UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
III ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

EVALDO AUGUSTO BIANCHI

SISTEMA DE ENVIO E RECEBIMENTO DE MENSAGENS PARA PLATAFORMA ANDROID UTILIZANDO SPRING BOOT E GOOGLE CLOUD MESSAGING

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2015

EVALDO AUGUSTO BIANCHI

SISTEMA DE ENVIO E RECEBIMENTO DE MENSAGENS PARA PLATAFORMA ANDROID UTILIZANDO SPRING BOOT E GOOGLE CLOUD MESSAGING

Trabalho de Conclusão de Curso, apresentado ao III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, campus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Robison Cris Brito.

PATO BRANCO
2015

SISTEMA DE ENVIO E RECEBIMENTO DE MENSAGENS PARA PLATAFORMA ANDROID UTILIZANDO SPRING BOOT E GOOGLE CLOUD MESSAGING

Por

Evaldo Bianchi

Esta monografia foi apresentada às 19h00 do dia 09 de setembro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Msc. Robison Cris Brito
Orientador
UTFPR – Campus Pato Branco

Prof. Msc. Vinicius Pegorini
Banca
UTFPR – Campus Pato Branco

Profa. Msc. Soetaine Rodrigues Ascari
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Coordenador do curso de Especialização
UTFPR – Campus Pato Branco

RESUMO

BIANCHI, Evaldo Augusto. Sistema de Envio e Recebimento de Mensagens para Plataforma Android Utilizando Spring Boot e Google Cloud Messaging. 2015. 56 f. Monografia de especialização. III Curso de Especialização em Java. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2015.

Alguns dos sistemas com o maior número de usuários e maior popularidade da atualidade, são os que, de alguma forma, permitem aos usuários, enviar e receber mensagens. O presente trabalho visa demonstrar o desenvolvimento de um sistema simples para troca de mensagens entre usuários, utilizando frameworks como Spring Boot e Google Cloud Messaging, demonstrando como é o funcionamento básico desse tipo de aplicação, além de integrar com um sistema Web já existente. Por ser de fácil acesso, possuir vários recursos e ferramentas que facilitam o desenvolvimento, a plataforma Android foi escolhida para implementação do aplicativo. Como resultado, obteve-se um aplicativo funcional, que aprimora a experiência do usuário e também foi possível concluir que a utilização de frameworks facilita e agiliza o desenvolvimento de aplicações Web.

Palavras-chave: Android, Spring Boot, Mensagem, Mobile.

ABSTRACT

BIANCHI, Evaldo Augusto. Sistema de Envio e Recebimento de Mensagens para Plataforma Android Utilizando Spring Boot e Google Cloud Messaging. 2015. 56 f. Monografia de especialização. III Curso de Especialização em Java. Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2015.

Some of the systems with more users and greater popularity today, are those who, somehow, allow users to send and receive messages. This study aims to demonstrate the development of a simple system for exchanging messages between users, using frameworks like Spring Boot and Google Cloud Messaging, demonstrating how the basic operation of this type of application, as well as integrate with an existing Web system. Because it is easily accessible, has several features and tools that facilitate the development, the Android platform has been chosen to application deployment. As a result, we obtained a functional application that enhances the user experience and it was also possible to conclude that the use of frameworks facilitates and speeds the development of Web applications.

Palavras-chave: Android, Spring Boot, Message, Mobile.

LISTA DE FIGURAS

LISTAGENS DE CÓDIGOS

LISTA DE QUADROS

LISTA DE SIGLAS

ADT	*Android Development Tools*
API	*Application Programming Interface*
APP	*Application*
AVD	*Android Virtual Device*
CMEI	*Centro Municipal de Ensino Infantil*
GCM	*Google Cloud Messaging*
HTTP	*HyperText Transfer Protocol*
IBM	*Internacional Business Machines*
ID	*Identity*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
MP3	*MPEG Layer 3*
PDA	*Personal Digital Assistants*
REST	*Representational State Transfer*
SADI	*Sistema Automatizado de Desenvolvimento Infantil*
SD	*Secure Digital*
SDK	*Software Development Kit*
SGDB	*Sistema de Gerenciamento de Banco de Dados*
SMS	*Short Message Service*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SysML	*Systems Modeling Language*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
WSDL	*Web Services Description Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, os seus objetivos e a justificativa. Este é finalizado com a organização do texto por meio da apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

O uso de aplicativos para comunicação instantânea nunca foi tão utilizado como nos dias de hoje. Os aplicativos para o envio de mensagens têm se tornado cada vez mais populares, e o grande motivo é a popularização dos smartphones e principalmente, da Internet.

Apesar de parecer simples para os usuários, esses aplicativos se utilizam de complexos sistemas de informação, que contemplam grande volume de dados, escalabilidade e confiabilidade.

Quando integrados a sistemas de gestão, esses aplicativos podem ampliar ainda mais as funcionalidades, trazendo o ambiente para a portabilidade.

O SADI ou Sistema Automatizado de Desenvolvimento Infantil (PEREIRA e MELLO, 2014), desenvolvido para o acompanhamento pedagógico de crianças em creches, possui uma funcionalidade muito importante, que é o envio de mensagens entre os usuários, que podem ser professores, diretores e principalmente, os pais dos alunos. Mensagens de avisos e informações, sendo estas de interesse dos usuários.

O aplicativo a ser desenvolvido pretende ampliar a funcionalidade dessas trocas de mensagens, tornando mais prático e eficiente para os usuários. As mensagens enviadas do sistema Web serão recebidas pelo aplicativo mobile, notificando o usuário e vice-versa.

O aplicativo mobile exige facilidade de uso, ou seja, o usuário deve ter todas as informações que precisa ao seu alcance e não deve ter dificuldade para usar o sistema e suas funcionalidades, desta forma, o desenvolvimento de interfaces simples e intuitivas são muito importantes para este tipo de aplicativo.

Assim, este trabalho apresenta o desenvolvimento de um sistema mobile utilizando ferramentas que facilitam a implementação e proporcionam uma boa experiência para o usuário.

1.2 OBJETIVOS

O objetivo geral se refere à finalidade principal de realização deste trabalho. Os objetivos específicos complementam o objetivo geral.

1.2.1 Objetivo Geral

Implementar um sistema mobile para o envio e recebimento de mensagens integrado a um sistema Web já existente.

1.2.2 Objetivos Específicos

- Demonstrar a utilização do serviço do Google Cloud Messaging (GCM);
- Demonstrar a facilidade do desenvolvimento com Spring Boot;
- Analisar e desenvolver o sistema proposto com uma interface simples e de fácil utilização.

1.3 JUSTIFICATIVA

A opção de desenvolver um sistema mobile visa, além das funcionalidades do sistema, demonstrar o uso das tecnologias Spring Boot e Google Cloud Messaging, destacando como esses recursos facilitam e ampliam o desenvolvimento de aplicações mobile com envio de informações.

A utilização do Spring Boot garante ao desenvolvedor um ganho de produtividade considerável no que diz respeito aos recursos e configurações dos projetos, pois utiliza um sistema de convenções, trazendo o que é necessário para o desenvolvimento de forma padrão, podendo ser personalizado sem dificuldades.

Um aplicativo mobile para a troca de mensagens entre os usuários de um sistema Web, garante mais facilidade para os usuários e amplia a utilização deste recurso, além de torna-lo mais eficiente, pois os usuários irão receber notificações onde quer que estejam, bastando apenas estar conectado à internet, com a possibilidade de responder ou ainda enviar novas mensagens para outros usuários, com toda a praticidade de um *smartphone*.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia e o contexto do sistema, incluindo os objetivos e a justificativa.

O Capítulo 2 contém o referencial teórico, que está centrado no funcionamento básico dos aplicativos de mensagens, no desenvolvimento de sistemas Web e mobile.

No Capítulo 3 estão os materiais e o método utilizados no desenvolvimento deste trabalho. Os materiais se referem às tecnologias e ferramentas utilizadas e o método contém as principais atividades realizadas para o desenvolvimento deste trabalho.

O Capítulo 4 apresenta o sistema desenvolvido, com a modelagem e documentação, a demonstração das telas, o funcionamento do sistema e alguns códigos importantes.

No Capítulo 5 está a conclusão com as considerações finais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho que está centrado em aplicativos de envio de mensagens, no sistema móvel Android, no desenvolvimento de Web services e SpringBoot e no sistema Web SADI. Isso por que o resultado desse estudo é justamente um aplicativo de mensagens para a plataforma Android comunicando-se com um sistema Web através de Web services desenvolvidos com o Spring Boot.

