

## **Relatório do aplicativo Ouvidoria móvel da USP**

Ana Paula Oliveira Bertholdo  
Rafael Brito de Oliveira  
Suelen Goularte de Carvalho

### **RELATÓRIO DO APLICATIVO OUVIDORIA MÓVEL DA UNIVERSIDADE DE SÃO PAULO PARA DISCIPLINA DE COMPUTAÇÃO MÓVEL**

Professor: Dr. Alfredo Goldman vel Lejbman  
Monitores: Antonio Deusany de Carvalho Junior  
Gilmar Rocha de Oliveira Dias

São Paulo, abril de 2014



# Sumário

<b>Lista de Figuras</b>	<b>v</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Motivação . . . . .	1
<b>2 Manual do usuário</b>	<b>3</b>
2.1 Requisitos para o dispositivo móvel . . . . .	3
2.2 Aplicativo da Ouvidoria . . . . .	3
2.2.1 Login . . . . .	4
2.2.2 Lista de Incidentes . . . . .	4
2.2.3 Lista de departamentos . . . . .	5
2.2.4 Busca de incidentes por palavra-chave . . . . .	5
2.2.5 Apresentação de detalhes de cada incidente registrado . . . . .	7
2.2.6 Definição de status do incidente . . . . .	7
2.2.7 Apresentação de incidentes em mapa . . . . .	10
2.2.8 Sincronização e armazenamento dos incidentes . . . . .	10
2.3 Aplicativo do usuário . . . . .	11
2.3.1 Login . . . . .	11
2.3.2 Registro e envio de incidente . . . . .	11
2.3.3 Lista de incidentes não anviados . . . . .	12
<b>3 Estrutura do código</b>	<b>15</b>
3.1 Principais classes para o aplicativo da Ouvidoria . . . . .	15
3.1.1 Principais classes do pacote Model . . . . .	15
3.1.2 Principais classes do pacote View . . . . .	17
3.1.3 Principais classes do pacote Preferences . . . . .	18
3.1.4 Principais classes do pacote Task . . . . .	18
3.1.5 Principais classes do pacote Util . . . . .	18
3.2 Principais classes para o aplicativo do Usuário . . . . .	19
3.2.1 Principais classes do pacote Dao . . . . .	19
3.2.2 Principais classes do pacote Location . . . . .	20
3.2.3 Principais classes do pacote Model . . . . .	20
3.2.4 Principais classes do pacote Preferences . . . . .	20
3.2.5 Principais classes do pacote Receiver . . . . .	21
3.2.6 Principais classes do pacote Task . . . . .	21

3.2.7	Principais classes do pacote Util . . . . .	21
3.2.8	Principais classes do pacote View . . . . .	21
3.2.9	Principais classes do pacote Web . . . . .	22
3.3	<i>Web Services</i> . . . . .	22
<b>4</b>	<b>Conclusões</b>	<b>23</b>
	<b>Referências Bibliográficas</b>	<b>25</b>

# Listas de Figuras

2.1	Interface de login para o aplicativo de Ouvidoria . . . . .	4
2.2	Interface de apresentação de todos os incidentes registrados . . . . .	5
2.3	Menu para visualização dos departamentos da USP . . . . .	6
2.4	Interface de apresentação de todos os departamentos da USP . . . . .	6
2.5	Interface para realizar a busca de incidentes por palavra-chave . . . . .	7
2.6	Interface para apresentação dos resultados da busca por palavra-chave . . . . .	8
2.7	Interface para apresentação de detalhes de um incidente com status aberto . . . . .	8
2.8	Interface para apresentação de detalhes de um incidente com status em andamento . . . . .	9
2.9	Interface para atualização do status de um incidente selecionado . . . . .	9
2.10	Interface para apresentação de incidentes no mapa . . . . .	10
2.11	Interface de login para o aplicativo do Usuário . . . . .	12
2.12	Interface de apresentação para inserir um novo incidente . . . . .	13
2.13	Interface de apresentação da lista de incidentes pendentes de envio . . . . .	14
3.1	Estrutura de pacotes Java do aplicativo da Ouvidoria . . . . .	16
3.2	Primaria parte da estrutura de pacotes Java do aplicativo d Usuário . . . . .	19
3.3	Segunda parte da estrutura de pacotes Java do aplicativo d Usuário . . . . .	20



# Capítulo 1

## Introdução

Ouvidoria é um órgão que tem a função de receber críticas, sugestões, denúncias, reclamações e que deve agir em defesa da comunidade. O profissional responsável por uma ouvidoria é chamado de ouvidor, que significa *ombudsman* (*ombuds*= representante; *man*= homem), que é uma palavra sueca que foi originalmente criada para designar o cargo de agente parlamentar de justiça para limitar os poderes do rei, mas que atualmente significa representante do cidadão [Bal].

Conforme [Bez], o primeiro *ombudsman* surgiu por meio da Constituição sueca de 1809 e constituiu-se em uma das mais importantes respostas aos perigos de uma Administração pública tirânica e para garantir os direitos e liberdades individuais.

Ainda segundo [Bez], "no Brasil, desde os anos noventa do século passado, tem-se visto um movimento de "ombudsmania", com o surgimento de centenas de ouvidores e *ombudsmen* públicos e privados".

A função principal de um *ombudsman* é "defender as manifestações do cidadão perante algum órgão, seja uma reclamação, um elogio ou uma sugestão. É o cargo dado a uma pessoa que vai ter relação direta com o dirigente da instituição, seja ela pública ou privada, para garantir um eficiente canal de comunicação entre o emissor e o receptor de serviços" [Bal].

A Ouvidoria móvel da Universidade de São Paulo (USP) é um aplicativo para atendimento à comunidade USP com o objetivo de encaminhar e acompanhar críticas e sugestões sobre incidentes ocorridos no interior da Cidade Universitária. É um meio mediador que exerce um elo de comunicação entre a universidade e a comunidade realizando atendimento eletrônico por meio de dispositivos móveis, como smartphones.

### 1.1 Motivação

Conforme [dO], faz parte do exercício das atividades dos Ouvidores/*Ombudsman* defender os direitos dos seres humanos, embasando suas ações por princípios éticos, morais e constitucionais. Sendo assim, os membros da Associação Brasileira de Ouvidores (ABO) instituíram um Código de Ética, que dentre os vinte e três termos apresentados, podemos destacar a preservação e respeito dos princípios da "Declaração Universal dos Direitos Humanos, da Constituição Federal e das Constituições Estaduais"; o estabelecimento de canais de comunicação de forma aberta, honesta e objetiva, procurando sempre facilitar e agilizar as informações e a ação com transparência, integridade e respeito.

Essas são as principais motivações que conduziram ao desenvolvimento de um aplicativo móvel para a implementação de uma Ouvidoria que atenda à comunidade da USP, tendo como base os princípios do código de ética dos ouvidores [dO].

A possibilidade de criação de um meio de comunicação direta com órgãos da universidade, de modo a apresentar críticas, sugestões ou reclamações a respeito de qualquer local ou evento da USP, contribui para a construção de uma universidade mais inclusiva e na qual seus membros podem atuar ativamente por sua melhoria.



