



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Wesley Belarmino da Silva

**PROTÓTIPO DE UM APLICATIVO PARA GESTÃO MUNICIPAL
DE OCORRÊNCIAS DE ORDEM PÚBLICA**

Americana, SP
2017



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

**PROTÓTIPO DE UM APLICATIVO PARA GESTÃO MUNICIPAL
DE OCORRÊNCIAS DE ORDEM PÚBLICA**

Wesley Belarmino da Silva

Trabalho de conclusão,
desenvolvido em cumprimento à
exigência curricular do Curso
Superior de Tecnologia em Análise e
Desenvolvimento de Sistemas da
Fatec-Americana, sob orientação do
Prof. Antônio Alfredo Lacerda.

Área: Tecnologia da Informação

Americana, SP
2017

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

S584p SILVA, Wesley Belarmino da
Protótipo de um aplicativo para gestão municipal de ocorrências de ordem pública. / Wesley Belarmino da SILVA. – Americana: 2017.
65f.
Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza
Orientador: Prof. Esp. Antonio Alfredo Lacerda
1. Dispositivos móveis – aplicativos I. LACERDA, Antonio Alfredo II.
Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.519

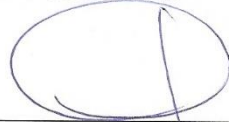
Wesley Belarmino da Silva

**PROTÓTIPO DE UM APLICATIVO PARA GESTÃO MUNICIPAL DE
OCORRÊNCIAS DE ORDEM PÚBLICA**

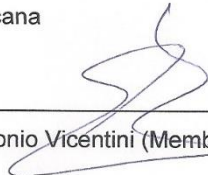
Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Tecnologia da Informação.

Americana, 27 de junho de 2017.

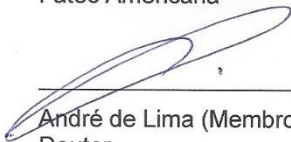
Banca Examinadora:



Antônio Alfredo Lacerda (Presidente)
Especialista
Fatec Americana



Eduardo Antonio Vicentini (Membro)
Mestre
Fatec Americana



André de Lima (Membro)
Doutor
Fatec Americana

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado forças, saúde e entendimento para superar todas os obstáculos e dificuldades, segundo aos meus pais Marcio e Cláudia pelo grande apoio, incentivo e por acreditarem em mim.

Ao meu orientador Prof. Antônio Alfredo Lacerda pelo empenho e dedicação à elaboração deste trabalho com as correções, dicas e novas ideias. Ao Prof. Wagner Siqueira que me ajudou muito no decorrer do curso. Aos meus colegas de classe e especialmente ao meu amigo Eliezer que me ajudou muito e foi um grande incentivador e motivador.

DEDICATÓRIA

Dedico este trabalho aos meus pais pelo apoio e paciência, aos meus colegas que aqui fiz e que eternamente estarão em minhas lembranças.

RESUMO

Atualmente a popularização dos dispositivos móveis vem crescendo em um ritmo bem acelerado. Cerca de 3 bilhões de pessoas possuem dispositivos móveis sejam eles smartphones, tablets ou qualquer outro dispositivo. Muitos desses dispositivos possuem uma infinidade de aplicativos e na maioria dos casos são indispensáveis para a humanidade. Grande parte da população deseja e opta por usufruir de aplicativos que lhe tragam maior facilidade, flexibilidade e comodidade. As pesquisas realizadas e detalhadas neste trabalho são referentes a elaboração de um Protótipo de aplicativo para Gestão municipal de ocorrências de ordem pública. O objetivo é apoiar a Gestão Pública de uma determinada cidade e conseqüentemente melhorar a qualidade de vida dos cidadãos pois os atendimentos e serviços prestados pelas prefeituras são, na maioria dos casos, ineficientes e burocráticos, desse modo, os cidadãos tornar-se-ão verdadeiros protagonistas das melhorias necessárias no meio público. Para tal feito e elaboração deste protótipo utilizou-se dos diagramas da Linguagem UML, um modelo conceitual do banco de dados e da ferramenta Android Studio um ambiente de desenvolvimento integrado bem conhecida no mercado. Após a conclusão futura do desenvolvimento deste trabalho os cidadãos poderão através deste aplicativo realizar suas ocorrências de uma forma mais rápida, clara e objetiva podendo assim colaborar na gestão da cidade.

Palavras Chave: Android. Aplicativo. Dispositivos móveis. Protótipo. Ocorrências.

ABSTRACT

Currently a popularization of mobile devices has been growing at a fast pace. About 3 billion people with smartphone devices, tablets or any other device. Often the devices are a multitude of applications and many of the cases are indispensable for a humanity. Great part of the population and chooses to take advantage of applications that greater ease, flexibility and convenience. As research carried out and detailed in this work are related to the elaboration of an application prototype for municipal management of occurrences of public order. The objective is to support a public management of a given city and consequently to improve the quality of life of the citizens for the services and services rendered to the municipalities are, in most cases, inefficient and bureaucratic, so the countries will become, Not real protagonists of improvements not in public. For this, the elaboration of this prototype used the diagrams of the UML Language, a conceptual database model and the Android Studio tool an integrated development environment well known in the market. After a future conclusion of the development of this work, the results of this study make their occurrences in a fast, clear and objective way.

Words Keys: Android. App. Mobile devices. Prototype. Occurrences

SUMÁRIO

1.	INTRODUÇÃO.....	14
1.1.	Descrição do Problema.....	15
1.2.	Aplicativos Semelhantes	16
1.3.	Hipótese	17
1.4.	Objetivo	18
1.5.	Objetivo específico.....	18
1.6.	Justificativa.....	19
2.	TECNOLOGIAS UTILIZADAS	21
2.1.	Android.....	21
2.1.1.	Arquitetura Android.....	21
2.1.2.	IDE - Ambiente de Desenvolvimento integrado	23
2.2.	Definição da API	24
2.3.	Infraestrutura do <i>Webservice</i>	25
2.3.1.	REST	26
2.3.2.	SOAP	27
2.4.	Segurança	28
2.5.	Criptografia	28
2.6.	Autenticação	28
2.6.1.	Sessão	29
2.6.2.	OAuth 2.0.....	29
2.6.3.	HTTP Basic Auth.....	29

2.6.4.	Comparação dos Métodos Apresentados	30
3.	ENGENHARIA DE SOFTWARE	31
3.1.	Projeto (UML)	32
3.1.1.	Diagramas da (UML)	33
3.1.2.	Diagrama de Caso de Uso	33
3.1.3.	Diagrama de Classes.....	36
3.1.4.	Diagrama de Objetos:.....	37
3.1.5.	Diagrama de Sequência:	38
3.1.6.	Diagrama de Estado	39
3.1.7.	Diagrama de Atividades	40
4.	BANCO DE DADOS	41
4.1.	Sistema Gerenciador de Banco de Dados (SGBD).....	42
4.2.	SQL Server	43
4.3.	MySQL	44
4.4.	Diferenças de Segurança entre MySQL e SQL Server	45
4.5.	Modelo Entidade-Relacionamento (ER).....	46
5.	APRESENTAÇÃO DO PROTÓTIPO	48
5.1.	Tela de Login	49
5.2.	Tela de Menu	50
5.3.	Tela de Cadastro de ocorrências	51
5.4.	Tela de acompanhamento das ocorrências.	52

5.5.	Tela de compartilhamento das ocorrências	53
5.6.	Tela de Mapa de ocorrências.....	54
5.7.	Tela de Cadastro de usuário.....	55
5.8.	Tela de Recuperação de senha	56
6.	CONSIDERAÇÕES FINAIS.....	57
	REFERÊNCIAS.....	58
	APÊNDICE A - Código-Fonte do Protótipo AmCity	61

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ART	Android Runtime
GB	Gigabyte
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
JVM	Java Virtual Machine
MB	Megabyte
PDA	Personal Digital Assistant
REST	Representational State Transfer
SD	Secure Digital Card
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
TLS	Transport Layer Security
UML	Unified Modeling Language
URL	Universal Resource Locator
XML	Extensible Markup Language

LISTA DE FIGURAS

Figura 1 - Aplicativo TakeVista.....	16
Figura 2 - Aplicativo Vigilante	17
Figura 3 - Arquitetura Android	22
Figura 4 - Tela Inicial do Android Studio	24
Figura 5 - <i>WebService</i>	26
Figura 6 - Diagrama de caso de Uso.....	33
Figura 7 - Diagrama de Classes.....	37
Figura 8 - Diagrama de Objetos	38
Figura 9 - Diagrama de Sequencia.....	39
Figura 10 - Diagrama de Estado	40
Figura 11 - Diagrama de Atividades	41
Figura 12 - SQL Server Management Studio	44
Figura 13 - MySQL WorkBench.....	45
Figura 14 - Exemplo de Modelo Conceitual	46
Figura 15 - Modelo Conceitual	47
Figura 16 - Tela de <i>login</i> de aplicativos famosos	48
Figura 17 - Tela de Login AmCity.....	49
Figura 18 - Tela de Menu	50
Figura 19 - Cadastro de ocorrências	51
Figura 20 - Tela de Acompanhamento	52
Figura 21 - Tela de compartilhamento das ocorrências	53
Figura 22 - Tela das ocorrências recentes	54
Figura 23 - Tela de cadastro de usuário.....	55
Figura 24 - Tela de recuperar senha	56

1. INTRODUÇÃO

A popularização dos dispositivos móveis, smartphones, tables, Ipad, Iphones, GPS, notebooks entre outros dispositivos tem apresentado uma evolução tecnológica notável. Atualmente os smartphones se tornaram objetos indispensáveis para a humanidade devido as suas inúmeras funcionalidades. Cerca de 3 bilhões de pessoas possuem um aparelho celular e com o passar dos anos os dispositivos móveis estão e estarão cada vez mais integrados e hoje a maioria dos usuários buscam celulares que possam ser utilizados como: câmera fotográfica, agenda eletrônica, GPS, monitor cardíaco, TV digital, controle remoto, entre outras finalidades e recursos, segundo Lecheta (2010).

“O Brasil conta hoje com mais de 168 milhões de smartphones ativos, segundo pesquisa realizada pela Fundação Getúlio Vargas (FGV). É quase impossível imaginar a vida desses usuários sem a utilização de aplicativos e a demanda por novidades nesse mercado está em alta. De acordo com a consultoria AppAnnie, que analisa o segmento comercial de aplicativos, o Brasil é um dos países com maior potencial de crescimento no ramo para os próximos cinco anos e deve apresentar aumento de 40% na receita em 2016.” (JORNAL USP, 2016).

Conforme o Dicionário Online de Português DICIO (2017), a palavra aplicativo pode ser definida da seguinte forma, “Tipo de programa de computador desenvolvido para processar dados de modo eletrônico, de forma a facilitar e reduzir o tempo do usuário ao executar uma tarefa”. Nos dias de hoje a maioria da população tem ou depende de um dispositivo móvel seja ele um: smartphone, tablet, padfone ou até mesmo um notebook híbrido. Grande parte dos dispositivos móveis contém uma variedade enorme de aplicativos.

De acordo com esses fatos a proposta deste trabalho será a elaboração de um protótipo de aplicativo para o gerenciamento das ocorrências de ordem pública para os municípios do interior de São Paulo. A modelagem e desenvolvimento deste protótipo utilizará os diagramas Use Case, Sequence Diagram e Class Diagram da Linguagem UML, a função dos diagramas é mostrar desde a forma com que o usuário interage com o aplicativo e também a sequência dos eventos ocorridos. Os diagramas também auxiliam no momento de desenvolvimento e implementação do

aplicativo, permitindo ao desenvolvedor uma visão mais clara e estruturada. Este trabalho está organizado da seguinte maneira. No Capítulo 2 é apresentado as Tecnologias utilizadas no projeto. O capítulo 3 expõe sobre a Engenharia de software e os diagramas da UML. O capítulo 4 mostra uma breve introdução do Banco de dados e o Modelo conceitual. Já o capítulo 5 apresenta as telas do protótipo do projeto. O capítulo 6 finaliza o trabalho com as Considerações finais. E por fim o capítulo 7 traz as Referências do trabalho.

1.1. Descrição do Problema

O estado de São Paulo possui uma quantidade de municípios relativamente grande, segundo o Instituto Brasileiro de Geografia e Estatística - IBGE, o estado conta com mais de 640 municípios e a maioria dessas cidades tem sérios problemas no que diz respeito à infraestrutura urbana. Boa parte dos problemas nessas localidades, sejam eles problemas na rede de água e esgoto ou iluminação pública, deve-se a uma gestão inadequada, corrupção e baixo orçamento, dentre outros fatores.

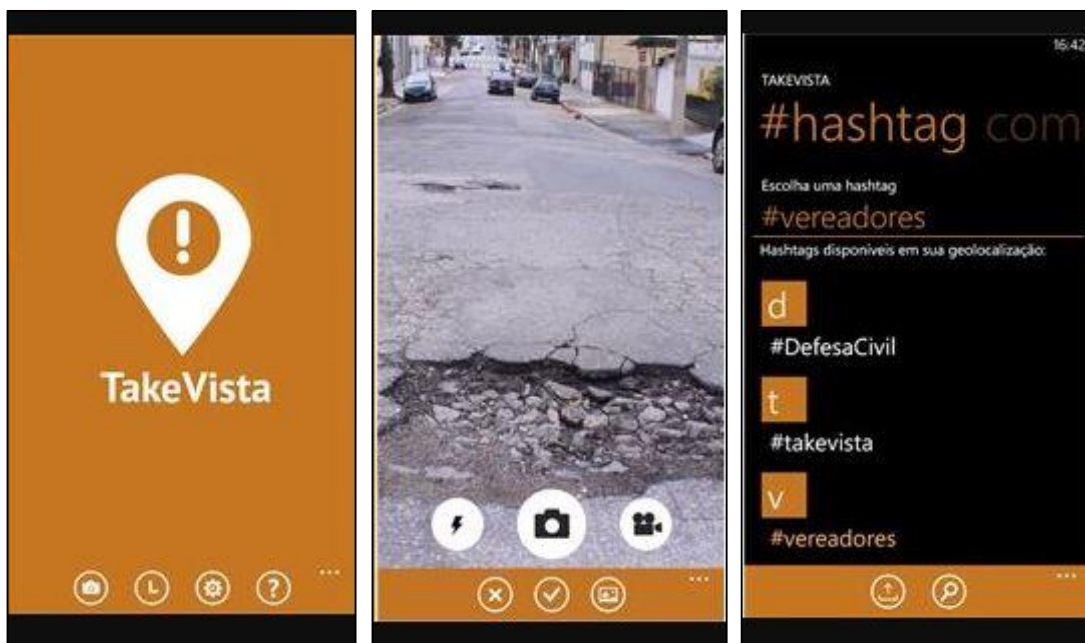
Devido à uma grande quantidade de problemas, tanto nas grandes metrópoles quanto nos municípios do interior de São Paulo, as prefeituras e os órgãos responsáveis não conseguem atender e gerenciar todas as ocorrências enviadas pelos meios tradicionais de denúncias. Outro fator relevante é a morosidade e a burocracia com que o poder público possui a fim de solucionar os problemas. A falta de regiões administrativas (subprefeituras), acabam tornando os Sistemas de Atendimento ao Cidadão (SAC) ainda mais difíceis para os munícipes da cidade, devido ser um processo muito centralizado.

A proposta deste projeto é levantar o problema e posteriormente enviar os dados via smartphone ou tablet aos órgãos competentes, tornando assim o processo altamente dinâmico. Cabe também à prefeitura obter um sistema web para que os dados cadastrados e as informações geradas pelos dispositivos móveis possam ser armazenadas de acordo com cada categoria, como por exemplo: Saneamento básico, Iluminação pública, transporte, saúde, conservação, entre outras categorias, sendo assim o aplicativo ficará encarregado de direcionar todas as ocorrências.