2.1 APLICATIVOS DE MENSAGENS

Os aplicativos de trocas de mensagem estão sendo cada vez mais utilizados, muitos usuários preferem esse tipo de serviço a ter que gastar dinheiro mandando um SMS (Short Message Service) ou realizando uma ligação telefônica.

Por esse motivo, o mercado está repleto de opções voltadas para este fim, com as mais variadas funções e funcionalidades, mas todos com o mesmo princípio de funcionamento.

Atualmente um dos aplicativos de maior sucesso é o WhatsApp (WHATSAPP, 2015), com inúmeras funções e uma navegação simples, consegue se destacar por utilizar apenas o número de telefone do usuário para enviar as mensagens via internet. O aplicativo já possui mais de 1 bilhão de downloads na loja virtual do sistema Android, com um total de aproximadamente 30 bilhões de mensagens por dia, superando o envio de SMS mundial (TUDOCELULAR, 2015). A Figura 1 demonstra algumas telas do aplicativo.

Figura 1 - WhatsApp
Fonte: GOOGLE PLAY, 2015.

Outros sistemas utilizados para troca de mensagens são o Viber, ICQ e Telegram, muito semelhantes ao já citado WhatsApp. Ainda existem as opções que são serviços on-line e também possuem o aplicativo para a troca de mensagens, como Messenger do Facebook, Skype e Hangout da própria Google.

No cenário nacional, um aplicativo que merece destaque é o ZapZap (ZAPZAP, 2015). Ele foi desenvolvido pelo analista de sistemas Erick Costa e tem como um dos diferenciais, permitir que o usuário compartilhe qualquer tipo de arquivo salvo no celular ou no computador como fotos, vídeos, jogos, músicas e etc. Diferente do WhatsApp, que se limita apenas em vídeos, imagens, musicas, localização do usuário e os contatos.

Nesse aplicativo também existe a opção de grupos para conhecer novas pessoas, além da garantia de manter o serviço de forma totalmente gratuita. Atualmente está disponível apenas para a plataforma Android, na Figura 2 são demonstradas algumas telas do aplicativo.

Figura 2 - ZapZap

Fonte: PINHEIRO, 2014.

2.1 ANDROID

O Android é um sistema operacional móvel que é executado sobre o núcleo Linux. Sua história inicia em 2005, quando a Google adquiriu a empresa Android Inc., que desenvolvia sistemas para celulares, três anos depois, no dia 22 de outubro de 2008, foi lançado o HTC Dream, o primeiro dispositivo a contar com este sistema operacional móvel.

Desde então, o sistema não para de crescer, no ano seguinte, em 2009, foram vendidos quase 7 milhões de aparelhos, logo em seguida veio a consolidação, no ano de 2010, houve um crescimento de mais de 800% nas vendas de aparelhos de várias marcas que contavam com o Android (RAMUSSEM, 2011). A partir daí os números não pararam de subir como pode ser observado no gráfico exibido na Figura 3, assim como os aplicativos disponíveis para a plataforma, que no fim de 2012, de acordo com o site G1, somavam mais de 700 mil disponíveis na loja virtual Google Play.

Figura 3 - Crescimento Estimado de Vendas de Aparelhos com Android
Fonte: RASMUSSEN, 2011.

Uma pesquisa realizada pela empresa de consultoria Gartner, aponta que em janeiro de 2014 o sistema operacional Android no Brasil, representou 85,1% das vendas de aparelhos (GUIMARÃES, 2013).

Alguns dos motivos para sua popularização pode estar no fato de que o Android permite aos desenvolvedores escreverem software na linguagem de programação Java controlando o dispositivo via bibliotecas desenvolvidas pela Google, como também as APIs (*Application Programming Interface*) que são bastante fáceis, além da excelente documentação fornecidas pela Google.

A atualmente conhecida Google Play, que já se chamou Android Market, é a loja virtual de aplicativos para Android, onde qualquer desenvolvedor pode colocar seu APP (*Application*) para download, seja de forma gratuita ou paga, sendo necessário apenas pagar uma taxa a Google pela criação da conta e uma participação de 15% na venda dos seus softwares.

O Google Play tem uma política que facilita a distribuição de aplicativos pelo desenvolvedor, ao contrário da loja virtual da Apple, a AppleStore com softwares para o sistema de dispositivos móveis IOS, onde os aplicativos necessitam passar pela aprovação da empresa para que possa ser disponibilizado em sua loja virtual.

A vantagem do Google Play é que se torna muito simples para o desenvolvedor distribuir seu software e ainda poder lucrar imediatamente com isso, porém o grande problema

são os malwares e os sistemas de baixa qualidade, seja referente a interface visual ou de funcionalidade.

O desenvolvimento de aplicativos para a tecnologia Android é um grande mercado de negócio e está em plena expansão, novas versões são lançadas constantemente, e grandes inovações e funcionalidades são incrementadas além é claro, dos recursos dos smartphones, cada vez com mais capacidade de processamento, armazenamento, conexões, etc., permitindo que os softwares possuam cada vez mais recursos e funcionalidades, que auxiliam e muito o cotidiano das pessoas.

2.2 WEB SERVICES REST E SPRING FRAMEWORK

REST significa *Representational State Transfer* (ou Transferência de Estado Representativo, em tradução livre), e é um estilo de desenvolvimento de Web services que teve origem na tese de doutorado de Roy Fielding. Este, que é coautor de um dos protocolos mais utilizados no mundo, o HTTP (*HyperText Transfer Protocol*) (AZEVEDO, 2009). Portanto o protocolo REST utiliza como base, o que seriam as boas práticas de uso de HTTP:

- Uso adequado dos métodos HTTP;
- Uso adequado de URL's (*Uniform Resource Locator*);
- Uso de códigos de status padronizados para representação de sucessos ou falhas;
- Uso adequado de cabeçalhos HTTP;
- Interligações entre vários recursos diferentes.

A Figura 4 demonstra um exemplo da utilização dos padrões de URL's do protocolo REST onde, na URL 1 irá retornar todos os clientes e na URL 2 apenas o cliente 1.

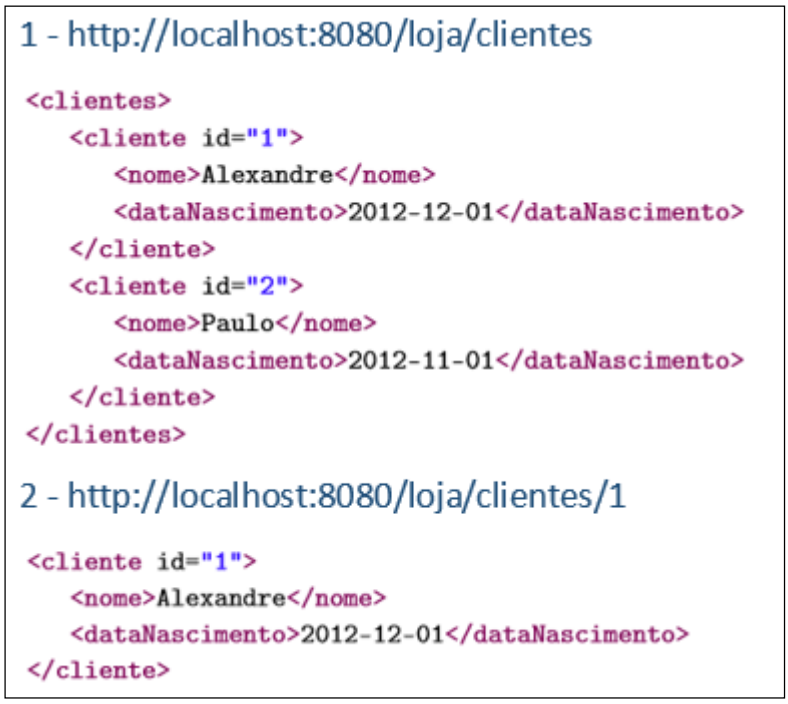

Figura 4 - Exemplo de REST

Diferente do SOAP (*Simple Object Access Protocol*), que é um protocolo de transferência de mensagens em formato XML (*Extensible Markup Language*) para uso em ambientes distribuídos, onde geralmente usa-se o WSDL (*Web Services Description Language*) na descrição da estrutura das mensagens e nas ações possíveis, o REST não impõe restrições ao formato da mensagem, apenas no comportamento dos componentes envolvidos, sendo então essa flexibilidade, uma das maiores vantagens desse protocolo.