# Capítulo 2

## Manual do usuário

A aplicação desenvolvida é um sistema de ouvidoria móvel, onde os usuários podem anunciar eventos ou condições que mereçam atenção das entidades de segurança e manutenção da USP. A ouvidoria móvel pode ser acessada por qualquer membro da comunidade da USP que possua uma conta no STOA. O projeto Stoa é "uma rede de colaboração dos estudantes, professores, funcionários e ex-membros da Universidade de São Paulo (USP). Os objetivos do Stoa são promover uma maior interação entre os membros da comunidade USP, criar um espaço onde cada pessoa dentro da Universidade tenha uma identidade digital de fácil acesso, tanto para quem está dentro da USP, quanto para a comunidade externa, e fornecer um sistema que facilite aos professores a administração de seus cursos para os estudantes"[\[Sto\]](#).

Para desenvolvimento do sistema de ouvidoria móvel da USP, foram criados dois aplicativos para públicos-alvo diferentes. O primeiro é o aplicativo para os usuários, ou membros da comunidade USP que possuem uma conta no STOA, e o segundo aplicativo é destinado a funcionários responsáveis pela segurança e manutenção do campus, para os quais serão encaminhadas as solicitações dos usuários. Os dois aplicativos foram desenvolvidos tendo como base os padrões de cores da USP.

### 2.1 Requisitos para o dispositivo móvel

Os dois aplicativos, do usuário membro da comunidade USP e da Ouvidoria, para funcionários da Segurança e Manutenção da USP, foram desenvolvidos tendo como tecnologia alvo a versão 4.4.2 do Android. Essa versão é necessária para que as funcionalidades de mapas e localização por GPS implementadas funcionem corretamente nos dispositivos móveis. A versão 1.7 do Java é utilizada para execução das duas aplicações.

### 2.2 Aplicativo da Ouvidoria

A aplicação da Ouvidoria possui as seguintes funcionalidades, que serão descritas em detalhes nas próximas seções.

1. Login
2. Lista de Incidentes
3. Lista de departamentos
4. Busca de incidentes por palavra-chave
5. Apresentação de detalhes de cada incidente registrado
6. Definição de status do incidente
7. Apresentação de incidentes em mapa
8. Sincronização e armazenamento dos incidentes

### 2.2.1 Login

A interface inicial do sistema de Ouvidoria é a tela de Login apresentada na Figura2.1. O usuário deste aplicativo é um funcionário responsável pela segurança e manutenção da USP e que fará o tratamento dos incidentes registrados pelos membros da comunidade.

Para que o login seja realizado com sucesso, o funcionário precisa informar o mesmo login e senha cadastrados na rede social STOA. Caso ainda não possua cadastro na rede social, este pode ser realizado por meio do link: <http://social.stoa.usp.br/account/signup>

Após informar o seu número USP e senha cadastrada no STOA, o aplicativo fará a verificação da conta do usuário e caso esteja correta, será apresentada a tela para apresentação de incidentes. Caso os dados de login e/ou senha estejam incorretos, o usuário receberá uma mensagem de erro informando que os dados para acesso ao sistema estavam incorretos.



**Figura 2.1:** Interface de login para o aplicativo de Ouvidoria

### 2.2.2 Lista de Incidentes

A tela de lista de incidentes apresenta os incidentes recuperados após sincronização com o servidor Web. No primeiro acesso, nenhum incidente será apresentado, pois é necessário que o usuário clique no botão Sincronizar para buscar todos os incidentes registrados no servidor Web. Esse procedimento deve ser realizado sempre que o usuário desejar obter os últimos incidentes do servidor.

A partir de um segundo acesso pelo usuário, os incidentes já carregados para o aplicativo não precisarão de nova busca no servidor. Os novos incidentes registrados serão carregados quando o usuário clicar no botão Sincronizar da tela de apresentação de incidentes, atualizando a lista de incidentes apresentada ao usuário.

Na tela de apresentação da lista de incidentes são exibidos quatro botões no lado superior direito da interface, são eles respectivamente da esquerda para a direita:

1. Botão Sincronizar (ícone de sincronização);

2. Botão para busca de incidentes por palavra-chave (ícone de lupa);
3. Botão para exibição dos incidentes no mapa (ícone de mapa); e
4. Botão para filtro dos incidentes por departamento (ícone de filtro);

A Figura2.2 apresenta a interface para apresentação dos incidentes registrados do aplicativo de Ouvidoria da USP.



**Figura 2.2:** Interface de apresentação de todos os incidentes registrados

### 2.2.3 Lista de departamentos

O botão filtro dos incidentes por departamento apresentado na tela de Lista de incidentes, permite filtrar os incidentes por departamento da USP, ordenando-os pelo departamento selecionado. Após clicar neste botão é possível acessar um menu de acesso à lista de departamentos da USP, conforme Figura2.3.

Após acessar o menu, o usuário é direcionado para uma tela com a lista de departamentos da Universidade de São Paulo, conforme Figura2.4. Neste ponto, o usuário pode selecionar o departamento pelo qual deseja que os incidentes sejam filtrados e uma mensagem de sucesso na operação é apresentada após a ordenação dos incidentes pelo departamento selecionado. Após a seleção do departamento, é possível visualizar os incidentes na tela de Lista de incidentes ordenados pelo departamento da USP selecionado como filtro.

### 2.2.4 Busca de incidentes por palavra-chave

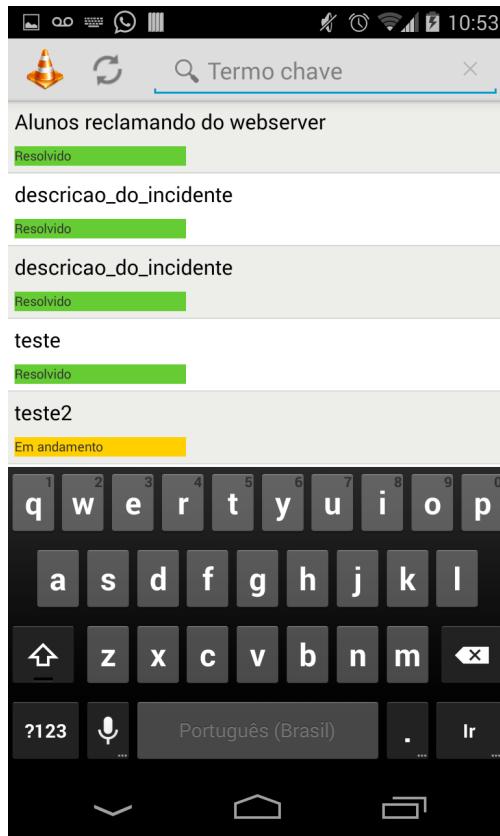
Botão para busca de incidentes por palavra-chave (ícone de lupa) permite que seja feita uma busca pela descrição do incidente registrado, conforme Figura2.5.



**Figura 2.3:** Menu para visualização dos departamentos da USP



**Figura 2.4:** Interface de apresentação de todos os departamentos da USP



**Figura 2.5:** Interface para realizar a busca de incidentes por palavra-chave

O usuário informa uma palavra-chave presente na descrição do incidente cadastrado e o aplicativo apresentará uma lista de todos os incidentes encontrados que possuam a palavra-chave informada em sua descrição. O resultado da busca será apresentado na lista de incidentes, conforme Figura2.6.