1.2. Aplicativos Semelhantes

A loja de aplicativos da Google *Play Store*, contém alguns aplicativos semelhantes ao que será proposto neste trabalho. Dentre eles podemos citar o aplicativo TakeVista, que permite denunciar problemas de uma determinada cidade, recebendo e enviando informações como as reclamações dos usuários para os órgãos públicos responsáveis. Entre um dos recursos o TakeVista possui um sistema de Hashtags que permite o usuário criar categorias das denúncias e permitir que elas sejam listadas a partir do acesso a hashtags. Na figura 1 apresenta o TakeVista.

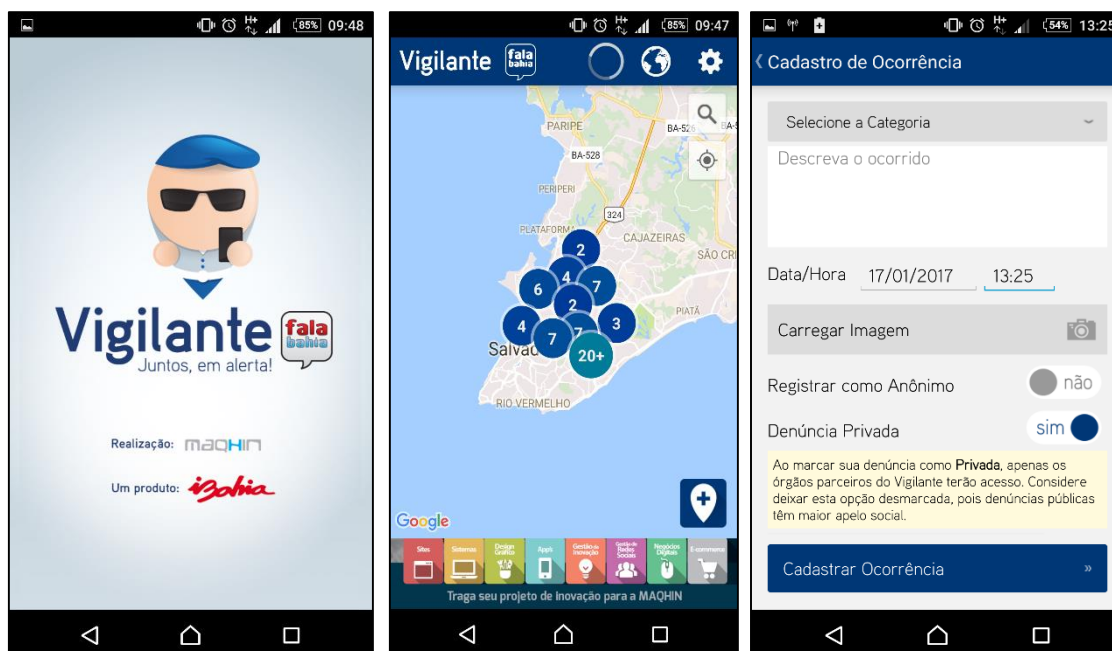
Figura 1 - Aplicativo TakeVista



Fonte: Autoria Própria

Outro aplicativo com objetivos e aplicações semelhantes é o Vigilante, onde através de uma determinada rede de usuários os munícipes podem tanto registrar uma nova ocorrência em prol da melhoria da cidade como até mesmo monitorar as que já estiverem em andamento. Na figura 2 apresenta o aplicativo Vigilante.

Figura 2 - Aplicativo Vigilante



Fonte: Autoria Própria

1.3. Hipótese

Suponha-se que uma pessoa está caminhando em uma determinada rua ou avenida e identifica um problema com mobilidade, iluminação pública ou até mesmo transporte público. Ao se deparar com essa situação o cidadão teria que ligar para a ouvidoria do município, ou até mesmo ir pessoalmente na região administrativa do bairro para tentar abrir um chamado para a solução do problema em questão. Com esta proposta do aplicativo de gestão de ocorrências o cidadão poderia simplesmente tirar fotos ou gravar vídeos com seu dispositivo móvel e automaticamente essa imagem/vídeo seria enviada ao órgão responsável. Sendo assim os cidadãos podem exercer sua função da cidadania de forma colaborativa em prol da cidade.

1.4. Objetivo

Este trabalho tem a finalidade de apoiar a Gestão Pública de uma determinada cidade, pois os atendimentos e serviços prestados pelas prefeituras são, na maioria dos casos, ineficientes e burocráticos, desse modo, os cidadãos tornar-se-ão verdadeiros protagonistas das melhorias necessárias no meio público. Desenvolvendo, através de estudos de *softwares*, *tecnologias* e diagramas, um protótipo de um aplicativo para melhorar o gerenciamento das denúncias de ocorrências feitas pelos moradores da cidade, como por exemplo: problemas de mobilidade, desordem urbana, criminalidade, meio ambiente, saneamento básico, focos de dengue ou até mesmo falhas na rede de fornecimento de água e energia. O aplicativo possibilitará ao cidadão registrar quaisquer problemas ocorridos dentro de tal cidade. Também poderão acompanhar, classificar e compartilhar suas denúncias através do aplicativo. Os munícipes podem realizar suas denúncias de forma plena e objetiva, não havendo a necessidade de se deslocar até o órgão competente ou até mesmo fazendo uso dos sistemas de atendimento computadorizado (SAC).

1.5. Objetivo específico

O objetivo deste trabalho está voltado a desenvolver um Protótipo de um aplicativo para dispositivos móveis que utilizará como base a plataforma Android. O desenvolvimento deste aplicativo fornecerá aos usuários as seguintes funcionalidades:

- Cadastrar e recuperar informações de Usuário;
- Cadastrar, atualizar e remover nova ocorrência;
- Acompanhamento e status das ocorrências;
- Classificação das Ocorrências, como nível de satisfação;
- Compartilhamento das ocorrências através das redes sociais;
- Opção de ocorrências por categoria;
- Obter Localização atual por meio de geocodificação;

O servidor a ser desenvolvido pela prefeitura da cidade será encarregado pelos seguintes serviços:

- Status das ocorrências, exemplo: se ocorrência foi ou não atendida;
- Manter dados dos usuários (CRUD) exemplo: Nome, sobrenome, cpf, cep, telefone, etc;
- Alteração e remoção das ocorrências já criadas, exemplo: alterar endereços, datas, ou até mesmo adicionar mais fotos na mesma ocorrência;
- Manter ocorrências (CRUD) através de textos, imagens ou vídeo;
- Armazenar mapa da cidade off-line para que usuário opte em baixar;
- Sincronizar o aplicativo com cada departamento dentro da Prefeitura;

1.6. Justificativa

A escolha de propor um protótipo de aplicativo para gestão pública de ocorrências se deu pelo fato de que a busca por novos métodos para se resolver problemas a partir dos dispositivos móveis vem crescendo cada dia mais.

Falando ainda sobre o aumento significativo da busca por dispositivos móveis como o smartphone:

O mercado de celulares está crescendo cada vez mais. Estudos mostram que hoje em dia mais de 3 bilhões de pessoas possuem um aparelho celular, e isso corresponde a mais ou menos metade da população mundial. Hoje em dia, os usuários comuns estão procurando cada vez mais celulares com diversos recursos como câmeras, músicas, Bluetooth, ótima interface visual, jogos, GPS, acesso à internet e e-mails, e agora ainda temos a TV digital. O mercado corporativo também está crescendo muito, e diversas empresas estão buscando incorporar aplicações móveis a seu dia-a-dia para agilizar seus negócios e integrar as aplicações móveis com seus sistemas de back-end. Empresas obviamente visam lucro e mais lucro, e os celulares e smartphones podem ocupar um importante espaço em um mundo onde a palavra mobilidade está cada vez mais conhecida. (LECHETA, 2010, p. 19).

Na área de mobilidade urbana, a maioria das ocorrências refere-se a buracos no asfalto, mas em muitas localidades as pessoas já se acostumaram com a buraqueira e nem reclamam mais, pois a burocracia para se resolver qualquer problema público é muito cansativa e demorada.

A maior parte do país está sofrendo com a crise hídrica e muitas das ocorrências se refere a vazamentos de água, saneamento básico entre outros problemas e esses problemas só chegam as autoridades depois de longas horas.

Aumentar a quantidade de funcionários em uma determinada prefeitura ajudaria a amenizar alguns dos problemas, porém não é a única solução.

Atualmente a Prefeitura da cidade de Americana localiza no interior de São Paulo, ainda não possui um aplicativo onde os cidadãos possam relatar os problemas urbanos. O aplicativo em questão irá otimizar os sistemas de atendimento (SAC) da prefeitura utilizando funcionalidades como localização da denúncia através do GPS enviando imagens ou vídeos do local relatado, conseqüentemente o aplicativo aumentará a confiabilidade dos serviços prestados pela prefeitura e ajudará melhorar a qualidade de vida dos cidadãos.

Além disso, se tornará um atrativo para o cidadão pela facilidade em solicitar e acompanhar os atendimentos solicitados. Com o desenvolvimento deste aplicativo os cidadãos podem exercer sua cidadania de forma colaborativa melhorando assim a qualidade de sua própria cidade, provendo uma melhor administração urbana como: infraestrutura, segurança, saúde, acessibilidade, transporte, etc.

2. TECNOLOGIAS UTILIZADAS

Neste capítulo serão apresentadas as ferramentas que serão utilizadas no decorrer do projeto, bem como a plataforma Android, sua arquitetura e a IDE - ferramenta de desenvolvimento para aplicativos. Apresentará também o uso de *webservices* e os aspectos importantes como autenticações, segurança e protocolos de comunicação entre o aplicativo e o servidor que será de responsabilidade da prefeitura.

2.1. Android

Segundo Lecheta (2010), o Android é uma plataforma que serve para desenvolvimento de aplicativos móveis como tablets e smartphones, a plataforma android conta com um sistema operacional baseado em Linux, uma interface visual rica com diversas aplicações já instaladas e ainda um ambiente de desenvolvimento muito eficiente e adaptável.

2.1.1. Arquitetura Android

A arquitetura do sistema operacional Android, como mostra a Figura 4, é dividida em camadas, aonde cada parte é responsável por gerenciar os seus respectivos processos. (LECHETA, 2010).

Figura 3 - Arquitetura Android



Fonte: (TISELVAGEM, 2012)

As cinco camadas demonstradas na Figura 3 são:

Linux Kernel: O Google usou a versão 2.6 do Linux para construir o *kernel* do Android, o que inclui os programas de gerenciamento de memória, as configurações de segurança, o software de gerenciamento de energia e vários *drivers* de hardware (STRICKLAND, 2009).

Bibliotecas: Nesta camada contém as bibliotecas básicas que são usadas pelo Android, junto a elas a *OpenGl/ES* para trabalhar com gráficos e a SQLite que permite trabalhar com banco de dados. Nessa mesma camada também pode-se encontrar a Dalvik, que é uma JVM (*Java Virtual Machine*) para rodar o conteúdo Java. Grande parte destas bibliotecas foi desenvolvida em C e C++ (PRADO, 2011).

Runtime do Android: Runtime é uma junção de várias bibliotecas centralizadas que permite o desenvolvimento de aplicativos para Android, usando linguagem Java, contendo também uma máquina virtual chamada Dalvik para que cada aplicativo rode em seu próprio processo, com sua própria instância, esta máquina foi projetada especialmente para o Android (LEE, 2011).

Frameworks: Desenvolvida basicamente por completo em Java, esta camada é responsável pela interface com as aplicações Android. Ela provê um conjunto de bibliotecas para acessar os diversos recursos do dispositivo como interface gráfica, telefonia, serviço de localização (GPS), banco de dados persistente, armazenamento no cartão SD, 2D etc. Fornece blocos de alto nível de construção utilizados para criação de aplicações. O framework vem pré-instalado com o Android (MOBILEIN, 2010).

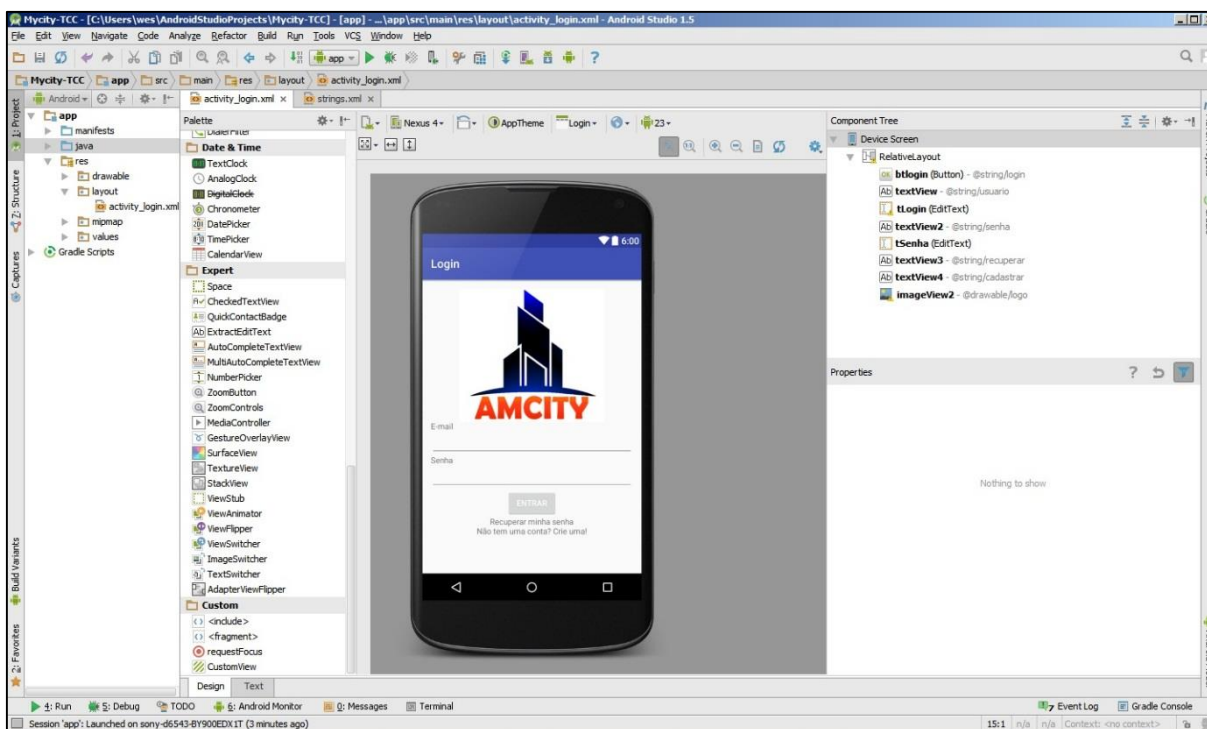
Aplicações: Nesta camada que ficam as aplicações (desenvolvidas em Java) para o Android. É um dos grandes segredos do sucesso da plataforma, já que possui mais de 250.000 aplicações no Android Market, e continua crescendo cada dia que passa (PRADO, 2011).

2.1.2. IDE - Ambiente de Desenvolvimento integrado

Um ambiente de desenvolvimento integrado (*Integrated Development Environment - IDE*), ou seja, agrupa várias ferramentas que servem para auxiliar no desenvolvimento de um *software/aplicativo*.