O framework de código aberto Spring, fornece suporte para a criação de aplicações REST. Com ele, é possível implementar de forma ágil e rápida Web services REST utilizando algumas anotações que atribuem comportamentos a classes e métodos, como o endereço para acessar via URL ou como definir uma classe como controladora, isso será melhor explicado a seguir. O Spring também fornece bibliotecas para o lado do cliente, através de uma API chamada RestTemplate, que fornece implementações que facilitam a montagem das requisições HTTP aos serviços (FONTOURA, 2013).

Para declarar um serviço é necessário anotar a sua classe com @Controller. As anotações @RequestMapping devem ser utilizadas nos métodos que irão tratar as requisições dos clientes e os atributos value e method são utilizados respectivamente para definir o caminho absoluto do serviço e o seu método HTTP. O DispatcherHandler do Spring irá direcionar as requisições para os métodos anotados com @RequestMapping, com base na URL e no método HTTP utilizado pelo cliente.

A anotação @PathVariable deve ser utilizada para obter variáveis da URL, já a anotação @RequestBody é utilizada para obter o conteúdo do corpo da requisição, assim como @ResponseBody é utilizada para associar um valor para o corpo da resposta HTTP. A Figura 5 apresenta o método getAll() de um serviço que utiliza algumas anotações explicadas anteriormente.

```
@Controller
@RequestMapping("/usuario")
public class UsuarioController {
    @Autowired private UsuarioService usuarioService;

    @RequestMapping(value="/lista/{login}/{senha}", method=RequestMethod.GET)
    public UsuarioList getAll(@PathVariable String login,@PathVariable String senha) {
        return usuarioService.getAll(login, senha);
    }
}
```

Figura 5 - Spring Framework

Para o desenvolvimento desde trabalho foi utilizado o módulo Spring Boot e Spring Android que serão apresentados mais à frente.

2.3 SISTEMA SADI

O Sistema Automatizado de Desenvolvimento Infantil – SADI, foi desenvolvido como uma solução para educação infantil, gerenciando e controlando alunos da educação infantil, bem como um acompanhamento pedagógico do indivíduo.

O acesso ao sistema acontece de forma on-line através de um navegador de internet, visto que o mesmo é desenvolvido para utilização Web, tal acesso será possível apenas a usuários cadastrados previamente, através de um login e senha, garantindo a segurança do sistema. A Figura 6 demonstra a tela de login do sistema.

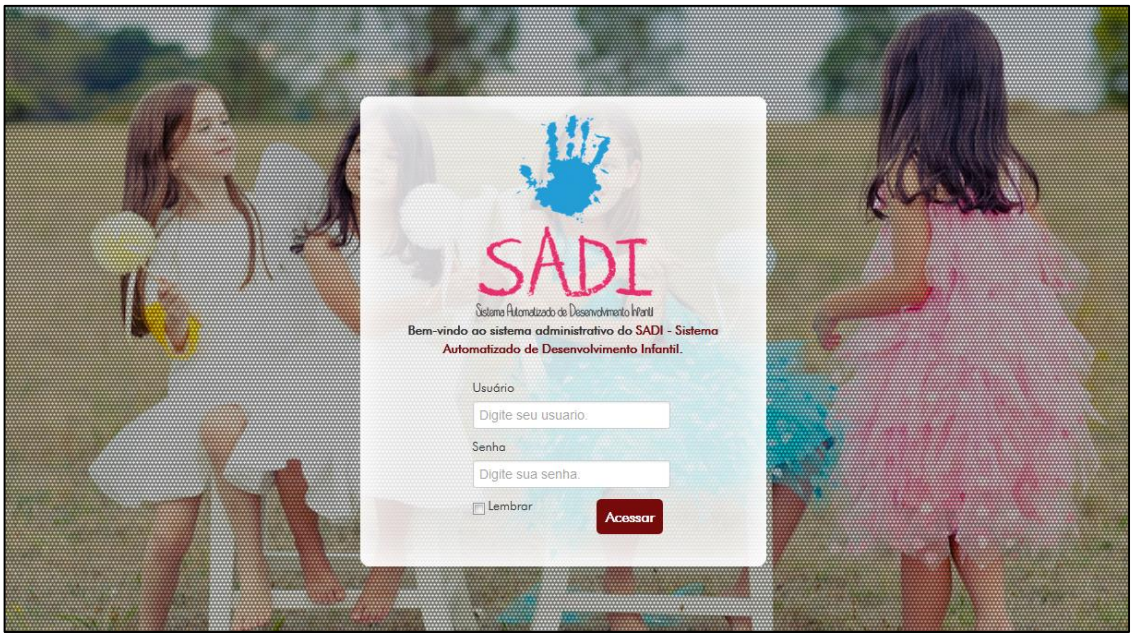

Figura 6 - Login SADI Web

Alguns recursos do sistema são:

- Cadastro dos usuários que poderão ter acesso ao sistema, sendo usuário administrador, professor, pedagogo ou diretor;
- Cadastro das CMEI's – Centro Municipal de Ensino Infantil;
- Cadastro dos alunos e de responsáveis pelos mesmos;
- Cadastro de séries e turmas da instituição, bem como os alunos que fazem parte da turma e professor responsável;
- Cadastro e controle de matrículas, pré-matrículas e rematrículas;
- Cadastro de avaliações pedagógicas e de desempenho dos alunos;
- Controle das atividades realizadas em sala de aula.

A Figura 7 demonstra o menu do sistema com algumas funcionalidades.

Figura 7 - Menu SADI Web

O sistema traz uma proposta de que os profissionais da educação (Professores e

Pedagogos) possam estar avaliando o comportamento do aluno através de um acompanhamento das atividades exercidas diariamente nas CMEI´s, de forma que seja possível obter dados mais concretos sobre o desempenho das crianças no processo educacional.

Buscando uma forma de comunicação entre usuários do sistema, foi desenvolvida a rotina de troca de mensagens, através dessa rotina é possível agilizar a troca de informações entre os usuários, sejam eles profissionais da educação ou responsáveis.

Abaixo, foi definida uma tabela das possibilidades e direitos de envio de mensagem de cada usuário, de forma que não seja possível enviar mensagem para qualquer usuário, visto que não há necessidade por exemplo, dos Responsáveis enviar mensagens aos Administradores ou a Professores que não sejam de seus filhos, como mostra a Quadro 1.

Usuário	Pode enviar para:
Administrador	- Todos os usuários
Diretor	- Administrador da Instituição - Professores da Instituição - Pedagogos da Instituição - Responsáveis da Instituição - Diretores de outras Instituições
Professor	- Administrador da Instituição - Diretor da Instituição - Professores da Instituição - Responsáveis dos alunos da sua turma - Pedagogos da Instituição
Pedagogo	- Administrador da Instituição - Diretor da Instituição - Professores da Instituição - Responsáveis da Instituição - Pedagogos de outras Instituições.
Responsáveis (Pais)	- Diretor da Instituição - Professores dos seus responsáveis - Pedagogo do seu responsável

Quadro 1 - Permissões de Usuários

Pode-se observar na Figura 8 a tela de envio de mensagem, no campo de destinatários só são listados os usuários que o emissor da mensagem tem direitos de envio, como citado anteriormente.