### 2.2.5 Apresentação de detalhes de cada incidente registrado

Sempre que uma lista de incidentes for apresentada no aplicativo, é possível acessar informações detalhadas sobre cada um dos incidentes cadastrados. O usuário terá acesso às informações adicionais ao clicar em um incidente apresentado na lista. O aplicativo exibirá uma tela com os detalhes do incidente: número do incidente, o login do usuário que cadastrou o incidente, o nome do departamento da USP onde o incidente ocorreu, a descrição do incidente, a data de criação do incidente no sistema, a data de atualização do incidente no sistema, a foto do incidente, a localização do incidente no mapa por meio das coordenadas GPS cadastradas e o status do incidente. As figuras 2.7 e 2.8 apresentam duas telas de Apresentação de detalhes de cada incidente registrado. A Figura2.7 exibe um incidente que está com o status aberto e a Figura2.8 exibe um incidente que está com o status Em andamento.

### 2.2.6 Definição de status do incidente

Ao clicar sobre o status atual na tela para Apresentação de detalhes de cada incidente registrado, será exibida uma janela pop-up onde será possível escolher um novo status para o incidente registrado. A Figura2.9 exibe a tela para atualização do status de um incidente selecionado.

Os estados possíveis para cada incidente são: Aberto, Em Andamento, Resolvido e Escondido.

- O status aberto define incidentes que foram registrados no sistema, por membros da comunidade USP, mas que ainda não receberam nenhum tratamento específico para o problema ou evento relatado.



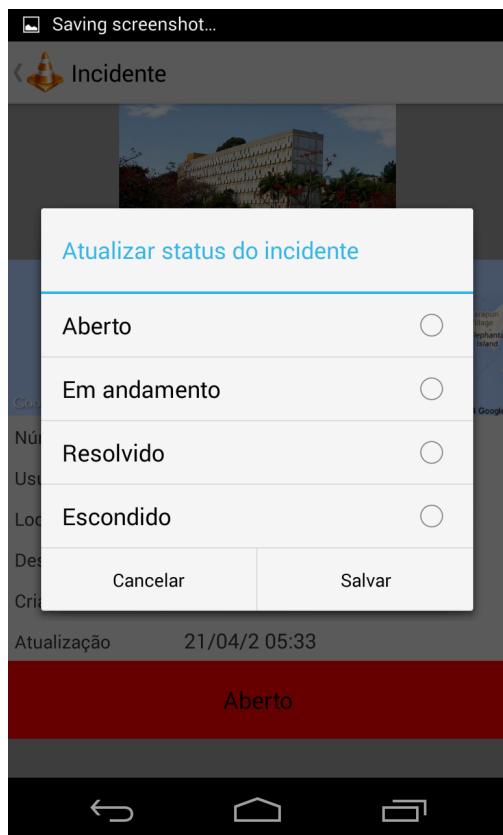
**Figura 2.6:** Interface para apresentação dos resultados da busca por palavra-chave



**Figura 2.7:** Interface para apresentação de detalhes de um incidente com status aberto



**Figura 2.8:** Interface para apresentação de detalhes de um incidente com status em andamento

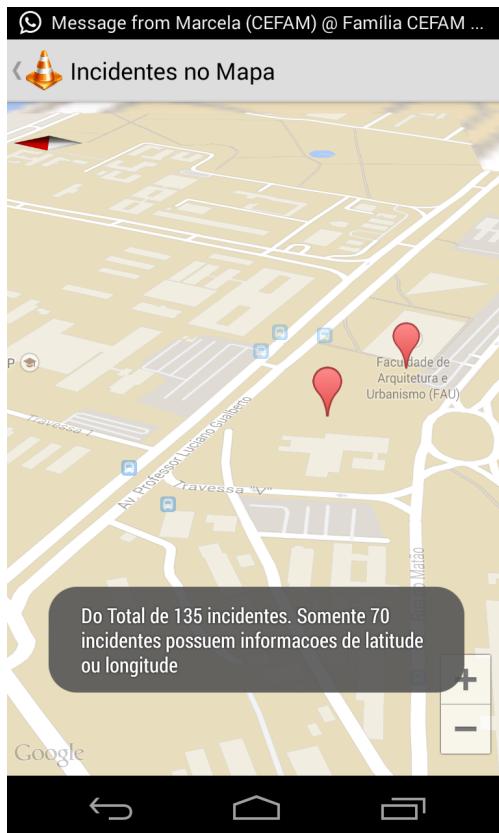


**Figura 2.9:** Interface para atualização do status de um incidente selecionado

- O status Em andamento define incidentes que já estão em tratamento pelos funcionários da Segurança e Manutenção da USP.
- O status Resolvido define incidentes que já foram tratados pelos funcionários da Segurança e Manutenção da USP.
- O status Escondido define incidentes que foram cadastrados no sistema, mas que não estão visíveis para todos os usuários.

### 2.2.7 Apresentação de incidentes em mapa

O botão para exibição dos incidentes no mapa (ícone de mapa) permite visualizar no mapa todos os incidentes cadastrados com coordenadas GPS. Marcadores indicam o local do incidente no mapa da USP. A Figura 2.10 apresenta a interface para visualização de incidentes no mapa.



**Figura 2.10:** Interface para apresentação de incidentes no mapa

### 2.2.8 Sincronização e armazenamento dos incidentes

O botão Sincronizar (ícone de sincronização) apresentado na interface de Lista de incidentes permite que o usuário recupere todos os incidentes que ainda não foram baixados para seu aplicativo local. Esse é o motivo pelo qual, quando o usuário acessa pela primeira vez a tela de incidentes, a tela será apresentada sem nenhum incidente. É preciso clicar no botão de Sincronização para ter acesso aos incidentes registrados no Servidor Web e carregá-los para o aplicativo local. A partir de um segundo acesso à interface de Lista de incidentes, apenas os últimos incidentes registrados no servidor e que ainda não foram carregados no aplicativo local serão recuperados, agilizando o processo de sincronização e atualização dos dados.

Os incidentes registrados pelos usuários são enviados diretamente para o Servidor Web. Caso algum problema ocorra no envio, por exemplo, caso o usuário esteja sem acesso à Internet, então

os dados do incidente registrado no aplicativo do Usuário (membro da comunidade USP) são armazenados em um banco de dados local (SQLite) de modo que quando o usuário clique no botão Sincronizar os dados dos incidentes do Aplicativo da Ouvidoria, os incidentes armazenados apenas no banco de dados local (SQLite), possam ser enviados para o servidor Web, mantendo todos os dados atualizados.

## 2.3 Aplicativo do usuário

O aplicativo do usuário tem como principal objetivo permitir que a comunidade USP possa registrar incidentes ocorridos dentro da Cidade Universitária, informando o local do evento, com dados do GPS, e de modo a inserir registros fotográficos.

Se a rede conseguir estabelecer conexão com o servidor Web, para o qual cada incidente será enviado, então o usuário será informado que seu incidente foi registrado com sucesso. O objetivo é mantê-lo informado de suas ações no sistema.