Existem algumas opções disponíveis para a linguagem JAVA, dentre as mais utilizadas estão o Eclipse, Android Studio que integra o SDK no IntelliJ IDEA e o NetBeans, no entanto apenas uma IDE para JAVA não é o suficiente para o desenvolvimento de aplicativos para plataforma Android, pois para desenvolver aplicativos é necessário um conjunto específico de ferramentas. Atualmente o conjunto de ferramentas disponível mais maduro e estável é o Android SDK (kit de desenvolvimento do Android da empresa Google. Com isso a IDE mais recomendada para o uso do kit SDK é o Android Studio, portanto o mesmo será utilizado para o desenvolvimento deste projeto. Os aplicativos são escritos usando a linguagem de programação Java e executados na ART e Dalvik, máquinas virtuais personalizadas e projetadas para rodar dentro dos dispositivos Android que funcionam em cima de um kernel Linux (CORDEIRO, 2017).

Figura 4 - Tela Inicial do Android Studio



Fonte: Autoria Própria

2.2. Definição da API

As informações contidas neste nível apresentarão as definições de API e qual seriam as suas utilizações no desenvolvimento deste projeto.

Medeiros (2014) afirma que “API significa *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos. Esta interface de programação é um conjunto de padrões de programação que permitem a construção de aplicativos e a sua utilização”.

Segundo Medeiros (2014), na grande maioria das vezes uma API é responsável por se comunicar com diversos outros códigos ou outras plataformas interligando diversas funções em um aplicativo ou site. Um exemplo de uma API bastante utilizada é a API do Facebook que tem como finalidade auxiliar a autenticação de usuários em websites e aplicativos. Atualmente existem vários tipos de API's que podem ser incorporadas tanto em websites quanto em aplicativos, dentre elas estão as API's do Google Maps que permite utilizar a maior parte das funcionalidades como: usar requisições para acessar informações de

geocodificações, rotas, elevações e lugares dos aplicativos clientes, segundo Douglas (2013).

A geocodificação é o que permite transformar dados de localização como coordenadas, endereços e nomes de estabelecimentos em uma geolocalização com latitude e longitude. Por meio dessa tecnologia, é possível encontrar endereços, localizações de clientes e rotas. (COGNATIS,2017).

O Google Maps API será de altíssima importância para o desenvolvimento deste projeto pois ela permite criar aplicações com a capacidade de recuperar informações de latitude e longitude de um determinado endereço quando solicitado por um usuário.

Baseando-se nessas informações o protótipo que será apresentado, deverá ser implementado a API do Google Maps, para que no momento em que o munícipe for cadastrar uma nova ocorrência o aplicativo através das coordenadas possa automaticamente marcar o local exato da mesma.

2.3. Infraestrutura do *Webservice*

As informações contidas neste capítulo serão destinadas a apresentação da estrutura, definição e aspectos importantes de autenticação, segurança e protocolos de comunicação de um *webservice*. Segundo Menéndez (2002) existe uma definição muito simples para *webservices*.

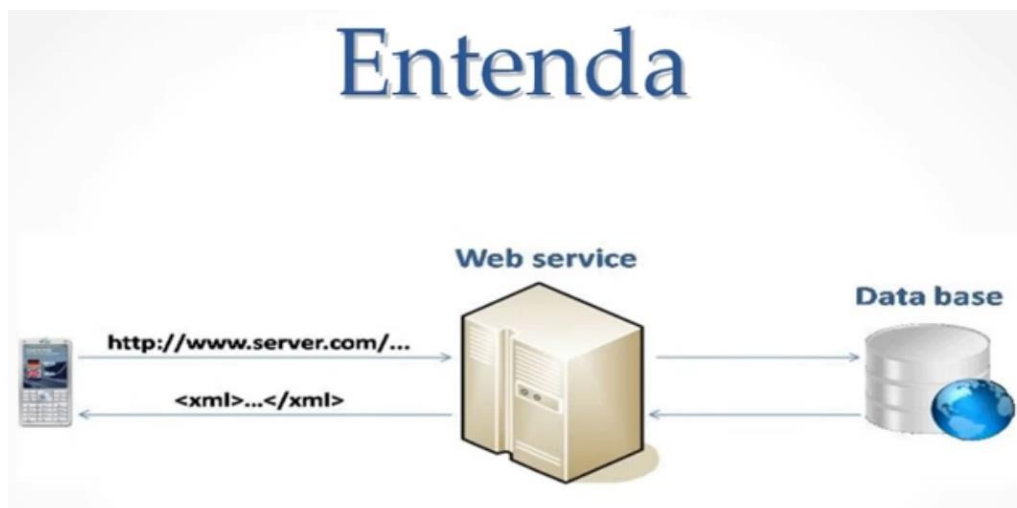
É uma aplicação que aceita solicitações de outros sistemas através da Internet, sempre baseado em padrões para facilitar a comunicação entre as máquinas. Esses padrões são baseados em XML (eXtensible Markup Language). (MENÉNDEZ, 2002).

As construções de *webservices* são criadas a partir de “*servlets*”, ou seja, classes Java que servem para expandir a funcionalidade de um servidor. *Servlets* vêm do inglês e podemos chamar de um servidor pequeno cujo principal objetivo é realizar a comunicação entre uma aplicação e um banco de dados, recebendo e enviando requisições padronizadas utilizando protocolos de comunicação HTTP para processá-las e responder ao cliente através de arquivos em formato XML (ZEIGER, 2008).

Para que um aplicativo possa obter dados de um servidor, é necessário que haja uma infraestrutura preparada para atender às requisições do app. Seria possível desenvolver um protocolo e utilizá-lo na disponibilização dessas informações, entretanto, utilizar tecnologias padronizadas e altamente difundidas permite que diversas aplicações possam interagir com o serviço,

bem como tornar o sistema mais confiável e seguro, graças à utilização de ferramentas desenvolvidas e testadas em ambientes de produção. Uma das formas mais comuns para a disponibilização de informações a serem acessadas por outros programas através da Internet é com o uso de *webservices*. (MEIRA, 2014, p.28).

Figura 5 - WebService



Fonte: Autoria Própria

2.3.1. REST

Toda a comunicação entre cliente e servidor se dará por meio do Hypertext Transfer Protocol (HTTP) atualmente o estilo de arquitetura mais utilizado para desenvolvimento de *webservices* é o REST (*Representational State Transfer*), justamente por trabalhar diretamente sobre o protocolo *hipermídia Hypertext Transfer Protocol* (HTTP) e por se tratar de um padrão mais novo, tem-se mostrado melhor em alguns testes de performance. REST é um estilo de arquitetura para a criação de *webservices* que vem sendo largamente adotada atualmente. Embora não seja especificado como um protocolo, em geral os serviços RESTful são criados sobre o HTTP, mapeando seus métodos para as operações a serem realizadas sobre recursos, segundo Fielding (2000).

A proposta de Fielding (2000), mostra que as aplicações que aplicam os princípios do REST são conhecidas como RestFull, o REST possui três características básicas que são:

- **Recursos:** São elementos de informação, que são identificados na requisição e podem usados na URI. O cliente e o servidor se comunicam através de uma interface HTTP para poder manipular esse recurso.

- **Ações:** O protocolo HTTP possui oito métodos, porém os mais utilizados são:

GET: Lista membros de uma coleção ou obtém informação sobre um membro específico da coleção.

PUT: Substitui uma coleção ou membro por outro.

POST: Cria um novo membro em uma coleção.

DELETE: Deleta uma coleção ou um membro dela.

- **Conteúdo:** Os mime-types são utilizados por alguns protocolos para identificar o que está sendo trafegado, a identificação é feita no header na requisição no campo Content-Type, os tipos mais comuns encontrados são: Json e Xml.

No aplicativo ocorrerão listagens, criações, exclusões e modificações de informações, dessa forma, neste trabalho serão utilizados os métodos mais utilizados, que são: GET, PUT, POST e DELETE. Além disso, seguindo as recomendações do REST, o protocolo deve ser totalmente stateless, ou seja, cada requisição contém toda a informação necessária para ser processada e não depende de outras requisições.

2.3.2. SOAP

Dantas (2007) afirma que, “SOAP é um protocolo baseado em XML para troca de informações em um ambiente distribuído. É utilizado para troca de mensagens entre aplicativos distribuídos pela rede”.

Podemos considerar também que “O protocolo de acesso simples a objetos (Simple Object Access Protocol-SOAP) é um protocolo de comunicação baseado em XML que permite o transporte de informações sobre HTTP”. (KUHN, 2012).

Estes aplicativos, ou “*webservices*”, possuem uma interface de acesso simples e bem definida. Os *webservices* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria “linguagem”, que é traduzida para uma linguagem universal, o formato XML. (DANTAS,2007).

Sendo assim o protocolo SOAP é projetado para chamar aplicações remotas por meio de trocas de mensagens em ambientes independentes de plataformas e linguagens de programação. Assim como o REST, o SOAP é um padrão normalmente aceito para utilizar-se com *webservices* e um elemento principal tanto da infraestrutura quanto no desenvolvimento de *webservices*.

2.4. Segurança

No que diz respeito à segurança os *webservices* referem-se a pontos importantes. Primeiramente e de altíssima importância é que todos os dados que foram transmitidos por algum usuário devem ser totalmente confidenciais, de forma que somente o usuário proprietário possa acessar seu conteúdo. Isto é, somente será permitido através do uso de criptografia onde o usuário deverá autenticar-se ao servidor utilizando suas informações de acesso, (MEIRA, 2014).

2.5. Criptografia

Toda a comunicação entre cliente e servidor será feita através do protocolo *Hypertext Transfer Protocol Secure*, ou HTTPS que terá a função de garantir maior segurança dos dados trafegados pela rede. O HTTPS não é somente um protocolo, ele é utilizado sobre um protocolo *Transport Layer Security* (TLS), segundo Ankit (2016).

O TLS trabalha com chaves assimétricas permitindo que um cliente e um servidor possam negociar uma chave simétrica. A chave simétrica tem a finalidade de criptografar toda a comunicação entre cliente e servidor e só é válida apenas para uma única sessão. Essa chave simétrica garante que nenhum dos dados enviados pelo cliente saia do servidor de maneira não segura e a função de segurança do webservice é recusar todas as conexões que não utilizem HTTPS, (SINHAL, 2016).

2.6. Autenticação

Em *webservices* existem três formas principais de autenticação. A seguir, as três formas serão apresentadas e cabe à prefeitura decidir qual melhor forma de autenticação utilizar no desenvolvimento.

2.6.1. Sessão

Através de uma requisição POST o usuário envia ao servidor suas credenciais para serem autenticadas, esta é a forma de autenticação mais utilizada, elas são geradas por meio de formulários de websites. Esta forma de autenticação permite o servidor verificar as credenciais informadas pelo usuário, se estiverem corretas o servidor inicia a sessão do usuário.

A sessão consiste em um conjunto de dados pertencentes ao usuário que são armazenados dentro de um servidor, esse conjunto de dados contém um indicador que é enviado ao usuário através de um cookie que fica localizado dentro de um determinado navegador. Esse procedimento pode ser encontrado em lojas virtuais ou sites que são acessados por meio de login, (VAMOSI, 2008).

2.6.2. OAuth 2.0

Segundo Ramsey (2016), A autenticação OAuth é a tecnologia que permite o usuário autenticar-se através das contas dos sites Google, Twitter ou Facebook, por exemplo. Utilizando a função OAuth o usuário decide junto ao serviço para que uma aplicação tenha acesso aos seus dados. Após o usuário permitir acesso aos seus dados, a aplicação recebe uma sequência de números, esta sequência é chamada de *token* com o qual consegue obter as informações desejadas do usuário. Dessa forma as informações do usuário nunca serão fornecidas a aplicações de terceiros. O OAuth é encarregado de gerenciar as autorizações concedidas pelos usuários, gerar e expirar os *tokens*. Atualmente existem implementações de servidores OAuth para várias linguagens como Python²⁷, Java²⁵, NodeJS²⁸ e PHP²⁶.

2.6.3. HTTP Basic Auth

Este é o tipo de sessão é usado por páginas em que o navegador exibe uma janela solicitando nome de usuário e senha para acessar, comuns em configuração de roteadores domésticos. No caso do acesso via navegador, quando as credenciais são informadas, o próprio navegador as armazena durante um tempo e as reenvia a cada requisição. Apesar da sensação de que foi criada uma sessão, na verdade cada requisição é autenticada individualmente.

No HTTP Basic Auth, as credenciais são enviadas no cabeçalho da requisição HTTP. Ao tentar acessar uma URL que necessita de autorização, o servidor responde com um código 401 Not Authorized. O cliente, então, deve gerar uma string no formato usuário: senha, codificá-la no formato na Base64 e enviá-la no cabeçalho *Authorization* da requisição, precedido da palavra *Basic*. Por exemplo, um usuário cujo *login* seja Joao e cuja senha seja senha123 deve enviar o seguinte cabeçalho: *Authorization: Basic Sm9hbzpzZW5oYTEyMw==*, (MEIRA, 2014, p.44).

Um detalhe importante deste mecanismo de autenticação é que as credenciais são enviadas com uma codificação diferente, mas não são criptografadas, de forma que qualquer um que tenha acesso ao cabeçalho da requisição poderá obter as credenciais desfazendo a codificação Base64.

2.6.4. Comparação dos Métodos Apresentados

Uma das autenticações mais utilizadas em websites e aplicativos é a autenticação por sessão, porém ela não segue as recomendações do REST de que cada requisição necessita ser stateless. Este tipo de autenticação desenvolve uma dependência de requisições pois, exige que haja uma requisição para o envio de credenciais e a obtenção de um cookie para somente depois disso, então, obter os dados do usuário.

Apesar de ser bastante utilizado pelos grandes *webservices* o OAuth, este tipo de autenticação ainda possui uma certa dificuldade na implantação, uma vez que é necessária a criação de um mini servidor OAuth. Mesmo sendo um método de eficiente por permitir que usuários nunca informe suas senhas para aplicativos de terceiros, a implementação deste tipo de autenticação está fora do escopo deste trabalho.

O HTTP Basic Auth é um tipo de autenticação mais dinâmica que pode trabalhar com cada requisição de maneira individual, podendo assim atender ao princípio de requisições stateless do REST. Apesar deste método enviar as credenciais sem nenhum tipo de criptografia não significa que há uma falha de segurança, uma vez que o servidor só responderá a requisições enviadas por meio de protocolos HTTPS, garantindo que todos os dados desde os cabeçalhos onde as credenciais são enviadas, sejam criptografadas.

3. ENGENHARIA DE SOFTWARE

A engenharia de *software* é uma disciplina ligada ao desenvolvimento e manutenção de um *software* visando qualidade, organização e produtividade, para atender as necessidades solicitadas pelos clientes.

Para que sejam atendidas essas necessidades o *software* deve possuir um bom tempo de resposta e ter como seu principal objetivo a usabilidade em seu alto nível para atender as expectativas do usuário, obtendo assim uma melhor qualidade de *software*.

Segundo a definição também de um dos maiores estudiosos da Engenharia de *Software*, Ian Sommerville.

A Engenharia de *Software* é uma disciplina de engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a sua manutenção desse sistema, depois que ele entrou em operação. (SOMMERVILLE, 2007, p. 5)

A proposta da Engenharia de *software* é conciliar organização ao projeto, portanto com esta organização, permite-se construir *softwares* de alta qualidade. Sommerville (2007, p. 5), diz que “[...] os engenheiros de *software* adotam uma abordagem sistemática e organizada em seu trabalho, que é, frequentemente, a maneira mais eficaz de produzir *software* de alta qualidade.

Na construção de sistemas de *software*, assim como na construção de sistemas habitacionais, também a uma graduação de complexidade. Para a construção de sistemas de *software* mais complexos, também é necessário planejamento inicial. (BEZERRA, 2007, p. 2).

Conforme a citação, Bezerra compara a engenharia civil com a engenharia de *software*, revelando as necessidades de planejamento do projeto para garantir uma conclusão com sucesso em projetos complexos.