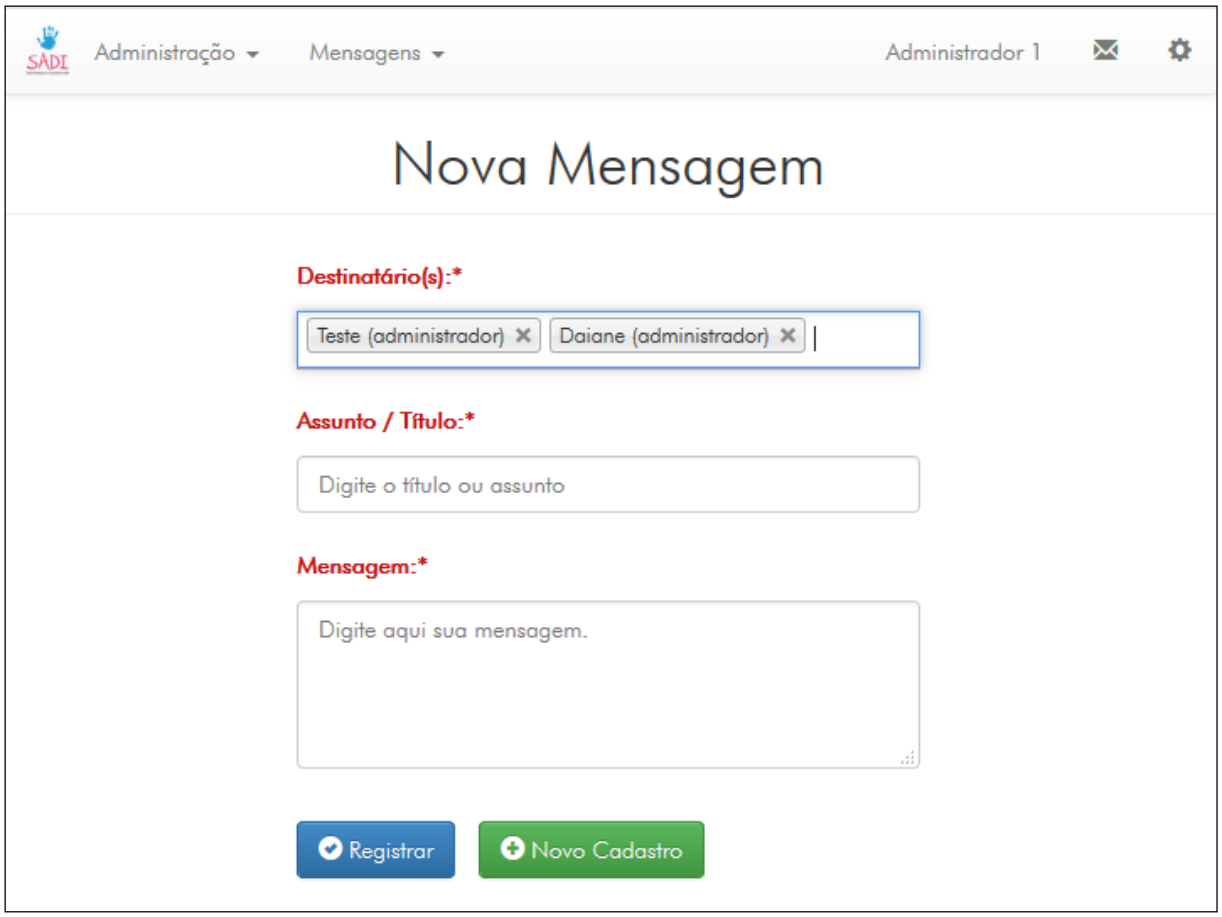

Figura 8 - Tela de Envio de Mensagem

O usuário que recebeu a mensagem, ao acessar o sistema, pode observar uma notificação de nova mensagem no canto superior direito da tela inicial, de acordo com a Figura 9.

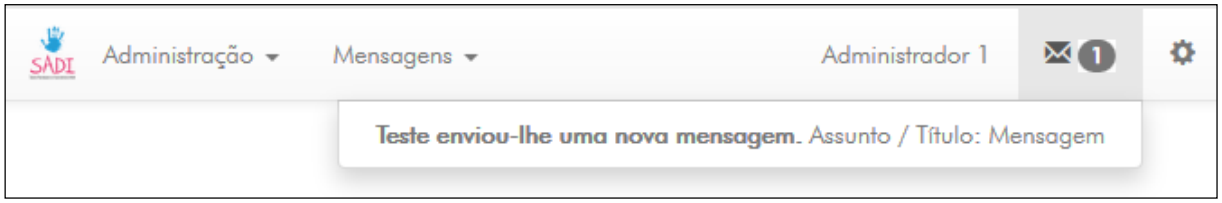

Figura 9 - Notificação de Mensagem SADI Web

Ao abrir a mensagem recebida, o usuário pode responder através do botão identificado, de acordo com a Figura 10.

Figura 10 - Tela de Recebimento de Mensagem

3 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais e o método utilizados na realização deste trabalho. Os materiais se referem às tecnologias como linguagens e ferramentas para a modelagem e a implementação do sistema. O método contém as etapas com os principais procedimentos utilizados para o desenvolvimento do sistema, abrangendo do levantamento dos requisitos aos testes.

3.1 MATERIAIS

As ferramentas e as tecnologias utilizadas para as atividades de modelagem, implementação e execução do sistema estão descritas no Quadro 2, a seguir:

Materiais			
Software	**Versão**	**Licença**	**Propriedade**
Java SE	1.7.0_75-b13	Gratuito	Oracle
ADT Bundle	23.0.6.1720515	Gratuito	Google
Android Plataform / API Level	7	Gratuito	Google
SQLite	3.5.9	Gratuito	SQLite Consortium
Android Annotations	2.7.1	Gratuito	Excilys Group
Visual Paradigm Community Edition	11.0	Gratuito	Visual Paradigm
Eclipse EE	4.4.2	Gratuito	Eclipse Foundation
Spring Boot	1.1.8	Gratuito	Pivotal Software
Spring Android	1.0.1	Gratuito	Pivotal Software
Google Cloud Messaging	1.0.2	Gratuito	Google
MySql	5.5.8	Gratuito	Orcale
Lombok	1.14.8	Gratuito	The Project Lombok Authors

Quadro 2 – Materiais

3.1.1 Java SE

O Java SE, ou Java *Standart Edition*, é uma ferramenta de desenvolvimento para a plataforma Java (ORACLE, 2015). Contém todo o ambiente necessário para a criação e

execução de aplicações Java, incluindo a máquina virtual Java ou JVM (*Java Virtual Machine*), o compilador e as APIs além de outras ferramentas utilitárias.

Está disponível para download no site da Oracle, empresa que gerencia a linguagem atualmente. A Figura 11 mostra a tela sobre a versão do Java.

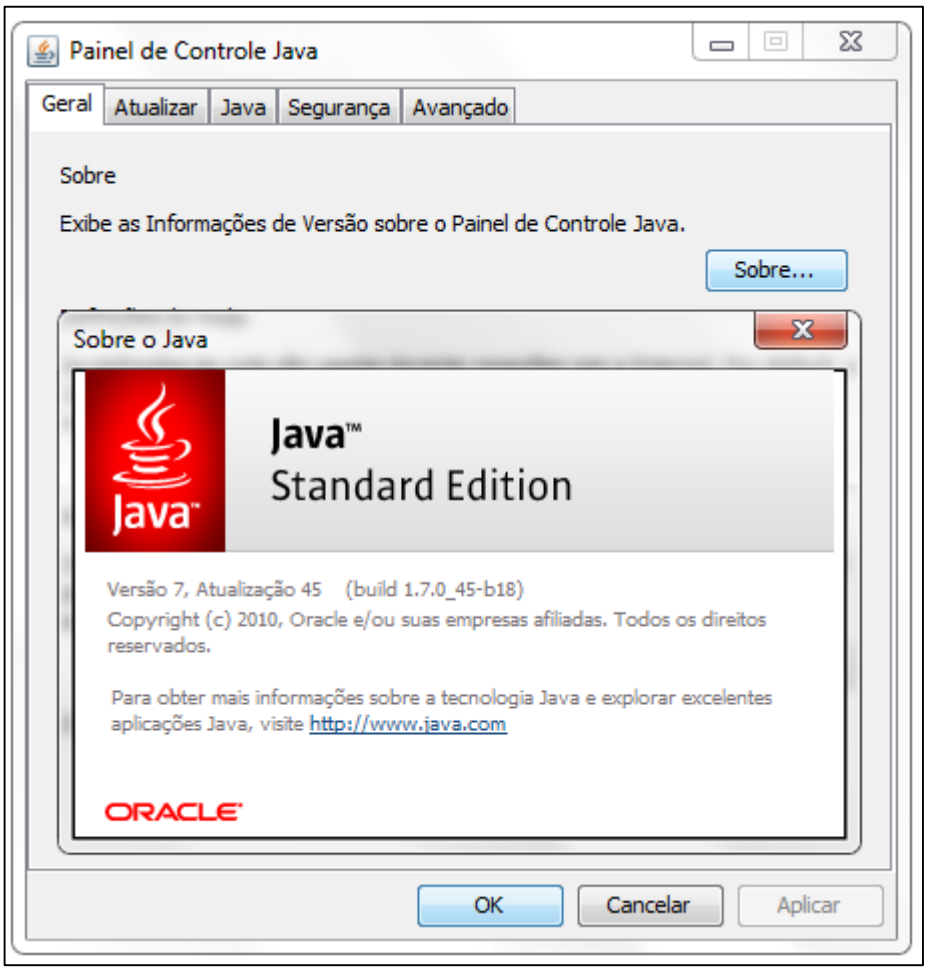

Figura 11 - Java SE

Como o desenvolvimento para a plataforma Android exige uma IDE (*Integrated Development Environment*) de desenvolvimento Java, assim como um ambiente de desenvolvimento, estes precisam da máquina virtual Java para executar, por esse motivo o Java SE deve ser instalado na máquina de desenvolvimento.

3.1.2 ADT Bundle

O ADT Bundle, ou Android Development Tools (DEVELOPER, 2015) foi criado pela Google para fornecer aos desenvolvedores um ambiente integrado para o desenvolvimento de aplicações para Android.