Caso, não seja possível estabelecer conexão com o servidor Web, o incidente registrado pelo usuário será armazenado em um banco de dados (SQLite), de modo que o registro não é perdido. Quando os dados de incidentes forem sincronizados, a partir do aplicativo da Ouvidoria, os registros armazenados no SQLite serão enviados para o servidor Web.

A aplicação do usuário possui as seguintes funcionalidades, que serão descritas em detalhes nas próximas seções.:

1. Login
2. Registro e envio de incidente
3. Lista de incidentes não enviados

### 2.3.1 Login

A interface inicial do sistema do Usuário é a tela de Login apresentada na Figura 2.1.

O usuário deste aplicativo pode ser qualquer pessoa que possua um número USP e deseja registrar algum incidente para que os responsáveis o tratem.

Para que o login seja realizado com sucesso, o usuário precisa informar o mesmo login e senha cadastrados na rede social STOA. Caso ainda não possua cadastro na rede social, este pode ser realizado por meio do link: <http://social.stoa.usp.br/account/signup>

Após informar o seu número USP e senha cadastrada no STOA, o aplicativo fará a verificação da conta do usuário e caso esteja correta, será apresentada a tela para apresentação de incidentes. Caso os dados de login e/ou senha estejam incorretos, o usuário receberá uma mensagem de erro informando que os dados para acesso ao sistema estavam incorretos. Caso não haja conexão com a internet, o mesmo será informado disso.

### 2.3.2 Registro e envio de incidente

A tela de novo incidentes apresenta os campos a serem preenchidos para registro e envio de um novo incidente.

Nesta tela deverão ser preenchidas as seguintes informações e depois clicar em "Salvar":

1. Categoria - Basta iniciar a digitação que será exibida uma lista de categorias pré-existentes.
2. Descrição do incidente - Preencher com uma breve descrição o incidente.
3. Salvar - Botão no canto superior direito ao qual realiza o envio dos dados do incidente ao servidor no caso de haver conexão de dados, do contrário, salva o incidente no banco de dados local para envio posterior, quando houver conexão.

A Figura 2.13 apresenta a interface para o registo e envio de incidentes.

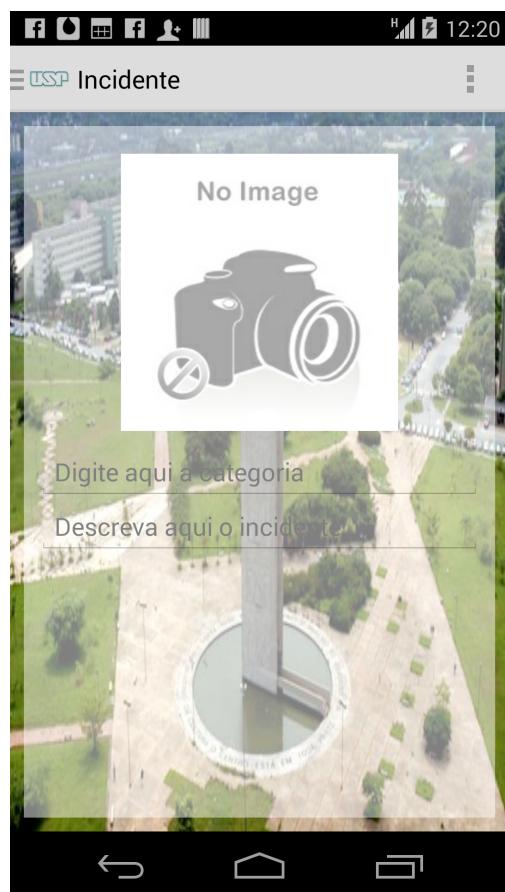


**Figura 2.11:** Interface de login para o aplicativo do Usuário

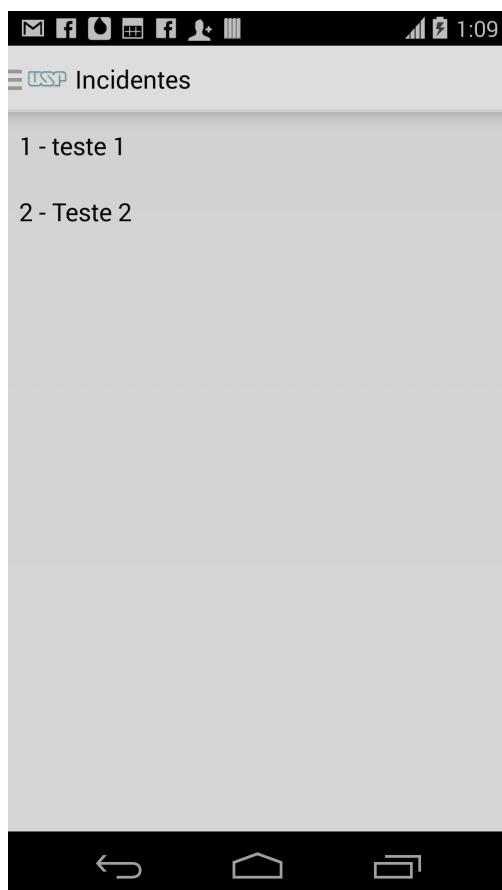
### 2.3.3 Lista de incidentes não enviados

Nesta parte do sistema de Usuário é exibida a lista de incidentes que foram salvos porém ainda não enviados ao servidor por falta de conexão de dados no momento do salvamento. Os incidentes nesta lista serão enviados ao servidor automaticamente assim que uma conexão de dados for identificada. Após o envio com sucesso, serão retirados desta lista.

A Figura?? apresenta a interface com a lista de incidentes pendentes de envio.



**Figura 2.12:** Interface de apresentação para inserir um novo incidente



**Figura 2.13:** Interface de apresentação da lista de incidentes pendentes de envio

# Capítulo 3

## Estrutura do código

O sistema trata de:

1. Comunicação com *webservers* e autenticação de usuário:

- Foi utilizado um *webserver* especialmente desenvolvido para o projeto com a funcionalidade de receber e responder requisições dos usuários utilizando a notação *JavaScript Object Notation* (JSON).
- Será realizada a autenticação de usuário por meio do login de usuários no STOA (<http://stoa.usp.br/>).

2. Funcionalidades extras dos dispositivos móveis:

- Câmera, Áudio, Vídeo, GPS, Acelerômetro.
- Compartilhamento de informações.

### 3.1 Principais classes para o aplicativo da Ouvidoria

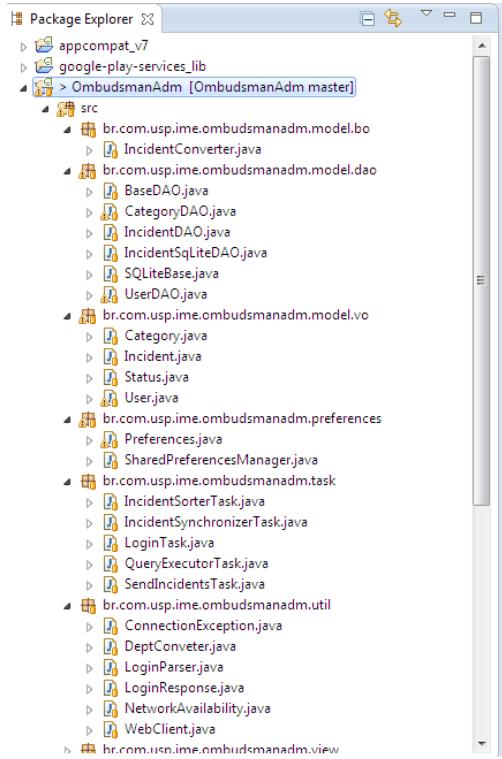
A Figura 3.1 exibe a estrutura de pastas e pacotes do aplicativo *OmbudsmanAdm*, disponível em: <https://github.com/rabriol/OmbudsmanAdm>, que contém o conteúdo relacionado ao aplicativo acessado pelos funcionários da Segurança e Manutenção da USP.