3.1. Projeto (UML)

A UML é uma abreviatura que vem do inglês *Unified Modeling Language*. Pela definição de seu nome, a UML é uma linguagem que define uma série de módulos afim de documentar os sistemas orientados a objetos que desenvolvemos, segundo Ribeiro (2012).

Última impressão em Criado em O principal produto é um bom *software* capaz de satisfazer as necessidades de seus usuários e respectivos negócios”. (BOOCH, RUMBAUGH e JACOBSON, 2000, p.03).

A modelagem é uma das partes fundamentais das atividades que dão início a implantação de um bom *software*. Os modelos são utilizados nas mais diversas formas, tendo em vista a especificação e comunicação correta entre eles. “Os modelos ajudam a visualizar o sistema como ele é, ou como desejamos que seja. Os modelos permitem especificar a estrutura ou comportamento de um sistema.

Os modelos proporcionam um guia para a construção do *software*. Os modelos documentam as tomadas de decisões”. (BOOCH, RUMBAUGH e JACOBSON, 2000, p.06).

“A UML é uma linguagem padrão para a elaboração e construção de *software*. A linguagem pode ser usada para especificação, visualização, construção e documentação do sistema”. (BOOCH, RUMBAUGH e JACOBSON, 2000, p.13).

A linguagem UML se adapta para modelar sistemas web, sistemas de informações e até mesmo sistemas mais complexos em tempo real.

Para compreender e implementar a UML em um sistema é fundamental o entender sobre as três principais regras; os blocos básicos de construção da UML; as regras de como esses blocos serão interligados; e os mecanismos que se aplicam a toda a linguagem.

A linguagem UML é destinada a fatores muito importantes no momento de modelar um sistema. Como foi citado acima a linguagem permite especificar, visualizar e documentar todos os dados de desenvolvimento tanto de um *software* quanto de um aplicativo.

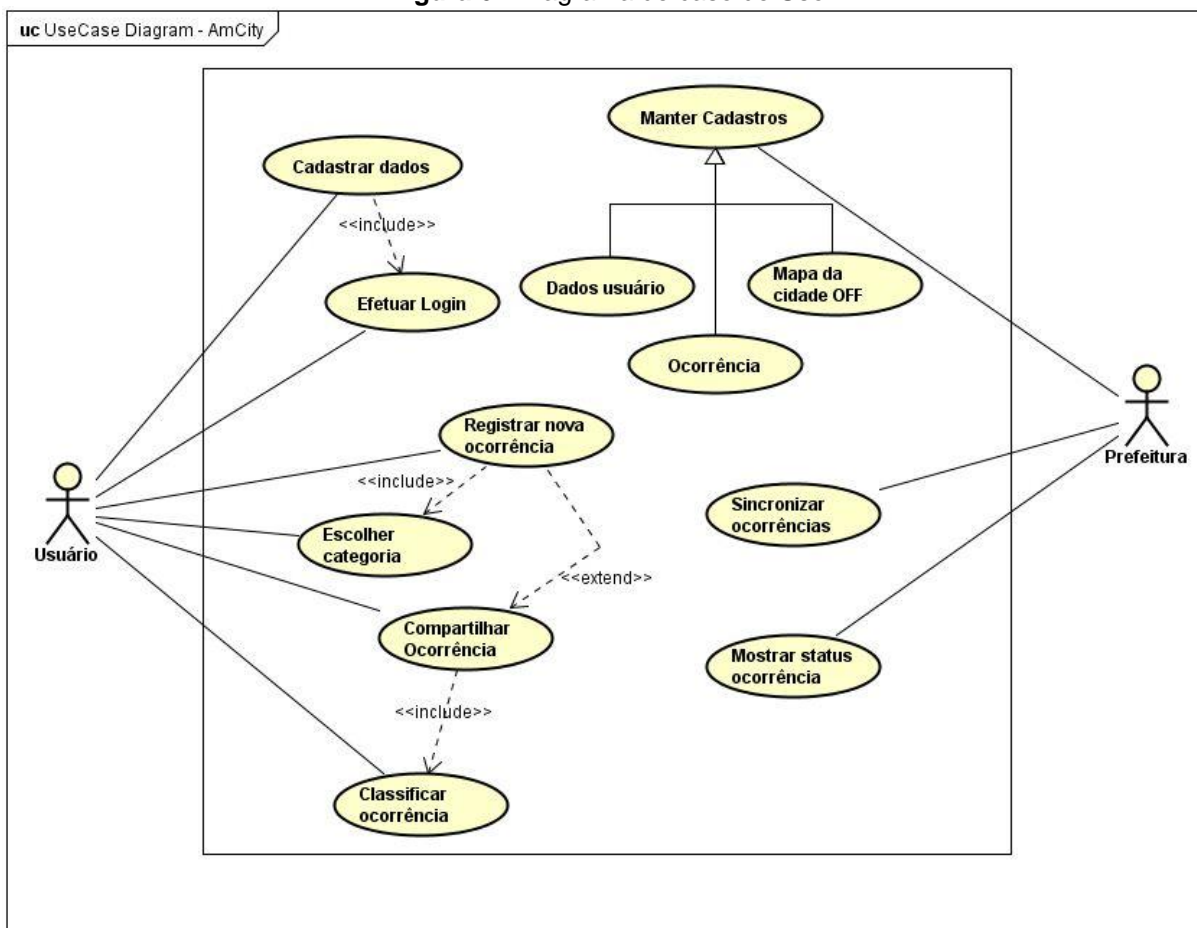
3.1.1. Diagramas da (UML)

Um diagrama UML é composto basicamente de elementos gráficos, ou seja, desenhos. Estes desenhos são feitos para uma melhor visualização dos diversos aspectos de um sistema. Segundo BOOCH, RUMBAUGH e JACOBSON (2000), a UML fornece os seguintes diagramas:

3.1.2. Diagrama de Caso de Uso

O diagrama de casos de uso usa um tipo especial de classe, os atores que representam os usuários. Estes diagramas são importantes porque apresentam como será a interação entre o usuário e o sistema.

Figura 6 - Diagrama de caso de Uso



Fonte: Autoria Própria

As informações detalhadas das interações dos atores com o sistema serão mostradas nos quadros a seguir.

Quadro 1 – Cadastrar dados.

Descrição	Fluxo	Pré condições	Pós condições
Cadastro do Usuário	<ol style="list-style-type: none"> 1. Usuário inicia o seu aplicativo móvel. 2. Aplicativo abre com a tela de <i>login</i>. 3. Usuário preenche informações de cadastro e aperta o botão “Cadastrar”. 4. Aplicativo valida informações e redireciona para a tela Principal. 	<i>Smartphone</i> deve possuir alguma conexão de rede.	Usuário cadastrado no sistema.

Fonte: Autoria própria de todos os quadros.

Quadro 2 – Efetuar login.

Descrição	Fluxo	Pré condições	Pós condições
Autenticação do Usuário	<ol style="list-style-type: none"> 1. Usuário inicia o seu aplicativo móvel. 2. Aplicativo abre com a tela de <i>login</i>. 3. Usuário preenche dados de usuário e senha e aperta o botão “Efetuar Login”. 4. Aplicativo valida o usuário e senha e redireciona para a tela Principal. 	<i>Smartphone</i> deve possuir alguma conexão de rede.	Usuário autenticado no sistema.

Quadro 3 – Escolher categoria.

Descrição	Fluxo	Pré condições	Pós condições
Categoria da Ocorrência escolhida pelo Usuário	<ol style="list-style-type: none"> 1. Usuário aperta botão “Nova Ocorrência”. 3. Usuário escolhe qual categoria deseja registrar a ocorrência. 4. Aplicativo redireciona para a tela de cadastrar ocorrência. 	<ol style="list-style-type: none"> 1. <i>Smartphone</i> deve possuir alguma conexão de rede. 2. Usuário deve estar autenticado no sistema. 	Categoria Selecionada.

Quadro 4 – Registrar ocorrência.

Descrição	Fluxo	Pré condições	Pós condições
Usuário Registra nova Ocorrência	<ol style="list-style-type: none"> 1. Usuário aperta botão “Nova Ocorrência”. 2. Usuário ativa opção acesso Local do Google. 3. Usuário escolhe qual opção de imagem/vídeo para registrar a ocorrência. 4. Aplicativo redireciona para a tela de ocorrências registradas. 	<ol style="list-style-type: none"> 1. Smartphone deve possuir alguma conexão de rede. 2. Usuário deve estar autenticado no sistema. 3. Categoria deve estar selecionada 4. Acesso Local 	Ocorrência Registrada.

Quadro 5 – Classificar ocorrência.

Descrição	Fluxo	Pré condições	Pós condições
Usuário Classifica a Ocorrência	<ol style="list-style-type: none"> 1. Usuário aperta botão “Classificar Ocorrência”. 3. Usuário escolhe a melhor classificação para a ocorrência. 4. Aplicativo redireciona para a tela de compartilhamento de ocorrências. 	<ol style="list-style-type: none"> 1. Smartphone deve possuir alguma conexão de rede. 2. Usuário deve estar autenticado no sistema. 3. Ocorrência deve estar finalizada. 	Ocorrência Classificada.

Quadro 6 – Compartilhar ocorrência.

Descrição	Fluxo	Pré condições	Pós condições
Usuário compartilha Ocorrência	<ol style="list-style-type: none"> 1. Usuário aperta botão “Compartilhar Ocorrência”. 3. Usuário escolhe qual rede social deseja compartilhar a ocorrência. 4. Aplicativo finaliza ocorrência. 	<ol style="list-style-type: none"> 1. Smartphone deve possuir alguma conexão de rede. 2. Usuário deve estar autenticado no sistema. 3. Ocorrência deve estar finalizada. 	Ocorrência Compartilhada.

Quadro 7 – Manter cadastros.

Descrição	Fluxo	Pré condições	Pós condições
-----------	-------	---------------	---------------

Prefeitura – Manter Cadastros	<ol style="list-style-type: none"> 1. Credenciais dos Usuários. 2. Informações das Ocorrências. 3. Mapa Off-Line da Cidade. 	<ol style="list-style-type: none"> 1. Prefeitura deve possuir Banco de Dados. 2. Possuir um SGBD. 3. Sistema de <i>Webservices</i> para interação com B.D. 	Informações dos usuários e mapa off-line da cidade mantidas.
-------------------------------	--	---	--

Quadro 8 – Status das ocorrências.

Descrição	Fluxo	Pré condições	Pós condições
Prefeitura Informar Status das Ocorrências	1. Servidor da Prefeitura “informa Status das Ocorrências”.	1. Prefeitura possuir tabelas com ocorrências cadastradas	Status Ocorrências

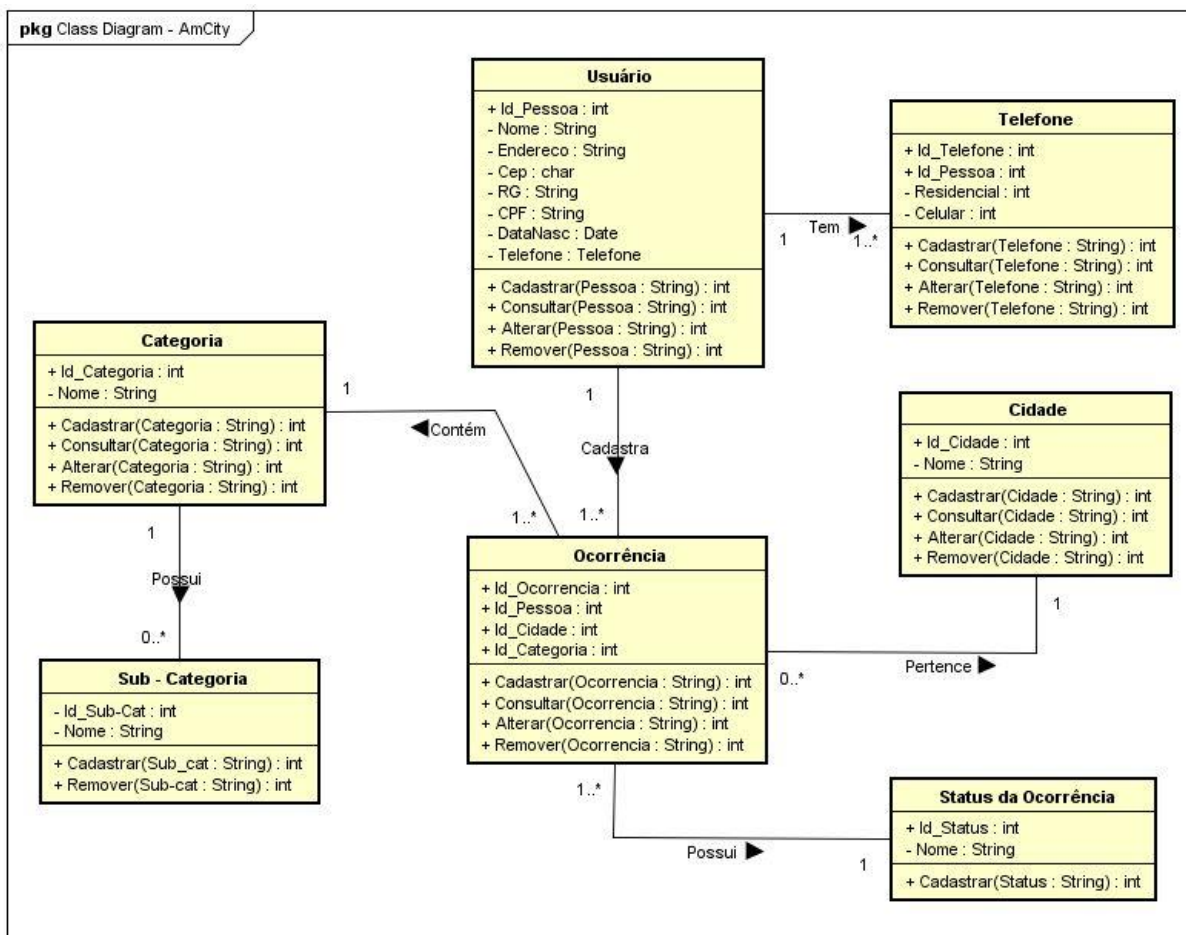
Quadro 9 – Sincronizar ocorrência.

Descrição	Fluxo	Pré condições	Pós condições
Prefeitura sincroniza secretarias responsáveis das Ocorrências	1. Servidor da Prefeitura encaminha ocorrências para as secretarias.	1. Prefeitura possuir tabelas com ocorrências cadastradas	Ocorrência encaminhada

3.1.3. Diagrama de Classes

Um diagrama de classes mostra um conjunto de classes. São geralmente usados e encontrados em modelagens orientadas a objetos, onde abrangem uma visão estática do sistema.