Está disponível de forma gratuita para download através do site (http://developer.android.com), já de forma integrada com a IDE Eclipse e com o Android SDK

(*Software Development Kit*), mas também pode ser adquirido na forma de plugin para instalação no IDE Eclipse.

Um dos componentes principais do ADT é a máquina virtual Android, ou AVD (*Android Virtual Device*), Figura 12, que permite emular o sistema operacional em todas as versões disponíveis, estas que podem ser observadas na Tabela 3, definindo inclusive modelos de aparelhos e configurações de hardware, como o tamanho da tela por exemplo. As AVDs são utilizadas para testar as aplicações desenvolvidas sem a necessidade de um aparelho real.

Figura 12 - AVD

O Android SDK, que compõe o ADT Bundle, é desenvolvido pela Google. Este fornece as bibliotecas das APIs e ferramentas de desenvolvimento necessárias para codificar, testar e depurar aplicativos para o sistema Android.

Ele é composto por pacotes modulares, que contemplam as APIs (*Application Programming Interface*), imagens do sistema para emular, documentação, etc. que podem ser

baixados separadamente usando o gerenciador de SDK Android. Quando as ferramentas do SDK são atualizadas ou uma nova versão da plataforma Android é lançada, é disponibilizado para baixá-los para o ambiente de desenvolvimento.

Existem vários pacotes diferentes disponíveis para o Android SDK como é apresentado na Figura 13.

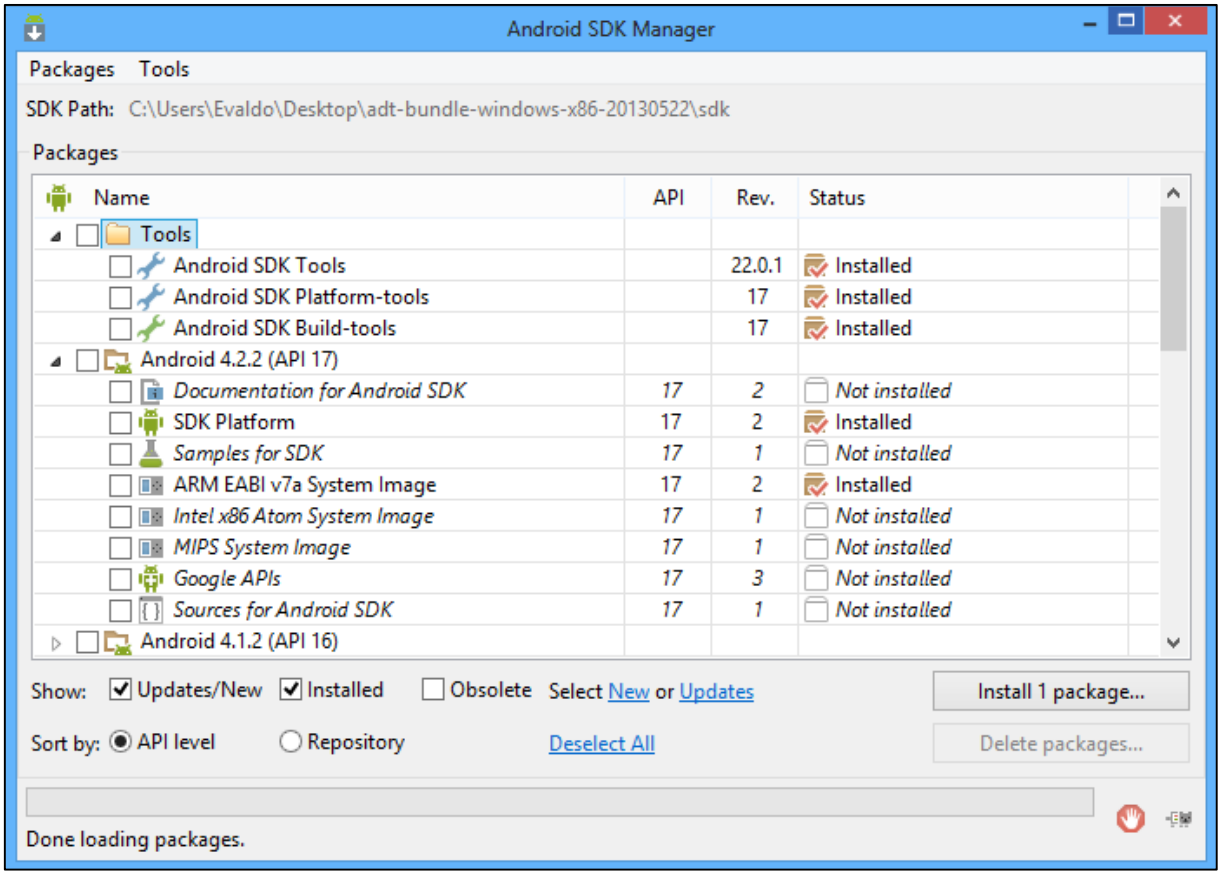

Figura 13 - Android SDK Manager

Cada plataforma Android tem um código para identificação, este código por sua vez, é chamado de API Level. Assim, quando são realizadas atualizações na plataforma, é gerado um novo API Level. De forma resumida, esta API são as versões do Android que existem no mercado, e quando se desenvolve, deve-se escolher a melhor opção para a versão da plataforma, para atender os dispositivos alvos.

É importante para os desenvolvedores entenderem isso, pois quando se cria um projeto é preciso definir qual a plataforma alvo. Um programa criado com a API Level 8 não deverá executar em dispositivos com APIs menores que essa.

Atualmente a última versão da API do Google é a 22, correspondente a versão 5.1.1 do sistema operacional Android, a versão utilizada para o desenvolvimento deste trabalho é a

14, correspondente ao Android 4.0, justamente por oferecer compatibilidade com as versões superiores garantindo que o software desenvolvido possa ser executado na maioria dos aparelhos com o sistema. Na Tabela 1 pode-se observar o histórico das versões do Android.

Versões do Android e seus Codinomes			
Versão do Android	**API level**	**Codinome**	**Ano**
1.0	1	Sem codinome	2008
1.1	2	Sem codinome	2009
1.5	3	Cupcake	2009
1.6	4	Donut	2009
2.0	5	Eclair	2009
2.0.1	6	Eclair	2009
2.1	7	Eclair	2010
2.2	8	Froyo	2010
2.3	9	Gingerbread	2010
2.3.3	10	Gingerbread	2011
3.0	11	Honeycomb	2011
3.1	12	Honeycomb	2011
3.2	13	Honeycomb	2011
4.0	14	Ice Cream Sandwich	2011
4.0.3	15	Ice Cream Sandwich	2011
4.1	16	Jelly Bean	2012
4.2	17	Jelly Bean	2012
4.3	18	Jelly Bean	2013
4.4	19	KitKat	2013
5.0	20	Lollipop	2014
5.0.1	21	Lollipop	2015
5.1.1	22	Lollipop	2015

Quadro 3 - Versões do Android

3.1.3 SQLite

O sistema gerenciador de banco SQLite (2015), padrão da plataforma Android, é uma biblioteca em linguagem C que implementa um banco de dados SQL (*Structured Query Language*) embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo SGBD (Sistema de Gerenciamento de Banco de Dados) separado.

A biblioteca SQLite lê e escreve diretamente no arquivo do banco de dados no disco, ou no caso, em um cartão SD (*Secure Digital*). Seu uso é recomendado onde há simplicidade da administração, implementação e manutenção são mais importantes que recursos avançados e complexos de SGBDs.

Mesmo todos os recursos ativados, o tamanho da biblioteca pode ser inferior a 500KB, dependendo da plataforma de destino e as configurações de otimização do compilador, por isso

o SQLite é uma escolha de banco de dados popular na memória de dispositivos limitados, como celulares, PDAs (*Personal Digital Assistants*) e MP3 players. O desempenho é satisfatório mesmo em ambientes com pouca memória.

Atualmente é mantida por várias empresas parceiras, como Oracle, Adobe, Mozilla e Nokia entre outras, e disponibilizado de forma gratuita para qualquer fim, seja particular ou comercial.

3.1.4 Android Annotations

Android Annotations (2015) é um *framework* de código aberto e gratuito, que tem por objetivo acelerar o desenvolvimento de aplicações Android, simplificando o código e facilitando a manutenção.

Ao incluir essa biblioteca ao projeto Android, é possível utilizar anotações que serão compiladas em tempo de execução, gerando o código referente, agilizando muito o desenvolvimento como demonstra a Figura 14.