Os pacotes do aplicativo da Ouvidoria começam com o padrão: br.com.usp.ime.ombudsmanadm. A partir desse prefixo, são adicionados os sufixos que especificam o conteúdo exato de cada pacote da pasta *Source* (src). Os pacotes na hierarquia de classes Java do aplicativo da Ouvidoria finalizam com os sufixos *Model*, *View*, *Preferences*, *Task* e *Util*. O sistema foi implementado utilizando o padrão de projeto *Model-View-Controller*(MVC), por esse motivo, as classes implementadas no pacote *Model*, correspondem a classes que contém a lógica de negócios do sistema, bem como a lógica de manipulação das informações armazenadas no banco de dados. As classes Java inseridas no pacote *View*, constituem as classes que tratam das interfaces ou visualização do sistema e as classes que estão em *View.adapter* são as que fazem a intermediação entre as classes de visualização e de negócio do sistema, de modo a implementarem o controle entre a *View* e o *Model*.

O pacote *Preferences* foi criado para gerenciar as preferências do usuário e registra quando o usuário está logado no sistema com seu identificador único. O pacote *Task* contém as classes responsáveis pela comunicação com o servidor Web e que estendem a classe *AsyncTask*. O pacote *Util* contém as classes utilitárias do aplicativo da Ouvidoria. As principais classes Java do aplicativo da Ouvidoria serão explicados em maiores detalhes nas próximas subseções.

#### 3.1.1 Principais classes do pacote Model

O pacote *Model*, a saber, br.com.usp.ime.ombudsmanadm.model é composto de *Model.vo*, *Model.bo* e *Model.dao*. Como dito anteriormente, o pacote *Model* é responsável pelas classes que implementam a lógica de negócio da aplicação e que fazem a manipulação de informações armazenadas no banco de dados.



**Figura 3.1:** Estrutura de pacotes Java do aplicativo da Ouvidoria

O pacote Model.vo, refere-se às classes responsáveis pelos modelos de valores de objetos, ou *Value Object* (VO). São as classes que representam os principais objetos manipulados pelo sistema: *Category*, *Incident*, *Status* e *User*. Cada um desses objetos é representado por uma classe, com exceção do Status que é representado por um Enum. Todos possuem métodos *Get* e *Set* de modo que seja possível acessar, visualizar e modificar seus atributos.

O pacote Model.bo, refere-se às classes responsáveis pela lógica de negócios do sistema e que possui a classe *IncidentConverter* que realiza a conversão de uma lista de incidentes para uma String JSON e a partir de uma String JSON obtém uma lista de Incidentes. Os métodos que realizam essas atividades, respectivamente, são: public String toJSON(List<Incident> incidents); e public List<Incident> toIncidentList(String json). Esses dois métodos são essenciais para que os dados recuperados no formato JSON do servidor Web possam ser convertidos em uma lista de incidentes que pode ser armazenada no banco de dados do sistema (SQLite) e a partir de uma lista de incidentes recuperadas do banco de dados é possível convertê-las para o formato JSON de modo a enviá-las para o servidor Web.

O pacote Model.dao contém as classes e interfaces de acesso ao banco de dados SQLite. A interface *IncidentDAO* é responsável por encapsular os métodos implementados na classe *IncidentSQLiteDAO*, que correspondem a todos os métodos de manipulação da tabela Incidente no banco de dados, a saber, *insert*, *update*, *delete*, *getIncidents*, *getIncidentsByKeyValue*, *getIncidentsById*, *getLastIncidentId* e *getIncidentById*. Além dos métodos de manipulação dos incidentes armazenados no banco, a classe também é responsável pela criação e atualização da tabela Incidentes por meio dos métodos, *onCreate* e *onUpgrade*, respectivamente.

A classe *IncidentSQLiteDAO* além de implementar a interface *IncidentDAO*, também estende a classe *SQLiteOpenHelper*. A classe *SQLiteOpenHelper* é uma classe utilitária para gerenciar a criação de banco de dados e a gestão de versões, onde é possível criar subclasses implementando os métodos *onCreate(SQLiteDatabase)*, *onUpgrade(SQLiteDatabase, int, int)* e a *SQLiteOpenHelper* cuida de abrir o banco de dados se este existir, criá-lo caso não exista e atualizá-lo se necessário.

A classe *SQLiteBase* também estende a classe *SQLiteOpenHelper* e é responsável por criar as tabelas *Category* e *User*, implementando os métodos *onCreate* e *onUpgrade*, de acordo com a

herança de *SQLiteOpenHelper*.

A classe abstrata *BaseDAO* obtém uma instância de *SQLiteBase* e define o método *getDDL*, que é usado pelas classes *CategoryDAO* e *UserDAO*, para obter o *Domain Definition Language* (DDL) das duas tabelas *Category* e *User*.

A classe *CategoryDAO* estende a classe abstrata *BaseDAO* e portanto implementa o método *getDDL* com a criação da tabela *Category*. A classe possui todos os métodos de manipulação do objeto *Category*, que refere-se aos departamentos da Universidade de São Paulo, portanto possui os métodos *insert(Category)*, *insert(List<Category>)*, *getAll* e *getCategoryFrom(Cursor)*.

A classe *UserDAO* também estende a classe abstrata *BaseDAO* e portanto implementa o método *getDDL* com a criação da tabela *User*. Assim como a classe *CategoryDAO*, realiza a manipulação dos dados de usuário e implementa os métodos *insert(User)*, *getUniqueUser* e *getUserFrom(Cursor)*.

### 3.1.2 Principais classes do pacote View

O pacote *View* contém as classes que tratam das interfaces ou visualização do sistema e as classes que estão em *View.adapter* são as que fazem a intermediação entre as classes de visualização e de negócio do sistema, de modo a implementarem o controle entre a *View* e o *Model*.

Dentro do pacote *View* e *View.map*, temos todas as classes que estão associadas a interfaces do sistema e que possuem em comum o sufixo *Activity*, representando uma *Activity* do *Android*.

A classe *IncidentActivity* é responsável pela apresentação da lista de incidentes, por meio do método *onCreate(Bundle)*, por apresentar as opções de menu por meio do método *onCreateOptionsMenu(Menu)* e por abrir as telas de acordo com a seleção do menu por meio do método *onOptionsItemSelected(MenuItem)*.

As opções embora sejam descritas em código como opções de menu, são exibidas na tela por meio de ícones de acesso ao mapa, a busca por palavras-chave, entre outras opções. O método *onSyncReturn* limpa a lista e traz todos os incidentes novos do servidor Web após adaptá-los de *Json* para uma lista de incidentes que será exibida na interface.