Figura 7 - Diagrama de Classes

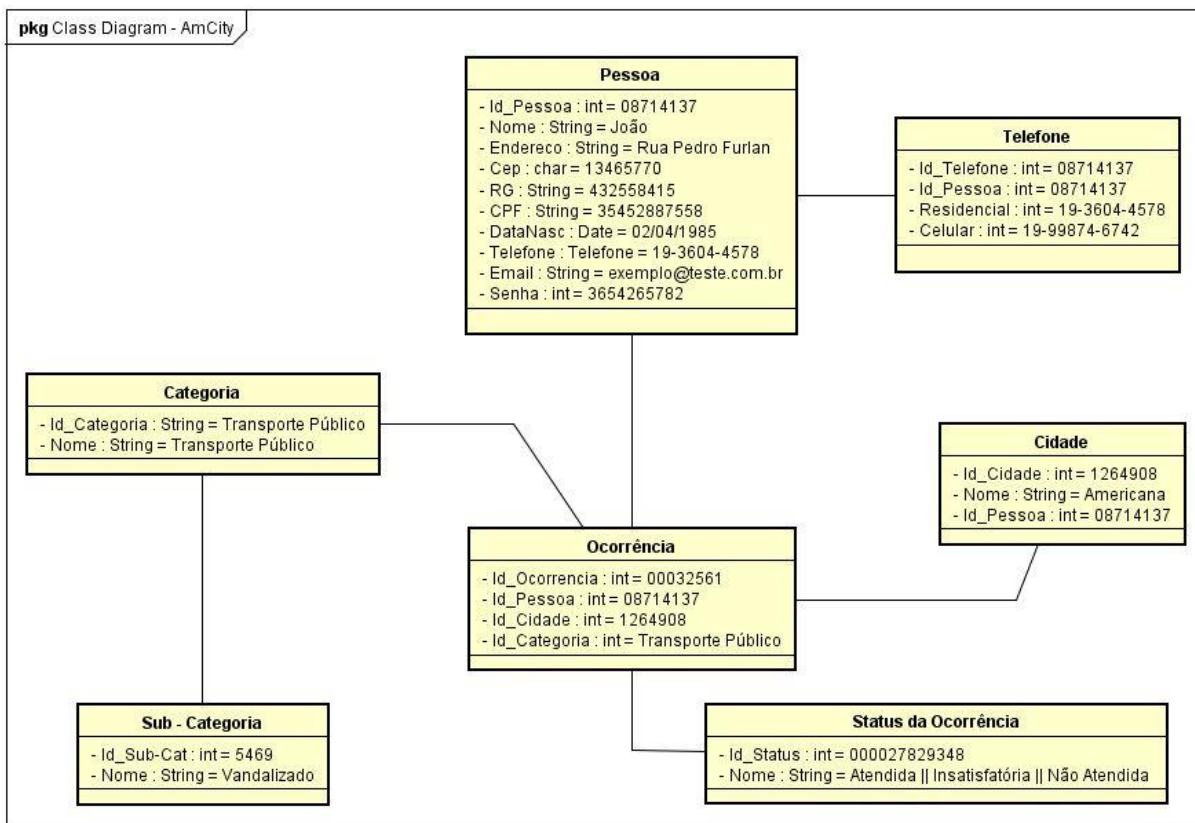


Fonte: Autoria Própria

3.1.4. Diagrama de Objetos:

Um diagrama de objetos mostra seus objetos e relacionamentos. Traz as instâncias que são encontradas no diagrama de classes, ou seja, as variáveis. Também mostram uma visão estática de um processo do sistema.

Figura 8 - Diagrama de Objetos

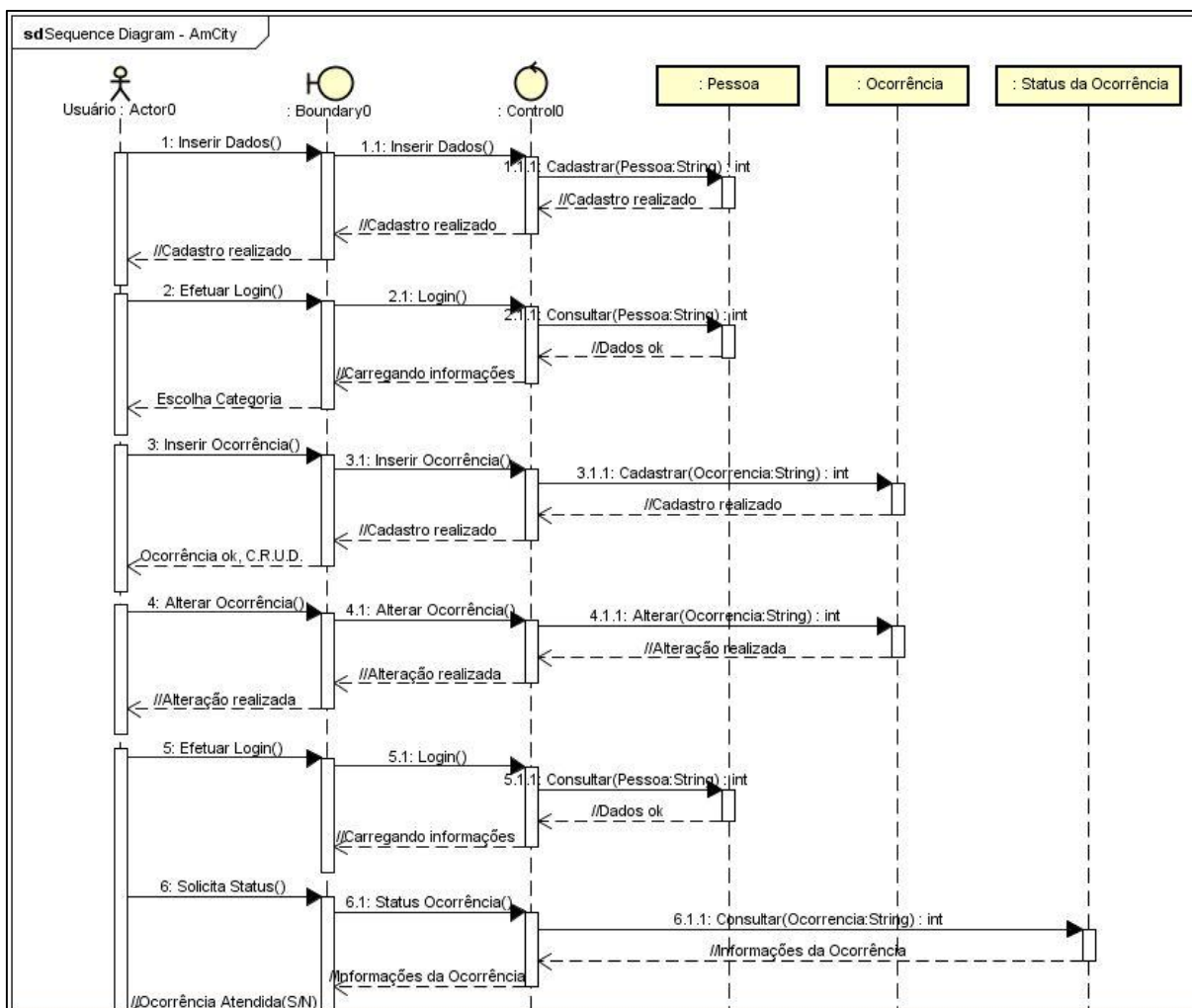


Fonte: Autoria Própria

3.1.5. Diagrama de Sequência:

Este diagrama pode ser visto como um diagrama de interação. Exibe interação dos objetos e seus respectivos relacionamentos. Além disso, inclui as mensagens trocadas entre os objetos. Este diagrama apresenta uma visão dinâmica do sistema.

Figura 9 - Diagrama de Sequência

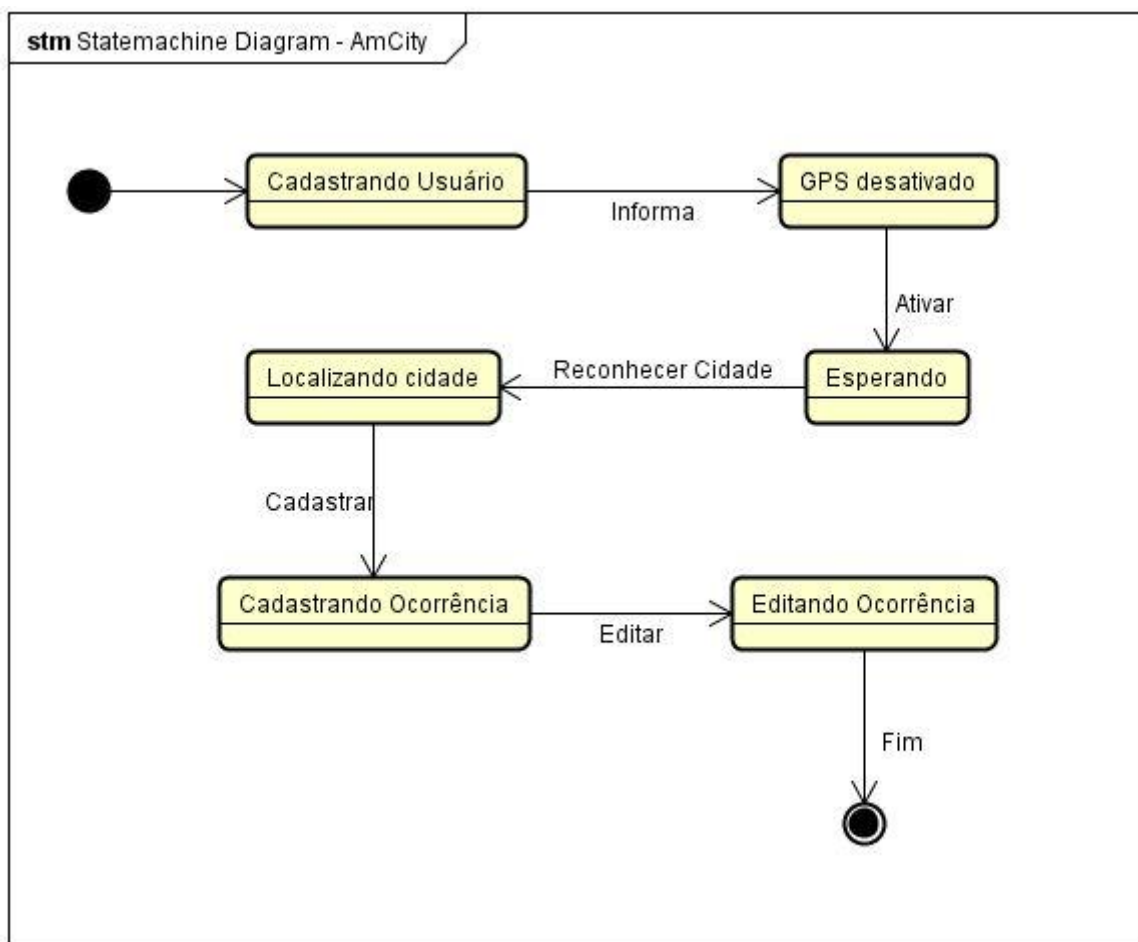


Fonte: Autoria Própria

3.1.6. Diagrama de Estado

Estes diagramas abrangem uma visão dinâmica de um sistema. Preocupa-se em exibir transições, eventos e atividades do sistema. São importantes porque analisam os comportamentos que acontecem nas interfaces.

Figura 10 - Diagrama de Estado

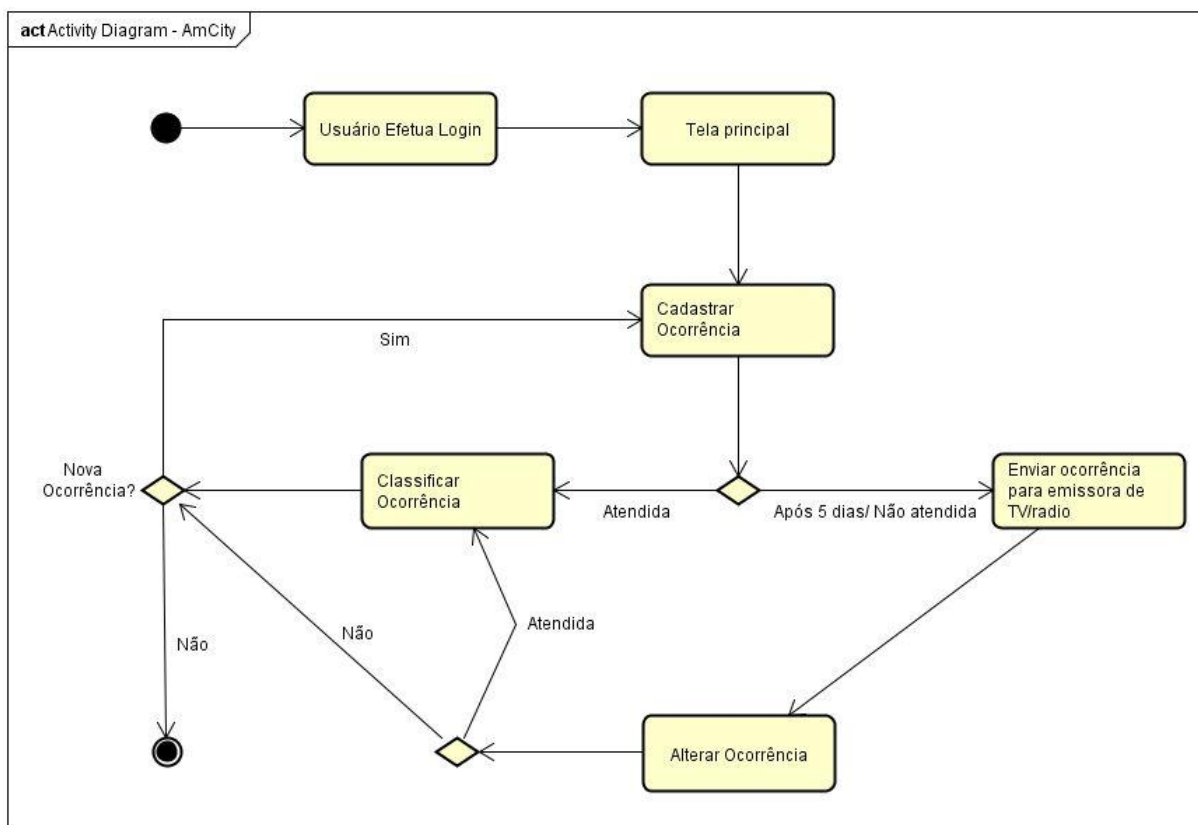


Fonte: Autoria Própria

3.1.7. Diagrama de Atividades

O diagrama de atividades apresentado na figura 11 basicamente mostra as atividades do sistema e como os fluxos ocorrem de uma atividade para a outra. Apresenta uma visão dinâmica do sistema e dá ênfase no controle de fluxos dos objetos.

Figura 11 - Diagrama de Atividades



Fonte: Autoria Própria

4. BANCO DE DADOS

Neste capítulo será apresentado uma breve introdução sobre o que é um Banco de dados e o que é um sistema de gerenciamento de Banco de dados

(SGBD), bem como suas diferenças de segurança tanto do SQL Server desenvolvido e comercializado pela Microsoft quanto MySQL adquirido atualmente pela Oracle.

Primeiramente deve-se entender as diferenças entre Banco de Dados e Sistema e Gerenciamento de Banco de Dados (SGBD). A expressão Banco de Dados originou-se do termo inglês Databanks. Este foi trocado pela palavra Databases – Base de Dados – devido possuir significação mais apropriada (SETZER; CORRÊA DA SILVA, 2005, p. 1).

Segundo DATE (2000, p. 10), “Um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa”. Em outras palavras, um banco de dados é um local onde são armazenados dados necessários à manutenção das atividades de determinada organização, sendo este repositório a fonte de dados para as aplicações atuais e as que vierem a existir.

Atualmente existem vários gerenciadores de banco de dados, dentre eles estão os que mantêm código aberto “*open source*”, porém com algumas diferenças. Nas versões gratuitas há limitações quanto ao tamanho dos dados, segurança, memória e ferramentas reduzidas, mas mesmo tendo algumas deficiências os SGBD’s gratuitos conseguem atender uma boa parte das empresas. Para ELMASRI e NAVATHE (2005, p. 3), na expressão Banco de Dados estão subentendidas as propriedades abaixo:

Um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – Universe of Discourse). As mudanças no minimundo são refletidas no Banco de Dados. Um banco de dados é uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados. Um banco de dados é projetado, construído e populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados. (ELMASRI e NAVATHE, 2005, p. 3)

4.1. Sistema Gerenciador de Banco de Dados (SGBD)

“O principal objetivo de um SGDB é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do banco de dados” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 1).

Entre o banco de dados armazenado e os usuários há um conjunto de programas denominado Sistema Gerenciador de Banco de dados (SGBD) (DATE, 2004, p. 8).