Sem annotations

```
public class MyClass extends Activity {

private EditText edit;
private Button button;

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

 edit = (EditText) findViewById(R.id.edit);
 button = (Button) findViewById(R.id.button);
}
}
```

Com annotations

```
@EActivity(R.layout.meu_layout)
public class MyClass extends Activity {

@ViewById
private EditText edit;

@ViewById
private Button button;

}
```

Figura 14 - Comparação entre código com e sem Android Annotations
Fonte: GitHub, 2015.

Uma das anotações mais utilizadas é @Backgroud, que irá executar o método anotado em uma nova *thread* no sistema operacional, muito importante no Android para grandes processamentos de dados sem bloquear a interface gráfica.

Mas existem algumas outras, como @AfterViews, que executa o método seguinte na criação da activity, logo após o método OnCreate, padrão do Android, a @ViewById permite vincular um elemento do XML da tela da activity á um objeto declarado na classe Java, basta que o ID (*Identity*) seja igual ao nome do objeto ou caso sejam diferentes, adicionar entre parênteses o nome do ID do XML, por exemplo @ViewById(R.id.IDdoComponente).

No site do projeto, androidannotations.org, é possível obter toda a documentação para a utilização dos recursos da biblioteca.

3.1.5 Visual Paradigm Community Edition

Visual Paradigm for UML (*Unified Modeling Language*) é uma ferramenta de modelagem para diagramas UML. Essa ferramenta fornece suporte para gerenciamento de casos de uso, classes, diagrama de requisitos SysML (*Systems Modeling Language*) e projeto de banco de dados com diagrama de entidades e relacionamentos organizando os fluxos de trabalho, além de gerar documentação (VISUAL PARADIGM, 2013).

Na Figura 15 pode ser observada a tela inicial do software, que para este trabalho foi utilizada a versão não comercial.

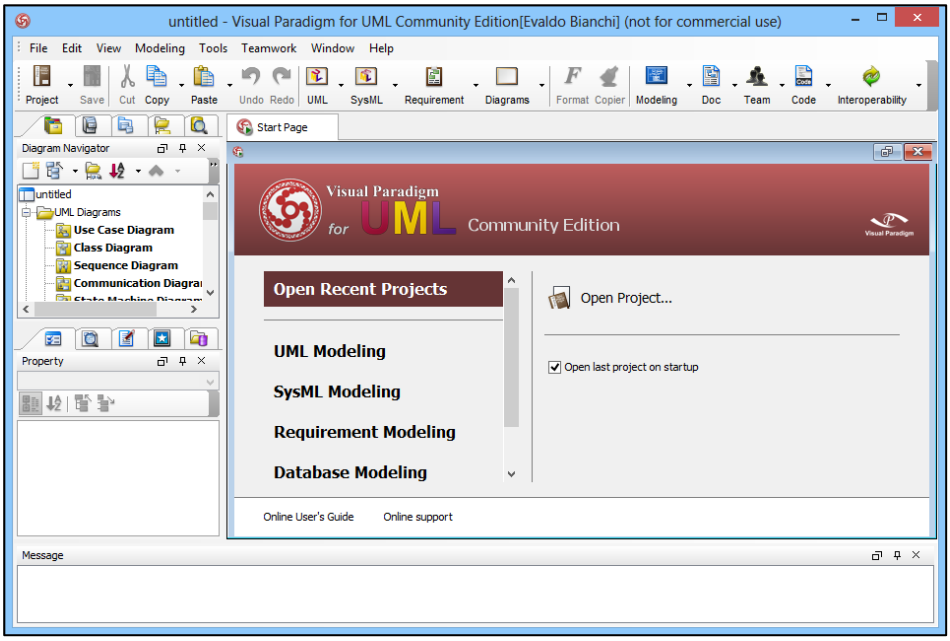

Figura 15 - Visual Paradigm UML Community Edition

3.1.6 Eclipse EE

O IDE Eclipse (2013), é desenvolvido em Java, e segue o modelo *open source* de desenvolvimento. Foi iniciado pela empresa IBM que após desenvolver a primeira versão do produto, doou como software livre para a comunidade.

Este IDE pode ser adquirido de forma gratuita diretamente no site oficial do projeto (http://www.eclipse.org) neste trabalho foi utilizada na versão 4.4.2.

Atualmente, este é um dos IDEs mais utilizados no mundo, possui amplo suporte e diversos plugins, como o ADT para o desenvolvimento Android, além de uma interface

funcional como demonstrado na Figura 16.

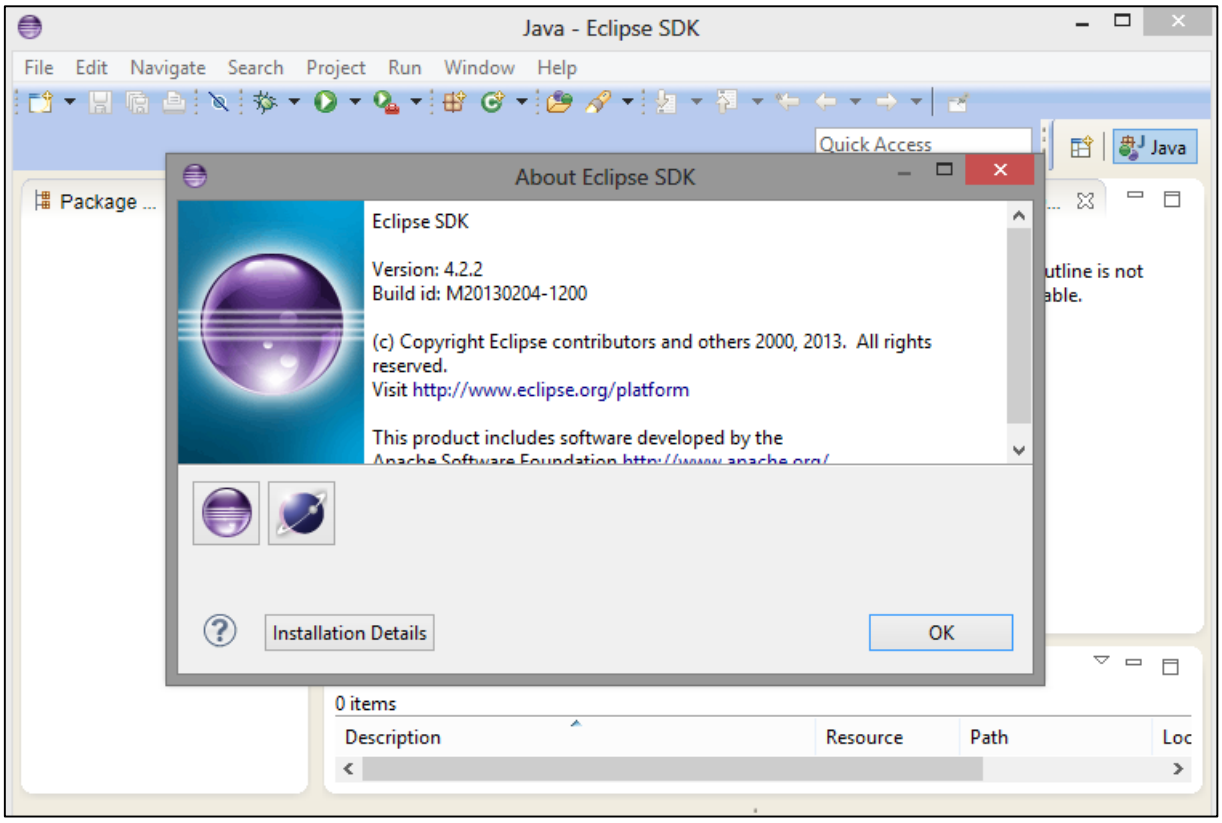

Figura 16 - Eclipse IDE

3.1.7 Spring Boot

Trata-se de um framework que tem como objetivo reaproveitar as tecnologias já existentes no Spring Framework e aumentar a produtividade do desenvolvedor. Introduzido na versão 4.0 do Spring, surgiu como necessidade em vista das críticas sobre a dificuldade e o tempo necessário para se iniciar o desenvolvimento de novos projetos.

A principal diferença se dá no modo de configurar, organizar o código e executar a aplicação, baseado em 4 princípios:

1. Prover uma experiência de início de projeto extremamente rápida e direta;

2. Apresentar uma visão simples sobre o modo como deve-se configurar os projetos Spring, e ao mesmo tempo flexível o suficiente para que possa ser facilmente substituída de acordo com os requisitos do projeto;

3. Fornecer uma série de requisitos não funcionais já pré-configurados para o desenvolvedor como, por exemplo, métricas, segurança, acesso a base de dados, servidor de aplicações/servlet embarcado, etc.;

4. Não prover nenhuma geração de código e minimizar a zero a necessidade de arquivos XML.

De modo resumido, o Spring boot pode ser entendido como uma fina camada sobre tecnologias já consagradas pelo mercado, tal como pode-se verificar na Figura 17. A grande mudança está no modo como se empacota e se acessa estas soluções.