A classe *IncidentFormActivity* é responsável pela edição de um incidente cadastrado pelo usuário, por meio do aplicativo voltado para membros da comunidade USP. No aplicativo da Ouvidoria, que tem como público-alvo os funcionários da área de Segurança e Manutenção da USP, esta interface possibilita que mudem o status de tratamento de cada incidente de aberto para Em andamento, Resolvido ou Escondido. Esta interface exibe todos os detalhes cadastrados para cada incidente, incluindo foto, informações geográficas do GPS, departamento, entre outras.

*LoginActivity* é responsável pelo login do funcionário da USP no sistema de Ouvidoria, que será realizado baseado no cadastro do funcionário na rede STOA. A interface retornará uma mensagem de sucesso para funcionários cadastrados no STOA e retornará uma mensagem de erro caso o funcionário não possua uma conta no STOA e portanto não pode acessar o sistema. Caso o usuário efetue login com sucesso, serão inicializada a *activity* para listar os incidentes deste usuário.

A classe *SearchResultActivity* é responsável por exibir os resultados encontrados para a busca por palavras-chave. A interface apresenta a quantidade de resultados encontrados para a busca realizada. A classe *SortedDepartmentActivity* é responsável pela ordenação dos incidentes por departamento da USP e por fim, a classe *SortedIncidentActivity* é responsável pela ordenação dos incidentes.

A classe *IncidentsMapActivity* no pacote *View.map* é responsável pela apresentação dos incidentes registrado em um mapa, de acordo com a posição GPS dos incidentes registrados. Cada incidente é apresentado com um marcador que exibe sua descrição e o departamento da USP que ocorreu. O mapa é centralizado no Instituto de Matemática e Estatística (IME) da USP. Esta classe usa a classe *GoogleMap* para exibir o mapa na interface.

Como classes do *Controller* (Controlador) temos as classes do pacote *View.Adapter* que adaptam os atributos a serem exibidos na interface de acordo com mudanças no conteúdo de cada objeto.

Na classe *IncidentListAdapter* os incidentes de uma lista são atualizados de acordo com a mudança de status de tratamento. Por exemplo, os incidentes com status em aberto são coloridos com a cor vermelha, definida no arquivo *res/values/colors.xml*. Para o status resolvido, a cor passa a ser verde, para o status em andamento a cor é amarela. O mesmo ocorre na classe *SearchListAdapter*,

para o resultado de uma busca. A classe *SortedIncidentListAdapter* apenas configura para que cada incidente de uma lista de incidentes seja exibido com uma cor diferente de linha para facilitar a sua visualização. A cor também neste caso é definida no arquivo res/values/colors.xml.

### 3.1.3 Principais classes do pacote Preferences

Como dito anteriormente, o pacote *Preferences* foi criado para gerenciar as preferências do usuário e registra quando o usuário está logado no sistema com seu identificador único. O pacote possui duas classes: *SharedPreferencesManager* e *Preferences*.

A classe *SharedPreferencesManager* gerencia as preferências compartilhadas para o aplicativo da Ouvidoria, possibilitando acessar, visualizar e atualizar as preferências para o aplicativo da Ouvidoria, por meio de métodos *Get* e *Set*. A classe *Preferences* recupera uma instância de *SharedPreferencesManager* e recupera as categorias de localização (departamentos da USP) e o usuário logado.

### 3.1.4 Principais classes do pacote Task

O pacote *Task* contém as classes responsáveis pela comunicação com o servidor Web e que estendem a classe *AsyncTask*. A classe *AsyncTask* permite realizar operações em *background* e publicar resultados sobre a *thread* de interface de usuário sem precisar manipular *threads* e/ou *handlers*. Esta classe foi projetada para ser uma classe utilitária em torno das classes *Thread* e *Handler* e não constitui um arcabouço de *threading* genérico. A classe deve idealmente ser usada para curtas operações, que devem ser de poucos segundo no máximo.

A classe *IncidentSorterTask* é responsável por ordenar os incidentes por departamento, utilizando-se da classe *DeptConverter* para realizar a conversão de departamentos. A classe *IncidentSynchronizerTask* é responsável por sincronizar o aplicativo com o servidor Web, por meio da URL <http://uspservices.deusanyjunior.dj/incidente/%s.json>, de modo a obter os últimos incidentes registrados no servidor e apresentá-los na lista de incidentes do funcionário. Essa operação depende de uma conexão com a Internet, que caso não esteja ativa, retornará uma mensagem indicando que não existe conexão com a Internet para realizar a operação.

A classe *LoginTask* é responsável pelo login do usuário (funcionário da área de Segurança e Manutenção da USP) no aplicativo da Ouvidoria. O login e senha do usuário será verificado em <https://maxwell.stoa.usp.br/plugin/stoa/authenticate>. Se o usuário possuir uma conta no STOA, será retornada uma mensagem de sucesso e a tela com a lista de incidentes será apresentada. Enquanto o aplicativo acessa a url acima, uma mensagem de progresso da operação é apresentada ao usuário: "Verificando usuário..".

A classe *QueryExecutorTask* é responsável por buscar os incidentes no banco de dados e apresenta mensagem de progresso na operação para o usuário e, por fim, a classe *SendIncidentsTask* é responsável por enviar os incidentes armazenados no banco de dados SQLite para o servidor Web por meio da URL: <http://uspservices.deusanyjunior.dj/incidente>. Também uma mensagem com o progresso da operação é exibida ao usuário.

### 3.1.5 Principais classes do pacote Util

O pacote *Util* contém as classes utilitárias do aplicativo da Ouvidoria. A classe *ConnectionException* trata de exceções que podem ocorrer durante a conexão com o servidor Web. A classe *DeptConverter* é responsável pela conversão de uma abreviatura de departamento para o nome completo do departamento e vice-versa.

A classe *LoginParser* por realizar a conversão de uma string JSON que traz os dados de login, a saber, número USP, e-mail e *username* para um objeto *User*, e seta o objeto *User* em um objeto *LoginResponse* que é uma classe também no pacote *Util* que representa um objeto login com os dados de login recebidos via JSON e o status recebido via JSON, sendo status Ok para quando o

objeto JSON retorna um login válido ou uma string de erro que será setada em *LoginResponse*, caso o login não seja efetuado com sucesso.

A classe *NetworkAvailability* utiliza-se da classe *ConnectivityManager* para gerenciar a conectividade da rede e da classe *NetworkInfo* para obter informações da rede, retornando uma classe *TypeNetworkAnalyser* que retorna o tipo de rede em uso pelo usuário, verificando se é uma rede 3G, se é uma rede *Wifi*.

Por fim, temos a classe *WebClient* que é uma classe importante para realizar a conexão com o servidor Web, implementando os métodos *Http Get* e *Http Post*, os quais recuperam e enviam dados via JSON para o servidor Web, respectivamente.

## 3.2 Principais classes para o aplicativo do Usuário

Os pacotes do aplicativo do usuário começam com o padrão `br.com.ouvidoria.client`. A partir desse prefixo, são adicionados os sufixos que especificam o conteúdo exato de cada pacote da pasta *Source* (src). Os pacotes na hierarquia de classes Java do aplicativo do Usuário para registro de incidentes finalizam com os sufixos *dao*, *location*, *model*, *preferences*, *receiver*, *task*, *util*, *view* e *web*.