Para tanto, o Sistema Gerenciador de Banco de Dados disponibiliza recursos para definir, construir, manipular, compartilhar, proteger e manter bancos de dados (ELMASRI; NAVATHE, 2005, p. 3). Por definição entende-se a especificação das estruturas de armazenamento (definição dos elementos e respectivos tipos de dados que compõem os registros).

Construção envolve o armazenamento dos dados (registros e relacionamentos). Manipulação refere-se à recuperação e a atualização – inclusão, exclusão e alteração – de dados. Por compartilhamento, a permissão de acesso concorrente a uma base de dados. Proteção” diz respeito à segurança contra falhas de hardware, software e contra acesso não autorizado. Por manutenção, o suporte para o crescimento do banco de dados (ELMASRI; NAVATHE, 2005, p. 4).

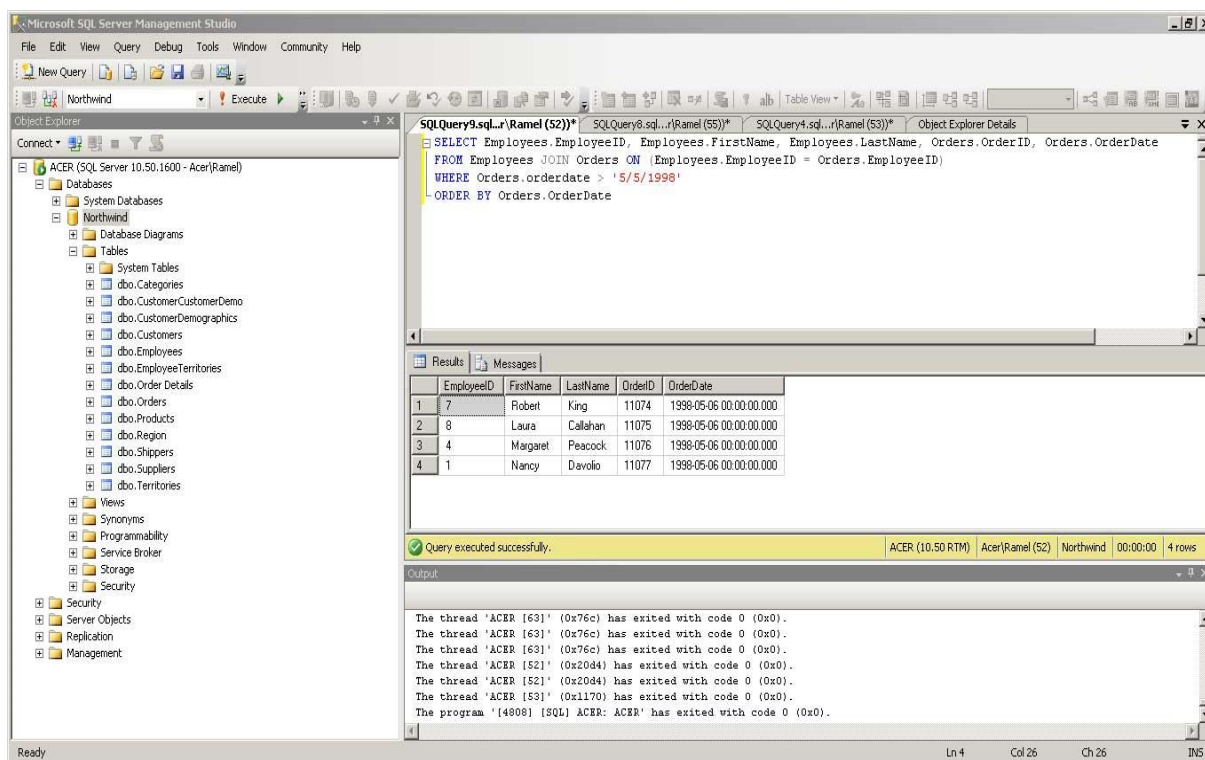
4.2. SQL Server

Segundo OFICINADANET (2007), o SQL Server é um gerenciador de Banco de dados relacional feito pela Microsoft. É um Banco de dados robusto e usado por sistemas corporativos dos mais diversos portes.

O SQL Server fornece uma plataforma de dados confiável, produtiva e inteligente que permite a execução de aplicações de missão críticas mais exigentes, reduz o tempo e o custo com o desenvolvimento e o gerenciamento de aplicações e entrega percepção que se traduz em ações estratégicas (HIRT, 2010). O SQL Server é um banco de dados robusto, usado por sistemas corporativos dos mais diversos portes, Segundo Hirt (2010).

A **Figura 12** apresenta um exemplo da ferramenta de gerenciamento da base de dados do SQL Server.

Figura 12 - SQL Server Management Studio



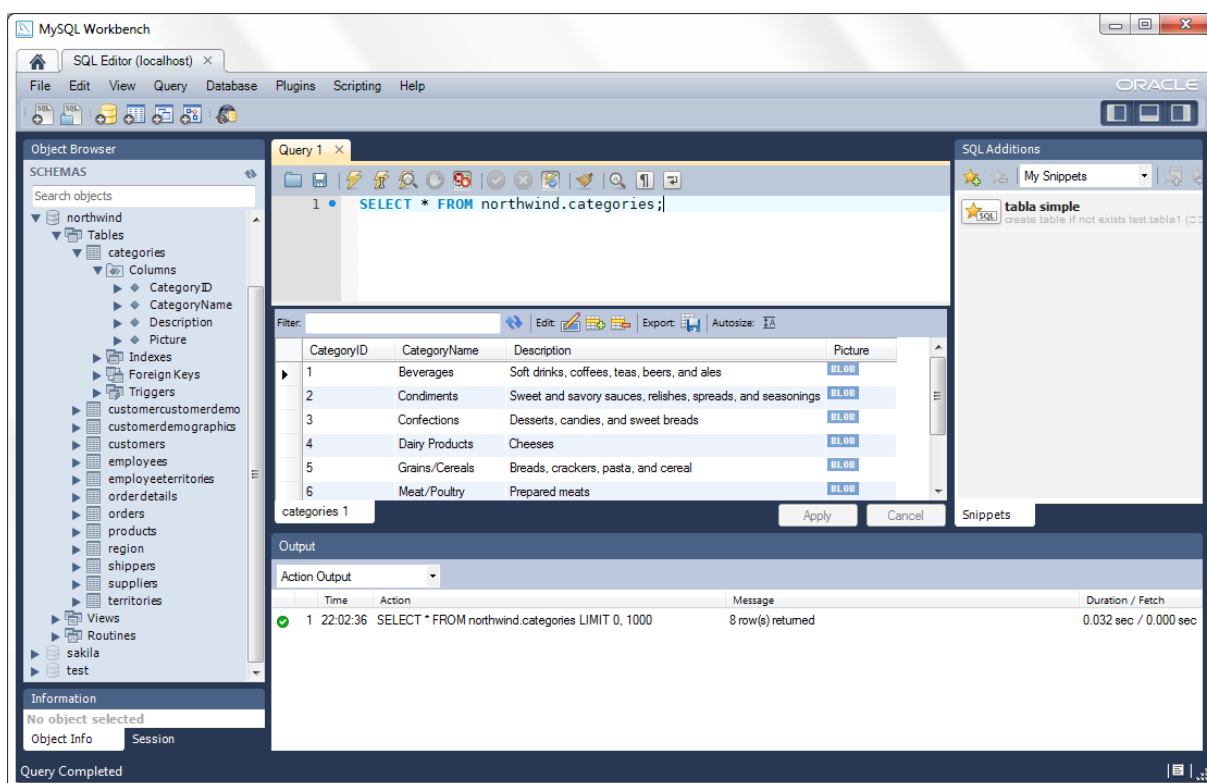
Fonte: <https://visualstudiomagazine.com/Blogs/Data-Driver/2011/05/SQL-Server-Management-Studio-Alternatives.aspx>.

4.3. MySQL

Segundo site do Mysql (2007), o MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. E é atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo (DB-ENGINES, 2017).

MySQL conta com uma importante vantagem competitiva que o fato de ser um software livre. Dentre os bancos de dados open source como o postgres, firebird e outros o Mysql tem se destacado, principalmente para uso na web, segundo Luis (2007). A **Figura 13** temos um exemplo da ferramenta de gerenciamento da base de dados do MySQL.

Figura 13 - MySQL WorkBench



Fonte: <http://mysqlworkbench.org/2012/07/migrating-from-ms-sql-server-to-mysql-using-workbench-migration-wizard/>.

4.4. Diferenças de Segurança entre MySQL e SQL Server

O quesito de segurança é extremamente importante no que se diz do gerenciamento de banco de dados. Tanto o SQL quanto o MySQL possuem um cuidado especial e por isso fazem de tudo para manter seus programas com o maior grau de segurança possível.

Ambos, por exemplo, fornecem suporte nesse quesito para seus clientes, que podem ser pagos ou gratuitos. Indo mais a fundo, o SQL ainda possui uma ferramenta de segurança que o MySQL não agrega, a chamada *Baseline Security Analyzer*.

Ou seja, ele possui essa opção complementar de segurança que é muito útil para os gerenciadores de banco de dados.

Todos os softwares de um mesmo segmento possuem várias semelhanças e diferenças. Não poderia ser diferente com o MySQL e o SQL Server.

4.5. Modelo Entidade-Relacionamento (ER)

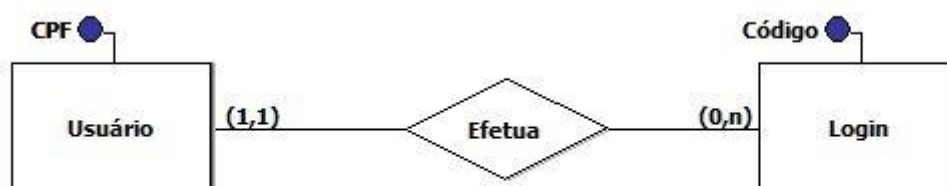
A abordagem deste capítulo será relacionado a Entidade-Relacionamento que é baseada no Modelo Entidade-Relacionamento E-R que foi introduzido por Peter Pin-Shan Chen, em 1976. É um aprimoramento do modelo originalmente proposto, sendo uma das técnicas de modelagem semântica mais conhecidas e, possivelmente, uma das mais utilizadas. DATE (2000, p. 355).

Uma das principais vantagens – talvez seja o motivo maior para sua popularidade – é que além de conceitos o modelo ainda conta com uma técnica de diagramação. Isto permite registrar e comunicar de forma simplificada os principais aspectos do projeto de banco de dados DATE (2000, p. 358). “O modelo ER descreve os dados como entidades, relacionamentos e atributos” (ELMASRI; NAVATHE, 2005, p. 132).

“Um relacionamento é uma associação entre uma ou várias entidades” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 24).

“Um conjunto de relacionamentos é um conjunto de relacionamentos do mesmo tipo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 25). No modelo E-R conjuntos de relacionamentos são representados por losangos. A **Figura 14** temos um exemplo de como relacionar duas entidades.

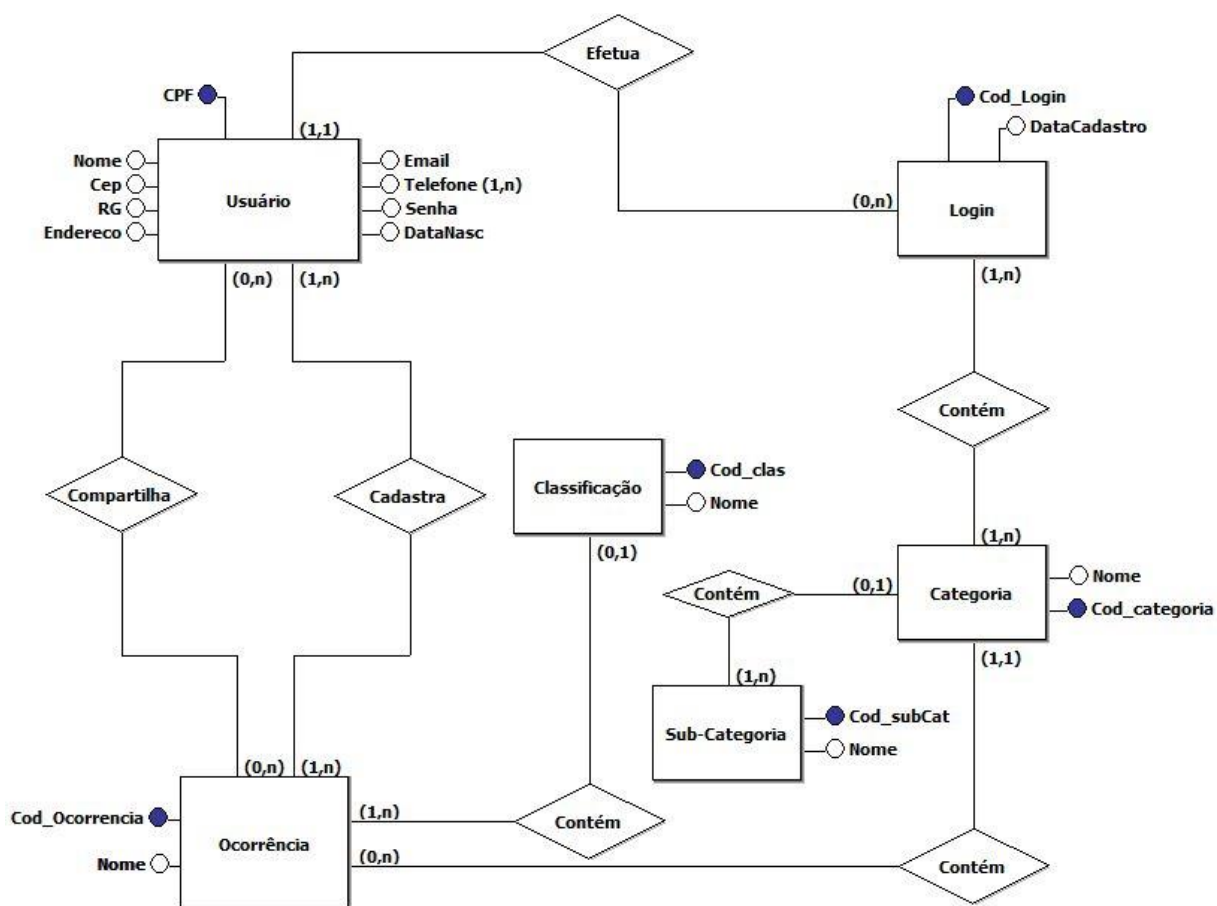
Figura 14 - Exemplo de Modelo Conceitual



Fonte: Autoria Própria

Deve-se ler o diagrama anterior da seguinte maneira: *Usuário* efetua *login*, da esquerda para a direita; e *login* inicia/efetua sessão de *Usuário*, da direita para a esquerda.

