** Monografia (Graduação em Tecnologia em Sistemas para Internet) – Departamento Acadêmico de Informática (DAINF), UTFPR, Curitiba, 2011.

MEDNIEKS, Zigurd et Al. **Programming Android**. 2ª ed. Sebastopol: O`Reilly Media, Inc, 2012.

PRESSMAN, Roger S.. **Engenharia de Software.** 6.ª ed. Nova Iorque: McGraw-Hill, 2006.

ZECHNER, Mario. **Beginning Android Games**. Nova York: Apress, 2011.

SANTOS, Jonasta. **Debitar: Gerenciamento de Débitos em Android com Sincronização Web/JSF Através de Web Service.** Monografia (Especialização em Java) – Departamento Acadêmico de Informática (DAINF), UTFPR, Curitiba, 2012.

4.2 MATERIAL *ONLINE*

SQLite Home Page. Disponível em: <http://www.sqlite.org/>. Acesso em: 07 de Julho de 2013.

kSOAP 2. Disponível em: <http://ksoap2.sourceforge.net/>. Acesso em 07 de Julho de 2013.

Kantar Worldpanel. Disponível em: <http://www.kantarworldpanel.com>. Acesso em 07 de Julho de 2013.

APÊNDICE A – QUESTIONÁRIO DE LEVANTAMENTO DE REQUISITOS

QUESTÕES TÉCNICAS

1) Deve existir um banco de dados que armazene informações, ou será utilizado um arquivo?

Por motivos de segurança e também de facilidade de manutenção, será empregado um sistema gerenciador de banco de dados.

2) Utilizando-se um sistema gerenciador de banco de dados, qual é o sistema de preferência do cliente?

O sistema gerenciador de banco de dados utilizado inicialmente será o Oracle, contudo, a implementação deverá ser flexível para aceitar outros tipos de sistemas gerenciadores de banco de dados no futuro.

3) O sistema será desktop ou web?

O sistema deverá ser web, além de disponibilizar *web services* para facilitar a integração com outros sistemas.

4) Existe uma preferência por linguagem de implementação? Em caso positivo, qual?

Não existe preferência por uma linguagem específica, desde que a escolhida atenda às necessidades definidas para este projeto. Entretanto, um dos requisitos para o projeto é que este tenha uma interface amigável ao usuário, de preferência com a utilização de elementos de interface rica para internet (*RIA – Rich Internet Applications*).

5) O software será executado em um sistema operacional específico, ou a meta é algo versátil?

Um dos principais requerimentos do software é que ele seja versátil e possa ser utilizado o mais independentemente possível da plataforma, no caso de usuário (utilização como cliente), contudo podem-se considerar restrições do lado do servidor.

6) O sistema deverá levar em conta restrições de desempenho de *hardware*?

O sistema deverá funcionar de forma satisfatória em máquinas legadas.

7) Os dados a serem armazenados no Banco de Dados deverão ser criptografados?

a) Sim.

b) Não.

c) Não sei responder.

Resposta: B.

8) O sistema deverá ter um *log* de erros?

Sim, pois tal procedimento torna a correção de erros um processo muito mais barato e eficiente.

QUESTÕES OPERACIONAIS

1) Qual o esquema de cores a ser usado nas interfaces com o usuário?

Não existe um esquema de cores pré-definido; o importante é que a interface seja atrativa e não seja cansativa. Cores muito vivas, se utilizadas, deveriam ser dosadas com cautela.

2) Qual é o nível de experiência de usuário recomendada para utilização deste software?

a) Usuários iniciantes.

b) Usuários intermediários.

c) Usuários avançados.

d) Não sei responder.

Resposta: A.

3) Qual o tempo de vida útil de tal projeto?

Em torno de dez anos.

QUESTÕES ORGANIZACIONAIS

1) O sistema será empregado apenas em uma intranet, ou através da internet também?

O sistema deve estar disponível na intranet, visto que é um sistema de gerenciamento interno. Contudo, haverá certa interação dos técnicos de campo com o ambiente de forma remota (apenas atualizações de status).

2) Existiram privilégios para determinados usuários?

Sim, o acesso às funções administrativas será restrito a poucos usuários.

3) Que benefícios adicionais o sistema deve trazer em relação aos clientes?

O sistema deverá ser suficientemente ágil e estável para garantir atendimentos rápidos e eficientes aos clientes, de forma a cativá-los.

4) O sistema deverá gerar relatórios?

Sim, o sistema deverá gerar relatórios de atividades e de chamados.

5) O sistema deverá ter um *log* de operações?

Sim, pois isso tornará possível o rastreamento de eventuais problemas operacionais.

QUESTÕES ECONÔMICAS

1) O ambiente de desenvolvimento utilizado para o projeto será gratuito?

Sim, deverão ser utilizadas apenas ferramentas livres ou licenciadas para estudantes, ficando a cargo do comprador do produto a obtenção das licenças definitivas.

2) Qual a expectativa de entrega do sistema?

O sistema deverá ser entregue em cerca de oito meses (negociável).

3) Quem será responsável por efetuar manutenção no sistema, depois de entregue?

A equipe de desenvolvimento, por um tempo ainda a determinar.

4) Algum novo hardware será adquirido para este projeto?

As compras de novos equipamentos deverão ser evitadas, contudo, poderão ser efetuadas em casos extremos.