As classes no pacote *Dao* fazem acesso e manipulam informações armazenadas no banco de dados. O pacote *Location* é responsável pela implementação para ter acesso à localização do usuário. O pacote *Model* contém a lógica de negócios do sistema. O pacote *Preferences* é responsável pela configuração de preferências do usuário. O pacote *Receiver* é responsável por enviar uma mensagem ao usuário quando o incidente por ele registrado for recebido com sucesso no servidor Web. O pacote *Task* executa tarefas por meio da conexão com o servidor Web. O pacote *Util* contém todos as classes utilitárias que são utilizadas para garantir a execução do projeto. O pacote *View* é responsável pelo tratamento das interfaces ou visualização do sistema e por fim, o pacote *Web* que contém a classe *WebClient*, principal responsável pela conexão com o servidor Web via JSON.

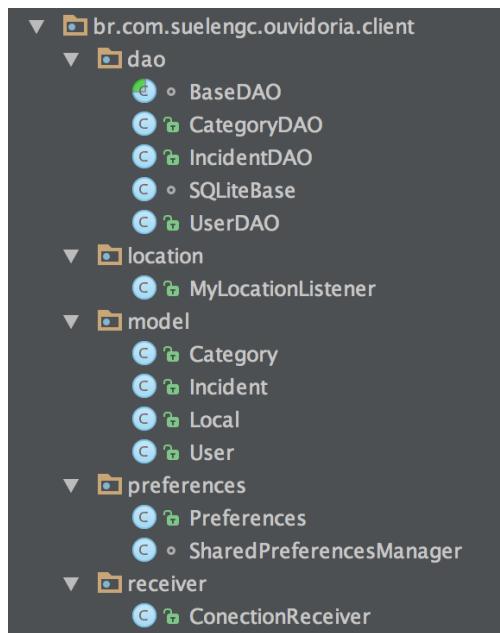
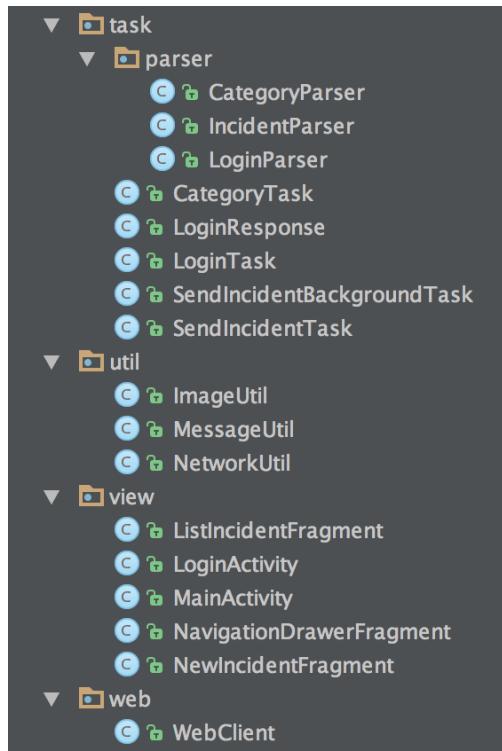


Figura 3.2: Primeira parte da estrutura de pacotes Java do aplicativo d Usuário

### 3.2.1 Principais classes do pacote Dao

O pacote *Dao* contém as classes de acesso ao banco de dados SQLite. A classe *SQLiteBase* estende a classe *SQLiteOpenHelper*, como descrito na seção 3.1.1, sobre Model.Dao. A classe tem o objetivo de criar o banco *OuvidoriaBD* e todas as tabelas do aplicativo do usuário, a saber, *Category*, *Incident* e *User*.



**Figura 3.3:** Segunda parte da estrutura de pacotes Java do aplicativo d Usuário

A classe abstrata `BaseDao` é responsável por recuperar o contexto da classe `SQLiteBase` e é estendida por todas as classes de acesso às tabelas do banco: `CategoryDAO`, `IncidentDAO` e `UserDAO`. Cada uma dessas classes realiza manipulações com informações de categorias (ou departamentos da USP), incidentes e usuários, respectivamente.

### 3.2.2 Principais classes do pacote Location

O pacote Location contém a classe `MyLocationListener` que implementa `LocationListener`. Esta classe tem como objetivo recuperar a localização do usuário, utilizando-se das classes `LocationListenerCallback` e `LocationManager`, as quais possibilitam encontrar o usuário pelo provedor da rede, mesmo que o GPS esteja desligado.

### 3.2.3 Principais classes do pacote Model

O pacote Model contém os valores de modelos de objetos que são manipulados pelo aplicativo, ou seja, (Value-Object), os quais representam os principais objetos do sistema, a saber, User, Incident, Category e Local. Todos são descritos por meio de classes Java que permitem o acesso, visualização e alteração de seus atributos por meio de métodos `Get` e `Set`.

### 3.2.4 Principais classes do pacote Preferences

O pacote Preferences contém as classes `SharedPreferencesManager` e `Preferences`. A classe `SharedPreferencesManager` gerencia as preferências compartilhadas para o aplicativo do Usuário, possibilitando acessar, visualizar e atualizar as preferências para o aplicativo, por meio de métodos `Get` e `Set`. A classe `Preferences` recupera uma instância de `SharedPreferencesManager` e recupera as categorias de localização (departamentos da USP) e o usuário logado.

### 3.2.5 Principais classes do pacote Receiver

O pacote *Receiver* contém a classe *ConectionReceiver*. A classe *ConectionReceiver* estende a classe *BroadcastReceiver* e implementa o método *onReceive*, responsável por enviar uma mensagem ao usuário quando o incidente por ele registrado for recebido com sucesso pelo servidor Web. O incidente enviado para o servidor Web é removido do banco de dados SQLite.

### 3.2.6 Principais classes do pacote Task

O pacote *Task* contém as classes responsáveis pela comunicação com o servidor Web e que estendem a classe *AsyncTask*. A classe *AsyncTask* permite realizar operações em *background* e publicar resultados sobre a *thread* de interface de usuário sem precisar manipular *threads* e/ou *handlers*. Esta classe foi projetada para ser uma classe utilitária em torno das classes *Thread* e *Handler* e não constitui um arcabouço de *threading* genérico. A classe deve idealmente ser usada para curtas operações, que devem ser de poucos segundo no máximo.

A classe *CategoryTask* é responsável por recuperar as categorias de local do servidor Web. As categorias são os departamentos da USP. A classe estende *AsyncTask* e recupera as categorias por meio da URL <http://uspservices.deusanyjunior.dj/categoriaslocal.json>.

A classe *LoginParser* é responsável por realizar a conversão de uma string JSON que traz os dados de login e *status* de login, a saber: número USP, e-mail, *username* e *status* para o objeto *LoginResponse*. Esta classe contém informações se o login foi efetuado com sucesso ou não e as respectivas informações de erro ou informações do login.