Figura 15 - Modelo Conceitual

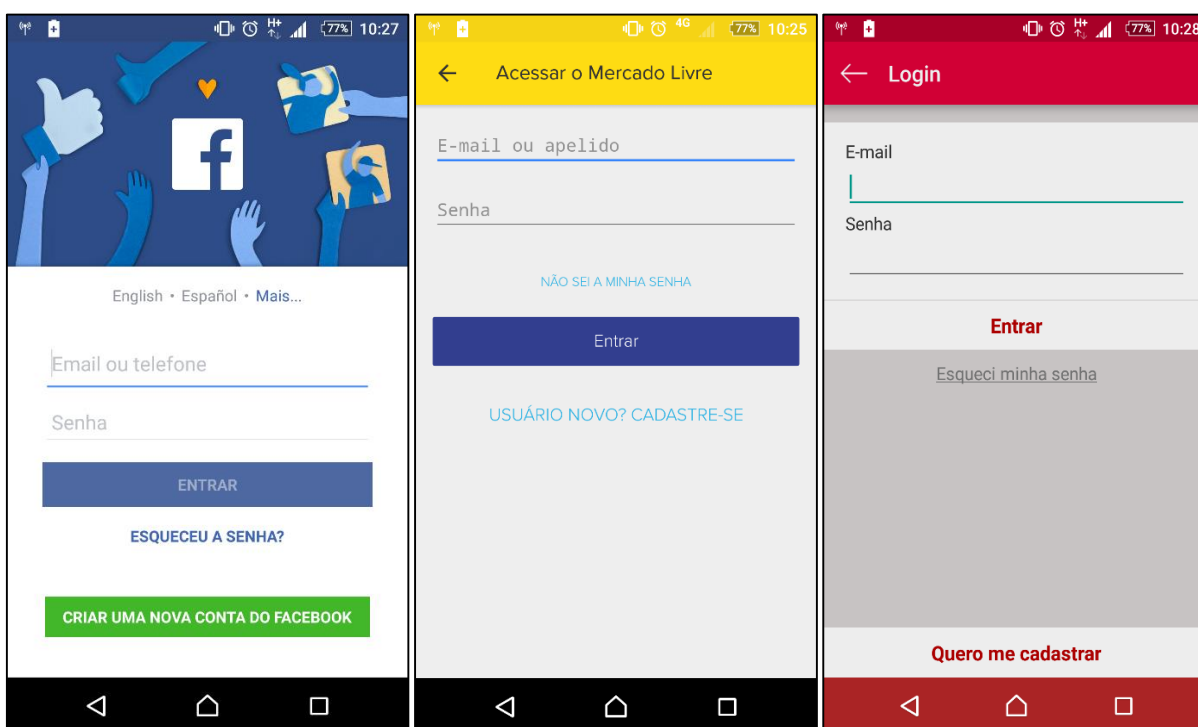


Fonte: Autoria Própria

5. APRESENTAÇÃO DO PROTÓTIPO

Este capítulo tem a finalidade de apresentar as telas do protótipo de um aplicativo de gestão de ocorrência municipal, bem como alguns exemplos das telas de *login* de aplicativos famosos já existentes. Esses exemplos servirão como base para o desenvolvimento futuro do projeto.

Figura 16 - Tela de *login* de aplicativos conhecidos.



Fonte: Autoria Própria

5.1. Tela de Login

A **figura 17** apresenta como seria a tela de *login* principal. A tela de *login* contém dois campos de textos e um botão de acesso, sendo um dos campos para informações de usuário e outro para informações de senha. Nesta tela o usuário também tem a opção de escolher se deseja cadastrar um novo usuário ou recuperar sua senha.

Figura 17 - Tela de Login AmCity

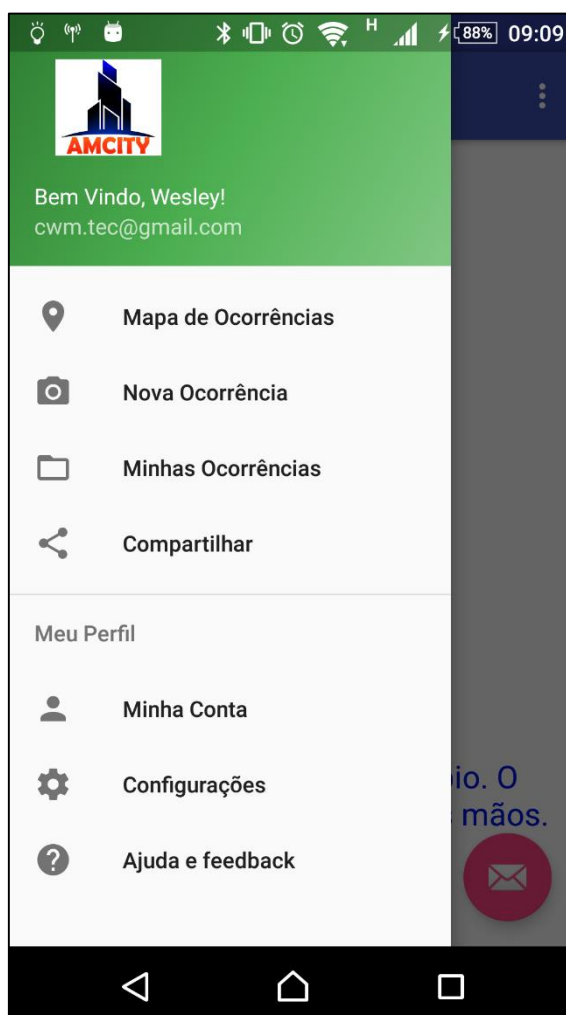


Fonte: Autoria Própria

5.2. Tela de Menu

Observa-se o exemplo da **figura 18** de como seria a tela do Menu principal fragmentada usando a função Navigation Drawer da ferramenta Android Studio. Esta função permite trabalhar com fragmentos de telas, onde o usuário escolhe qual item do menu deseja selecionar, seja cadastrar ou acompanhar dados de uma ocorrência ou até mesmo modificar dados do perfil.

Figura 18 - Tela de Menu

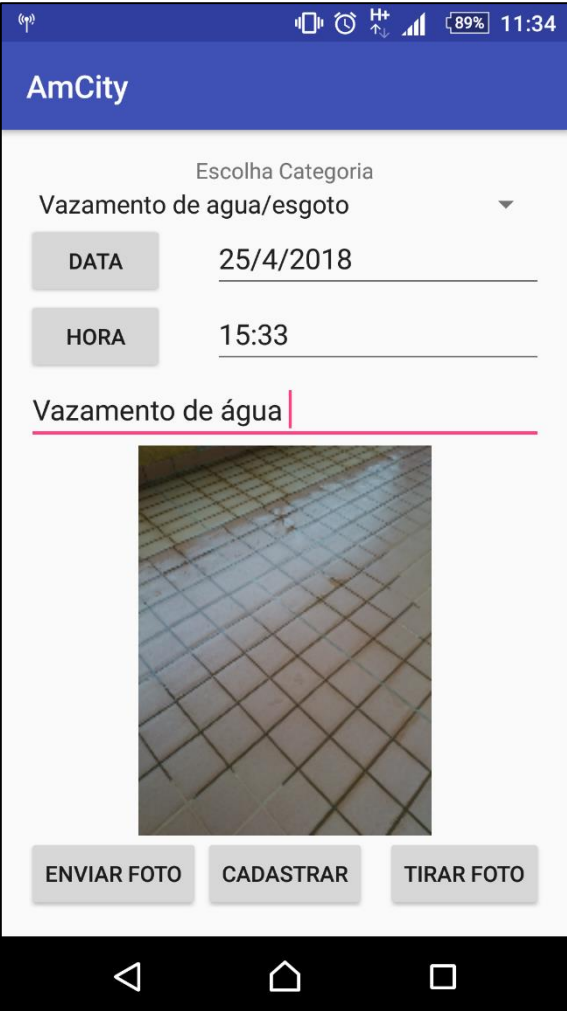


Fonte: Autoria Própria

5.3. Tela de Cadastro de ocorrências

Observa-se o exemplo da **figura 19** de como seria a tela de cadastro de ocorrências, podendo escolher categoria, cadastrar Data, horário e inserir foto tanto da galeria de imagens quanto da câmera referente ao local da ocorrência.

Figura 19 - Cadastro de ocorrências



The screenshot displays the 'AmCity' mobile application interface for incident registration. At the top, there is a blue header with the app name 'AmCity'. Below the header, the screen shows a form with the following elements:

- A dropdown menu labeled 'Escolha Categoria' with the selected option 'Vazamento de agua/esgoto'.
- An input field labeled 'DATA' with the value '25/4/2018'.
- An input field labeled 'HORA' with the value '15:33'.
- A text input field containing 'Vazamento de água' with a pink underline.
- A photo preview showing a tiled floor with a water stain.
- Three buttons at the bottom: 'ENVIAR FOTO', 'CADASTRAR', and 'TIRAR FOTO'.

The status bar at the top indicates 89% battery and the time 11:34.

Fonte: Autoria Própria

5.4. Tela de acompanhamento das ocorrências.

Observa-se o exemplo da **figura 20** de como seria a tela de acompanhamento das ocorrências podendo verificar se foram atendidas ou finalizadas.

Figura 20 - Tela de Acompanhamento



Fonte: Autoria Própria

5.5. Tela de compartilhamento das ocorrências

Observa-se o exemplo da **figura 21** de como seria a tela de compartilhamento de ocorrências podendo compartilhar as ocorrências nas redes sociais mais conhecidas atualmente.

Figura 21 - Tela de compartilhamento das ocorrências

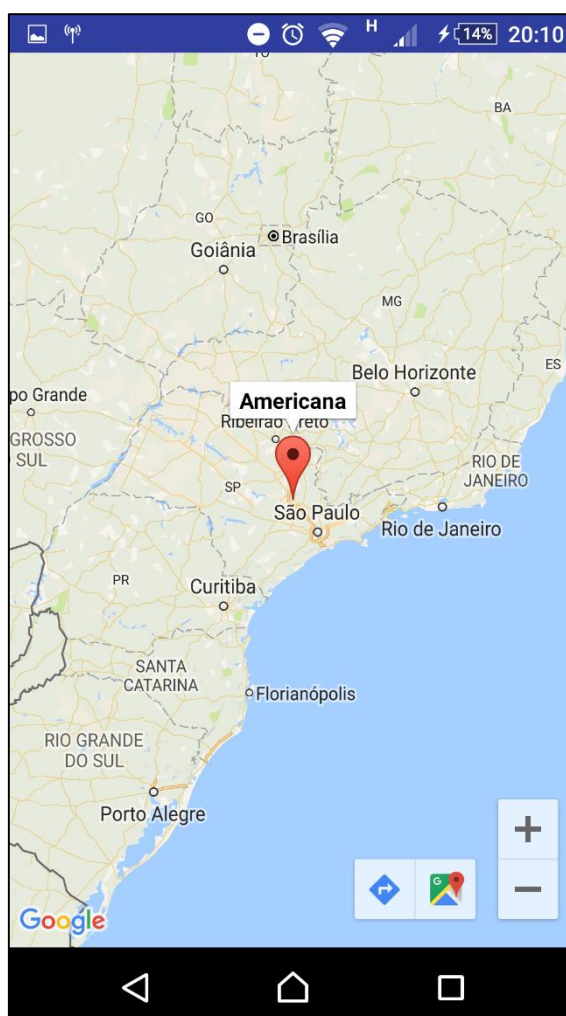


Fonte: Autoria Própria

5.6. Tela de Mapa de ocorrências

Observa-se o exemplo da **figura 22** de como seria a tela das ocorrências mais próximas da cidade, para tal feito foi adicionado API do Google maps para informar aos usuários quais as ocorrências mais próximas feitas por outros usuários.

Figura 22 - Tela das ocorrências recentes

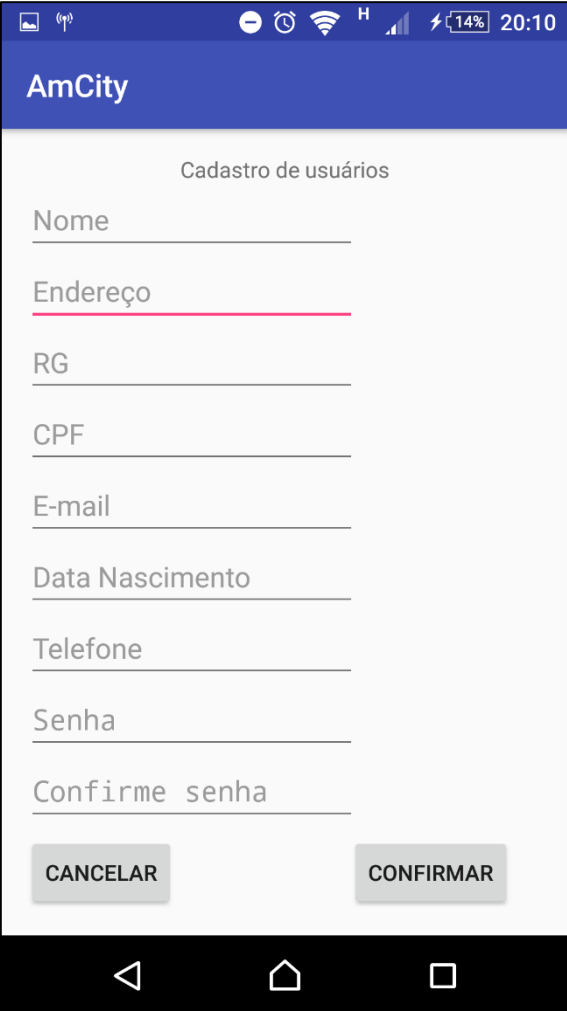


Fonte: Autoria Própria

5.7. Tela de Cadastro de usuário

Observa-se o exemplo da **figura 23** de como seria a tela de cadastrar um novo usuário. A tela de cadastro de usuário possui dois botões um para cancelar e outro para confirmar cadastro.

Figura 23 - Tela de cadastro de usuário



A imagem mostra a tela de cadastro de usuário do aplicativo AmCity. No topo, há uma barra azul com o nome 'AmCity'. Abaixo, o título 'Cadastro de usuários' é centralizado. A tela contém os seguintes campos de entrada:

- Nome
- Endereço
- RG
- CPF
- E-mail
- Data Nascimento
- Telefone
- Senha
- Confirme senha

Na base da tela, há dois botões: 'CANCELAR' e 'CONFIRMAR'. A barra de navegação do Android está visível na base da tela.

Fonte: Autoria Própria

5.8. Tela de Recuperação de senha

Observa-se o exemplo da **figura 24** de como seria a tela de recuperar senha de usuário.

Figura 24 - Tela de recuperar senha



Fonte: Autoria Própria

6. CONSIDERAÇÕES FINAIS

Após uma breve análise sobre a grande quantidade de problemas ocorridos tanto nas metrópoles quanto nos municípios do interior de São Paulo, sejam eles problemas na rede de água e esgoto ou iluminação pública, deve-se a uma gestão inadequada da cidade.

A partir da apresentação e análise desses dados, este trabalho de conclusão de curso teve como objetivo principal apresentar a elaboração de um protótipo de um aplicativo para melhorar e apoiar na gestão municipal de ocorrências de uma determinada cidade.

Em relação aos objetivos específicos, pode-se afirmar, que a maioria deles foram atendidos, o aplicativo possibilita ao cidadão cadastrar, atualizar, remover as ocorrências, bem como acompanhar, classificar e escolher uma ocorrência por categoria ou até mesmo compartilhar a ocorrência através das redes sociais. Porém as etapas de desenvolvimento deste protótipo foram bem complicadas, devido à grande complexidade da ferramenta Android e seus componentes, também por não ter experiência com desenvolvimento de aplicativos.

O aplicativo por sua vez proporcionou este conhecimento me aprimorando dos estudos sobre as tecnologias utilizadas atualmente como a plataforma Android SDK, Diagramas da linguagem UML, *WebServices* e a API que são componentes fundamentais para o desenvolvimento de um aplicativo.

Além do levantamento dessas tecnologias, houve também grande preocupação com os aspectos de segurança, como, qual dos tipos de protocolos de autenticação deveriam ser utilizados. Dessa forma percebeu-se que o uso do protocolo HTTP Basic Auth é a opção de autenticação mais viável para o desenvolvimento deste protótipo.

O protótipo foi desenvolvido e testado através de um smartphone, podendo assim ser apresentado para uma instituição Municipal para que possa ser avaliado e possivelmente implementado. Sugere-se para trabalhos futuros a implementação de novos recursos tais como: nível de prioridade da ocorrência, baixo, médio ou alto; link de denúncia por meio de emissoras de rádio ou televisão; mostrar ocorrências

nas proximidades do local. Esses seriam atrativos interessantes para contribuir e apoiar na gestão das prefeituras municipais.