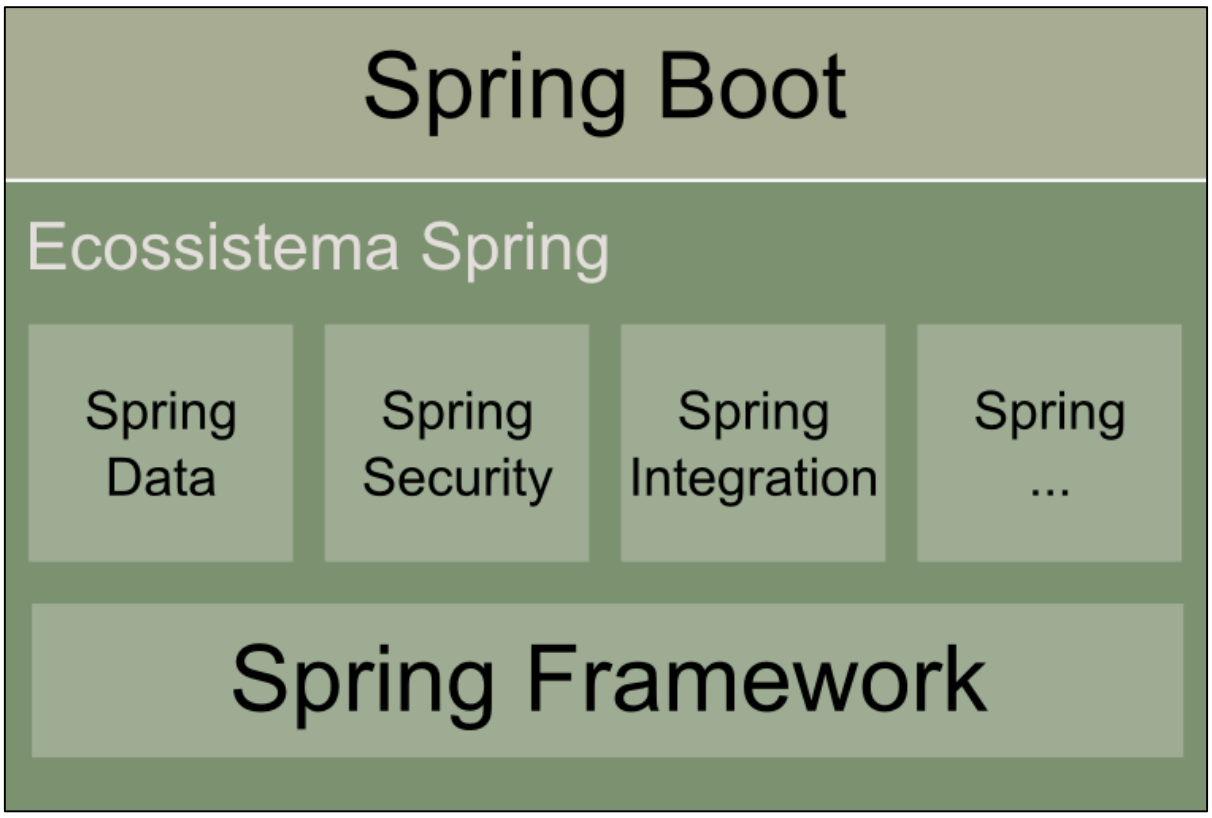

Figura 17 - Posicionamento do Spring Boot no ecossistema Spring
Fonte: WEISSMANN, 2014.

O conceito de convenção sobre configuração é o grande motor por trás do ganho de produtividade do Spring Boot, ou seja, a maior parte das configurações que o desenvolvedor precisa escrever no início de um projeto são sempre as mesmas, dessa forma, o Spring Boot já inicia novo projeto com todas estas configurações definidas podendo ser alteradas a qualquer momento.

Para o desenvolvimento desta aplicação, foi criado um projeto Spring Boot utilizando o Maven, este que é o responsável pelo gerenciamento de *builds*. A partir disso, utilizando-se uma dependência do Spring, foi obtido tudo que é necessário para o desenvolvimento do servidor como o Hibernate para o mapeamento das tabelas do banco de dados e um servidor de aplicação que é embutido no empacotamento da aplicação, neste caso o TomCat.

3.1.8 Spring Android

O framework Spring para Android, fornece ferramentas para facilitar a comunicação via HTTP com o padrão REST além de recursos de autenticação para acesso a APIs.

3.1.9 Google Cloud Messaging

O Mensagens do Google Cloud ou GCM é um serviço gratuito que ajuda os usuários a enviarem mensagens em várias plataformas: Android, IOS e Google Chrome. Por exemplo, um servidor pode enviar mensagens diretamente para um dispositivo, para grupos ou para dispositivos inscritos em tópicos. Além disso, o APP em um dispositivo pode enviar mensagens diretamente para um servidor e para dispositivos que pertençam ao mesmo grupo.

Ou seja, o serviço GCM do Google basicamente facilita o envio de mensagem chamadas PUSH, onde o servidor consegue acordar o cliente para executar uma determinada tarefa.

Para adicionar o GCM ao aplicativo, é necessário acessar o site do Google Developers em “Cloud Messaging” e criar uma chave para o pacote da aplicação, conforme mostra a Figura 18.

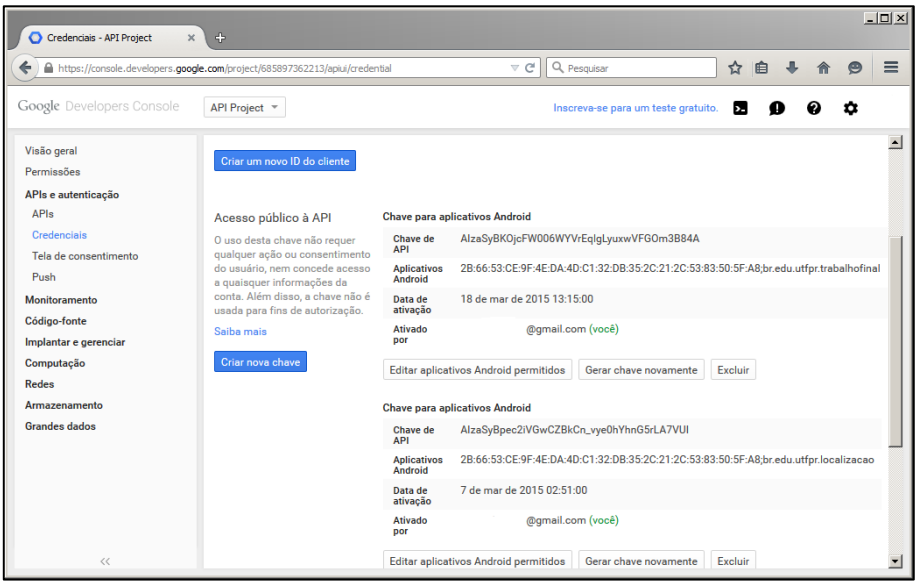

Figura 18 - Google Developers Console

3.1.10 MySql

O MySQL é uma ferramenta que gerencia bancos de dados de forma relacional, utiliza código aberto e é usado na maioria das aplicações gratuitas para gerenciar suas bases de dados

(MYSQL, 2013). MySQL utiliza a linguagem SQL para inserir, acessar e gerenciar o conteúdo armazenado num banco de dados (PISA, 2012).

3.1.11 Lombok

O desenvolvimento em Java envolve escrever um monte de código que as vezes pode ser encaixado como "clichê". Como por exemplo, criar um objeto de valor (ou "objeto de transferência de dados") e é possível ter um comportamento desse tipo pelo menos com uma dúzia de propriedades em uma classe. Essas doze declarações são a única coisa importante na classe. Depois de criar uma centena de linhas de getters, setters (se não imutável), de equals/hashCode e um método toString, possivelmente todos gerados pela IDE.

Este é um processo sujeito a erros, mesmo quando gerado por uma IDE. Uma IDE não irá dizer se, por exemplo, adicionar outro membro de dados e não se regenerarem seus métodos equals e hashCode.

O projeto Lombok visa reduzir muito a necessidade de tais "clichês" usando anotações para gerar tais métodos automaticamente.

Atualmente, algumas das principais anotações do Lombok possui as anotações listadas na Quadro 4, exibida a seguir.