A classe *SendIncidentTask* é responsável por enviar os incidentes para o servidor Web por meio da URL: <http://uspservices.deusanyjunior.dj/incidente>. Ela também exibe uma mensagem com o progresso da operação para o usuário.

A classe *SendIncidentBackgroundTask* fica atenta as alterações de conexão de dados e quando identifica uma conexão ativa, verifica se há incidentes no banco de dados, havendo, os envia para o servidor Web por meio da URL: <http://uspservices.deusanyjunior.dj/incidente>. Apenas haverá incidentes no banco de dados caso no momento do salvamento do incidente, não foi possível enviá-lo ao servidor por falta de conexão de dados. Após o envio em background do incidente, é enviada uma mensagem informativa ao usuário e apagado do banco o incidente em questão.

Dentro do pacote *Task*, existe o pacote parser, que contém as classes responsáveis pela conversão dos objetos para JSON e vice-versa. A classe *CategoryParser* realiza a conversão de uma string JSON para uma lista de *Category*. A classe *IncidentParser* realiza a conversão de um *Incident* para uma string JSON. E por fim, a classe *LoginParser* realiza a conversão de uma string JSON para um objeto *LoginResponse*.

### 3.2.7 Principais classes do pacote Util

O pacote *Util* contém as classes *ImageUtil*, *MassageUtil* e *NetworkUtil* para auxiliarem respectivamente no tratamento de imagens, mensagens e conexão com a rede. A classe *ImageUtil* se utiliza das classes *Bitmap*, *ByteArrayOutputStream*, *Base64* e *BitmapFactory* que torna possível recuperar a imagem no formato de 64bits.

### 3.2.8 Principais classes do pacote View

O pacote *View* contém as classes responsáveis pela visualização das informações pelo usuário.

A classe *ListIncidentFragment* é responsável pela visualização da lista de incidentes pendentes de envio ao servidor. Ela extende a classe *Fragment* que faz parte da API Fragment que auxilia na criação de telas responsivas.

A classe *LoginActivity* é responsável pela visualização da tela de login ao qual o usuário usa para se logar no sistema. Ela extende de *Activity* que é a classe responsável pela criação de telas no Android.

A classe *MainActivity* é responsável por compôr menu e conteúdo de tela e por realizar a transição dos fragmentos criados.

A classe *NavigationDrawerFragment* é responsável por montar e exibir o menu lateral denominado padrão *Navigation Drawer*.

A classe *NewIncidentFragment* é responsável por apresentar a tela de cadastro do incidente a ser enviado ao servidor. Esta é a tela apresentada imediatamente após realizado com sucesso o login. Ela extende a classe *Fragment* que faz parte da API Fragment que auxilia na criação de telas responsivas.

### 3.2.9 Principais classes do pacote Web

O pacote Web contém a classe *WebClient*, principal responsável pela conexão com o servidor Web via JSON. A classe *WebClient* é uma classe importante para realizar a conexão com o servidor Web, implementando os métodos *Http Get* e *Http Post*, os quais recuperam e enviam dados via JSON para o servidor Web, respectivamente.

## 3.3 Web Services

Para o serviço de autenticação no STOA foi necessário utilizar os seguintes dados via *JavaScript Object Notation* (JSON) com um contrato via Post, passando o número USP e a senha do usuário:  
URL: <https://social.stoa.usp.br/plugin/stoa/authenticate/>

Quando senha inválida: Retorna "ok":false,"error":"O par usuário/senha está incorreto."  
Quando senha correta: Retorna "ok":true,"nusp":"00000000","username":"xxx","email":"xxxusp.br".

Os registros de incidentes são enviados por meio de um contrato via post, informado o número USP, a descrição do incidente, a categoria da localização, o valor da latitude da localização do incidente em tipo Float, longitude da localização do incidente em tipo Float e a foto do incidente, um binário da imagem em Base 64, para a seguinte URL:

<http://uspservices.deusanyjunior.dj/incidente>

É possível requisitar novos incidentes, solicitando o último incidente registrado:  
<http://uspservices.deusanyjunior.dj/incidente.xml> (ou .json)

Solicitar todos os incidentes a partir de um determinado incidente:  
<http://uspservices.deusanyjunior.dj/incidente/20.xml> (ou .json)  
obs.: No exemplo acima, retornaria todos os incidentes a partir do vigésimo incidente.

Solicitar incidentes em uma faixa específica:  
[http://uspservices.deusanyjunior.dj/incidentrecords/get\\_incidents.xml?first=1&last=3](http://uspservices.deusanyjunior.dj/incidentrecords/get_incidents.xml?first=1&last=3)  
obs.: Retorna todos os incidentes dentro de um intervalo especificado. No exemplo acima, retornaria a partir do incidente de id 1 até o de id 3.

Solicitar incidentes em uma faixa específica em uma determinada localização:  
[http://uspservices.deusanyjunior.dj/incidentrecords/get\\_incidents.xml?first=1&last=3&localization=Instituto%20de%20Matemática%20e%20Estatística](http://uspservices.deusanyjunior.dj/incidentrecords/get_incidents.xml?first=1&last=3&localization=Instituto%20de%20Matemática%20e%20Estatística)  
obs.: No exemplo acima, retornaria todos os incidentes a partir do id 1 até o id 3 que fazem parte do Instituto de Matemática e Estatística da USP (IME).

## Capítulo 4

# Conclusões

O aplicativo foi desenvolvido utilizando como referência o material de aula da disciplina e estudos sobre a tecnologia Android [dSOB14]. Durante o processo de aprendizado da tecnologia, o grupo enfrentou alguns problemas com respeito à compatibilidade do aplicativo com smartphones de membros do grupo. A versão 2.3.6 do Android não é compatível com as funcionalidades de mapas e GPS que são apresentadas no aplicativo da Ouvidoria. Por esse motivo, o processo de desenvolvimento se tornou mais demorado, contudo foi possível implementar todas as funcionalidades solicitadas.

Por meio desse projeto foram exercitadas as principais características de implementação com a tecnologia Android e de utilização das principais APIs para uso de mapas e identificação da localização do usuário via GPS. Além disso, o conhecimento sobre o envio de dados via JSON por meio de um aplicativo móvel foi treinado neste exercício-projeto.



# Referências Bibliográficas

- [Bal] Cristina Balerini. Ombudsman interno: Sua empresa ainda vai ter um. [http://www.ouvidor.com.br/artigo\\_8.html](http://www.ouvidor.com.br/artigo_8.html). Último acesso em 20/04/2014.
- [Bez] Helga Maria Saboia Bezerra. Defensor do povo: origens do instituto do ombudsman e a malograda experiência brasileira. <http://direitoestadosociedade.jur.puc-rio.br/media/3bezerra36.pdf>. Último acesso em 20/04/2014.
- [dO] Associação Brasileira de Ouvidores. Código de Ética do ouvidor/ombudsman. <http://www.abonacional.org.br/artigo.php?codigo=8>. Último acesso em 20/04/2014.
- [dSOB14] Ricardo da Silva Ogliari e Robison Cris Brito. *Android - Do básico ao avançado*. Ciência Moderna, 1º edição, 2014.
- [Sto] Wiki Stoa. O que é o stoa? <http://wiki.stoa.usp.br/Stoa:Sobre>. Último acesso em 20/04/2014.