REFERÊNCIAS

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas UML**, Rio de Janeiro, Elsevier, 2007 – 2ª Reimpressão

BOOCH, G; RUMBAUGH, J e JACOBSON, I: **UML, Guia do Usuário**. Tradução; Fábio Freitas da Silva, 2ª. ed. Rio de Janeiro, Campus ,2005. (p. 3-13).

COGNATIS. **O que é geocodificação?** 01/01/2017. Disponível em: <<http://www.cognatis.com.br/o-que-e-geocodificacao>>. Acesso em 011 mai. 2017, às 08:23mim.

CORDEIRO, Felipe. **O que é Android SDK**. 20/02/2017. Disponível em <<http://www.androidpro.com.br/android-sdk/>>. Acesso em: 28 fev. 2017, às 14:45mim.

DANTAS, Daniel Chaves Toscano. **Simple Object Access Protocol (SOAP)**. Disponível em:< <http://videira.ifc.edu.br/fice/wp-content/uploads/sites/27/2015/11/15-artigo-Kellen.pdf>>. Acesso em 06 de mai de 2017, às 17:49.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. 8ª. ed. Rio de Janeiro: Campus, 2000 (p. 355-358).

DB-ENGINES, Engines. **DB-Engines Ranking**. 01/05/2017. Disponível em: <<https://db-engines.com/en/ranking> >. Acesso em 01 mai. 2017, às 15:21mim.

DEVMEDIA, Allan Douglas. **Introdução à Google Maps API**. 17/01/2013. Disponível em: <<http://www.devmedia.com.br/introducao-a-google-maps-api/26967>>. Acesso em 08 mai. 2017, às 20:14mim.

DEVMEDIA, Higor Medeiros. **Application Programming Interface: Desenvolvendo APIs de Software**. 27/05/2014. Disponível em: <<http://www.devmedia.com.br/application-programming-interface-desenvolvendo-apis-de-software/30548>>. Acesso em 05 mai. 2017, às 22:02mim.

DEVMEDIA, Leandro Ribeiro. **O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML**. 19/01/2012. Disponível em: <<http://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>>. Acesso em 12 mai. 2017, às 11:05mim.

DICIO. **Mobilidade**. Disponível em <<https://www.dicio.com.br/aplicativo/>>. Aceso em 15 fev. 2017, às 18:05mim.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 6^a. ed. São Paulo: Pearson, 2005 (p. 3-132).

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doutorado) — University of California, Irvine, 2000. Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 08 mar. 2017.

HIRT, Alan. Pro SQL Server 2008: **Failover Clustering**. New York, NY: Apress, 2010, (p. 335-337).

JORNAL DA USP In: **Jornal da Usp**, Disponível em: <<http://jornal.usp.br/universidade/mercado-de-aplicativos-cresce-no-brasil-e-alunos-da-usp-em-sao-carlos-conquistam-espaco-no-cenario/>>. Acesso em: 18 fev. 2017, às 17h45mim.

KUHN, Clayton Eduardo KUHN. **Elaboração de um Protótipo de Aplicativo para Acompanhamento de Requisições de Táxi**. Formato PDF. Disponível em: <http://www2.dainf.ct.utfpr.edu.br/esp/monografias-de-especializacao-da-turma-vii-2011-2012/CT_JAVA_VII_2011_03.PDF/at_download/file>. Acesso em: 22 mar. 2017, às 11:52mim.

LECHETA, Ricardo R. **Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 2^a. ed. - São Paulo, Novatec, 2010, (p. 19-30).

LEE, Wei-Meng. **Introdução ao Desenvolvimento de Aplicativos para Android**. 1^a. ed. Rio de Janeiro, Ciencia Moderna Ltda, 2011. (p. 4-12).

LUIS, Blog. **Banco de dados Oracle, Mysql Access e SQL Server: Diferenças**. 14/09/2007. Disponível em: < Banco de dados Oracle, Mysql, Access e SQL Server – Diferenças>. Acesso em 29 mai. 2017, às 21:44mim.

MEIRA, Guilherme Tebaldi Meira. **Projeto e desenvolvimento de um aplicativo móvel para acesso ao Portal do Aluno da UFES na plataforma Android**. Formato PDF. Disponível em:<<http://docplayer.com.br/18455917-Projeto-e-desenvolvimento-de-um-aplicativo-movel-para-acesso-ao-portal-do-aluno-da-ufes-na-plataforma-android.html>>. TG (Trabalho de Graduação do Curso superior de Engenharia da Computação), Universidade Federal do Espírito Santo, 2014. Acesso em: 15 mar. 2017.

MENÉNDEZ, Andrés Ignacio Martínez. **Uma ferramenta de apoio ao desenvolvimento de Webservices**. Formato PDF. Disponível em: <http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2002/Dissertacao_AndresIgnacioMartinezMenendez.pdf>. Acesso em: 05 mar. 2017, às 09:22mim.

MOBILEIN. **Conceitos chave do Android – parte 1 – Camadas**. 2010. Disponível em: <<http://mobileinn.com.br/?p=55>>. Acesso em: 25 fev. 2017, às 18:19mim.

MYSQL, Mysql. **Why Mysql?** 10/02/2007. Disponível em: <<https://www.mysql.com/why-mysql/>>. Acesso em 01 mai. 2017, às 16:58mim.

OFICIALDANET. **SQL Server.** 13/09/2007. Disponível em: <http://www.oficinadanet.com.br/artigo/501/sql_server>. Acesso em 01 mai. 2017, às 13:33mim.

PRADO, Sérgio. **Introdução ao funcionamento interno do Android.** 15/08/2011. Disponível em <<http://www.sergioprado.org/2011/08/15/introducao-aofuncionamentointerno-do-android/>>. Acesso em: 21 fev. 2017, às 22:13mim.

RAMSEY, Bem. **Dominando OAuth 2.0.** 09/12/2016. Disponível em: <<https://imasters.com.br/desenvolvimento/dominando-oauth-2-/?trace=1519021197>>. Acesso em 14 mai. 2017, às 19:23mim.

SETZER, Valdemar W.; SILVA, Flávio Soares Corrêa da. **Bancos de Dados: Aprenda o que são, melhore seu conhecimento, construa os seus.** São Paulo: Edgard Blücher, 2005 (p. 1-18).

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de Banco de Dados.** 3ª. ed. São Paulo: Makron Books, 1999 (p. 1-25).

SINHAL, Ankit. **Transport Layer Security.** 02/12/2016. Disponível em: <<http://androidjavapoint.blogspot.com.br/2016/12/transport-layer-security.html>>. Acesso em: 14 mai. 2017, às 16:11mim.

SOMMERVILLE, Ian, **Engenharia de Software**, Tradução; de Selma Melnikoff, Reginaldo Arakaki e Edilson de Andrade Barbosa, 8ª. ed. – São Paulo, Pearson Addison Wesley, 2007, (p. 5-10).

STRICKLAND, Jonathan. **Como funciona o Android (Google Phone).** 16/09/2009. Disponível em <<http://informatica.hsw.uol.com.br/google-phone2.htm>>. Acesso em 19 fev. 2017, às 19:30mim.

TAKASHI, Nicolas. **REST ou SOAP: Qual eu devo usar?** 02/02/2015. Disponível em: <<http://ntakashi.net/rest-ou-soap-qual-eu-devo-usar/>>. Acesso em 29 mar. 2017, às 16:43mim.

VAMOSI, Robert. **Gmail cookie stolen via Google Spreadsheets.** 14/01/2008. Disponível em: <<https://www.cnet.com/news/gmail-cookie-stolen-via-google-spreadsheets/>>. Acesso em 13 mai. 2017, às 07:09mim.

ZEIGER, S. **An Invitation to Servlets.** 2008. Disponível em: <<http://www.novocode.com/doc/servlet-essentials/chapter1.html>>. Acesso em: 18 mai. 2014. Citado na página 40.

APÊNDICE A - Código-Fonte do Protótipo AmCity

Tela Login:

```

public class Login extends AppCompatActivity {

    TextView txtCadastro;
    TextView txtRecuperar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        Button btlogin = (Button)findViewById(R.id.btlogin);

        //Abrindo nova activity pelo Textview
        txtCadastro = (TextView) findViewById(R.id.txtCadastro);

        txtCadastro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent abrecadastro = new Intent(Login.this, CadastroUsuario.class);
                startActivity(abrecadastro);
            }
        });
        txtRecuperar = (TextView) findViewById(R.id.txtRecuperar);

        txtRecuperar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent abreRecuperar = new Intent(Login.this, RecuperarSenha.class);
                startActivity(abreRecuperar);
            }
        });

        btlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TextView tLogin = (TextView) findViewById(R.id.tLogin);
                TextView tSenha = (TextView) findViewById(R.id.tSenha);
                String login = tLogin.getText().toString();
                String senha = tSenha.getText().toString();
                if (login.equals("Admin") && (senha.equals("Admin"))) {
                    alerta("Login Realizado com Sucesso");
                    Intent mudanca = new Intent(Login.this, TelaPrincipal.class);
                    startActivity(mudanca);

                } else {
                    alerta("Usuário ou senha incorretos, tente novamente!"); } } }); }

    private void alerta(String s)
    {
        Toast.makeText(this,s,Toast.LENGTH_LONG).show();
    }
}

```

```

    }
}

```

Tela de Menu - Navigation Drawer:

```

public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_inicio) {

        Intent abremapa = new Intent(TelaPrincipal.this, MapsActivity.class);
        startActivity(abremapa);
        //InicioFragment iniciofragment = new InicioFragment();

        //getSupportFragmentManager().beginTransaction().replace(R.id.conteudo_fragment, iniciofragment).commit();

    } else if (id == R.id.nav_novaoco) {

        Intent abreocorrencia = new Intent(TelaPrincipal.this, NewOcorrencia.class);
        startActivity(abreocorrencia);

        //NovaOcorrenciaFragment novacorrencia = new NovaOcorrenciaFragment();
        //
        getSupportFragmentManager().beginTransaction().replace(R.id.conteudo_fragment, novacorrencia).commit();

    } else if (id == R.id.nav_minhasoco) {

        Intent abreminhas_oco = new Intent(TelaPrincipal.this, MinhasOcorrencias.class);
        startActivity(abreminhas_oco);

    } else if (id == R.id.nav_compartilhar) {

        Intent abre_compartilhar = new Intent(TelaPrincipal.this, Compartilhar.class);
        startActivity(abre_compartilhar);

    } else if (id == R.id.nav_conta) {

    } else if (id == R.id.nav_config) {

    } else if (id == R.id.nav_ajuda) {

    }

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
}

```

Tela de Cadastro de ocorrências:

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_nov);

    btData = (Button) findViewById(R.id.btData);
    btHora = (Button) findViewById(R.id.btHora);
    edData = (EditText) findViewById(R.id.edData);
    edHora = (EditText) findViewById(R.id.edHora);

    btData.setOnClickListener(this);
    btHora.setOnClickListener(this);
    sistemas = (Spinner) findViewById(R.id.spinSO);
    ArrayAdapter adapter = ArrayAdapter.createFromResource(this, R.array.categorias,
    android.R.layout.simple_spinner_item);
    sistemas.setAdapter(adapter);

    TextView ok = (TextView) findViewById(R.id.msg);

    AdapterView.OnItemClickListener escolha = new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            String item = sistemas.getSelectedItem().toString();
            Toast.makeText(getApplicationContext(), "Item Selecionado" + item,
    Toast.LENGTH_SHORT).show();
        }
        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        } };
    sistemas.setOnItemClickListener(escolha);
}
private void alerta(String s)
{
    Toast.makeText(this,s,Toast.LENGTH_LONG).show();
}
public void pegarImg (View view)
{
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    startActivityForResult(intent, GALERIA_IMAGES);
}
public void tirarfoto (View view)
{
    Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
    startActivityForResult(intent, 0);
}
} // @Override

public void onActivityResult(int requestCode,int resultCode,Intent data){
    if(data != null)
    {
        Bundle bundle = data.getExtras();
        if(bundle != null)
    
```

```

{
    Bitmap img = (Bitmap) bundle.get("data");

    ImageView iv = (ImageView) findViewById(R.id.imageView3);
    iv.setImageBitmap(img);

} }
else //MINI BANCO DE DADOS PARA BUSCAR UMA FOTO
if(requestCode == GALERIA_IMAGES){
    if(requestCode == RESULT_OK){
        Uri imagemSelecionada = data.getData();
        String[] columnas = {MediaStore.Images.Media.DATA};
        Cursor cursor = getContentResolver().query(imagemSelecionada, columnas, null, null,
null);

        cursor.moveToFirst();
        int indexColuna = cursor.getColumnIndex(columnas[0]);
        String pathIng = cursor.getString(indexColuna);
        cursor.close();

        Bitmap bitmap = BitmapFactory.decodeFile(pathIng);
        ImageView campolmagem = (ImageView) findViewById(R.id.imageView3);
        campolmagem.setImageBitmap(bitmap);
    } } }

@Override
public void onClick(View v) {

    if(v==btData)
    {
        final Calendar c = Calendar.getInstance();
        dia =c.get(Calendar.DAY_OF_MONTH);
        mes =c.get(Calendar.MONTH);
        ano =c.get(Calendar.YEAR);
        DatePickerDialog datePickerDialog = new DatePickerDialog(this, new
DatePickerDialog.OnDateSetListener(){

            @Override
            public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
                edData.setText(dayOfMonth+"/"+(monthOfYear+1)+"/"+year);
            }
        },dia,mes,ano);
        datePickerDialog.show();
    }
    if(v==btHora)
    {
        final Calendar c= Calendar.getInstance();
        hora = c.get(Calendar.HOUR_OF_DAY);
        minutos =c.get(Calendar.MINUTE);

        TimePickerDialog timePickerDialog = new TimePickerDialog(this, new
TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                edHora.setText(hourOfDay+":"+minute);
            }
        },hora,minutos,false);
        timePickerDialog.show();
    }
}

```


Tela de acompanhamento das ocorrências:

```
public class MinhasOcorrencias extends AppCompatActivity {

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

Tela de compartilhamento das ocorrências:

```
public class Compartilhar extends AppCompatActivity {

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

Tela de Mapa de ocorrências:

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
    GoogleMap.OnMapClickListener {

    private GoogleMap mMap;
    private LocationManager locationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {

        mMap = googleMap;
        mMap.setOnMapClickListener(this);
        //opcao de dar ZOOM + ou -
        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.getUiSettings().isRotateGesturesEnabled();
        mMap.getUiSettings().isMyLocationButtonEnabled();
        // coordenadas de americana
        LatLng americana = new LatLng(-22.7388,-47.3319);
        MarkerOptions marker = new MarkerOptions();
        marker.position(americana);
    }
}
```

```

marker.title("Americana");
mMap.addMarker(marker);
mMap.moveCamera(CameraUpdateFactory.newLatLng(americana));
}
@Override
public void onMapClick(LatLng latLng) {
    Toast.makeText(this, "Cordenadas: " + latLng.toString(), Toast.LENGTH_SHORT).show();
}
}

```

Tela de Cadastro de usuário:

```

public class CadastroUsuario extends AppCompatActivity {

    Button btnCancelar, btnConfirmar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cadastro_usuario);

        btnCancelar = (Button) findViewById(R.id.btnCancelar);

        btnCancelar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
}

```

Tela de Recuperação de senha:

```

public class RecuperarSenha extends AppCompatActivity {

    Button btnVoltar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recuperar_senha);
        btnVoltar = (Button) findViewById(R.id.btnVoltar);
        btnVoltar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
}
}

```