Principais anotações do Lombok	
@Getter / @Setter	Gera os métodos get e set de todos os atributos, ou em atributos individuais
@ToString	Gera o método toString com todos os atributos
@EqualsAndHashCode	Gera os métodos equals e hashCode
@NoArgsConstructor, @RequiredArgsConstructor e @AllArgsConstructor	Anotações para gerar o construtor sem parâmetros, com parâmetros específicos e com todos os parâmetros
@Data	Gera os métodos get e set, toString e equals e hashCode além do construtor com parâmetros
@Builder	Gera o padrão builder para a classe

Quadro 4 - Principais anotações do Lombok

Usar o Lombok com Maven é simples, a documentação não é extensa e há exemplos de todas as anotações. O projeto gera os métodos toString(), hashCode() e equals() automaticamente, mas ao contrário de algumas bibliotecas, ele não faz isso através da reflexão em tempo de execução, mas sim em tempo de compilação, não impactando no desempenho da aplicação.

3.2 MÉTODO

Para o desenvolvimento deste trabalho, alguns métodos foram utilizados a fim de concretizar o desenvolvimento do mesmo. A seguir serão descritas as etapas para o desenvolvimento juntamente com as principais atividades.

I) Definição inicial do escopo do sistema a ser desenvolvido: após a análise dos softwares semelhantes existentes, foi definido o funcionamento do sistema, quais regras e métodos seriam seguidos.

II) Definição das tecnologias a serem utilizadas: definido quais seriam os recursos e tecnologias necessárias para o desenvolvimento do software.

III) Definição do layout do sistema: nesta etapa, por se tratar de um sistema mobile, onde o principal foco é a interface entre o aplicativo e o usuário, foram definidas como seriam as telas do sistema tanto em termos gráficos como funcionais.

IV) Análise e projeto: definido o fluxo do sistema e a estrutura do banco de dados.

V) Codificação: realização da codificação utilizando o ADT do Google e suas tecnologias nativas, juntamente com as demais tecnologias citadas.

VI) Testes: após todas as implementações, foram realizados os testes em uma máquina virtual executando Android para identificar possíveis falhas no projeto, e também em aparelhos reais.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta o sistema SADI Mobile implementado como resultado do desenvolvimento deste trabalho. Inicialmente o sistema é apresentado e em seguida está a sua modelagem. Na Seção 4.3 uma das telas do sistema é apresentada, com o objetivo de mostrar as funcionalidades e a forma de interação com o sistema e na Seção 4.4 estão os exemplos da codificação gerada.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema SADI Mobile, desenvolvido como resultado deste estudo, tem como finalidade básica facilitar o envio e recebimento de mensagens entre os usuários do sistema SADI Web, para isso, foi dividido em dois projetos principais, o Servidor, que irá conectar com o banco de dados oficial e gerenciar o envio e recebimento de mensagens, assim como as notificações, e o aplicativo mobile, que se comunica com o servidor e estará disponível para os usuários trocarem mensagens. A Figura 19 demonstra a visão geral do funcionamento do sistema.

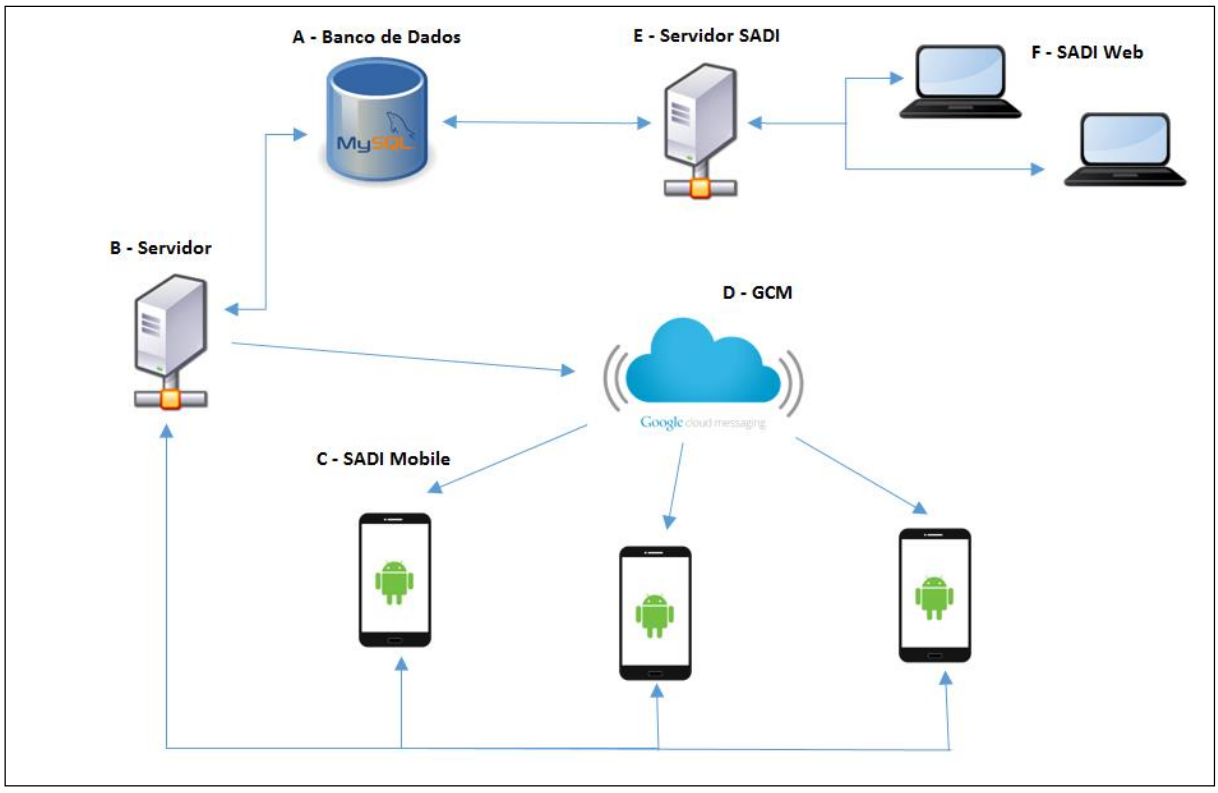

Figura 19 - Funcionamento geral do Sistema SADI Mobile

Na Figura 19 tem-se:

A) Banco de Dados: representa o banco de dados MySql do sistema SADI Web, onde estão todas as tabelas do sistema, inclusive referentes as mensagens enviadas e recebidas entre os usuários;

B) Servidor: desenvolvido neste trabalho, o servidor conecta-se diretamente ao banco de dados para ler e registrar as mensagens que serão transportadas entre os aplicativos Web e mobile;

C) SADI Mobile: aplicativo também desenvolvido neste trabalho, representa a utilização dos usuários do sistema da plataforma, para o envio e recebimento das mensagens;

D) GCM: o sistema do Google Cloud Messaging, irá receber as mensagens de notificação do servidor, e encaminhar individualmente para cada usuário registrado no sistema, este por sua vez irá buscar as mensagens no servidor;

E) Servidor SADI: sistema já existente, o servidor SADI executa em na linguagem php e fornece acesso ao SADI Web além de conectar-se ao bando de dados MySql;

F) SADI Web: apresenta os usuários da plataforma Web que estarão utilizando todas as funções do sistema e diretamente ligados com as aplicações mobile.

4.2 MODELAGEM DO SISTEMA

Para o desenvolvimento do sistema foram definidos alguns requisitos funcionais e não funcionais, estes listados nas Tabelas 5 e 6, respectivamente.

Identificação	Nome	Descrição
RF001	Primeiro acesso	Exibir uma tela de login ao iniciar o aplicativo, o usuário deverá informar seu login e senha já cadastrados no SADI Web.
RF002	Atualizar	Na tela principal, exibir uma opção para atualizar as mensagens.
RF003	Lista de mensagens	Listar todas as mensagens na tela principal, indicando em negrito as não lidas. Ao tocar em uma mensagem, exibi-la em detalhes.
RF004	Excluir mensagem	Permitir que o usuário exclua as mensagens com um toque longo na lista de mensagens da tela principal.
RF005	Nova mensagem	Exibir uma opção de nova mensagem, ao tocar, o usuário será direcionado para a tela de contatos.
RF006	Tela de contatos	Exibir todos os usuários disponíveis para o envio de mensagens, mostrando que tipo de usuário este contato é (administrador, professor, etc.).

RF007	Responder mensagem	Ao exibir os detalhes de uma mensagem recebida, haverá uma opção para responder, direcionando para a tela de envio de mensagem.
RF008	Notificação	Exibir uma notificação para novas mensagens recebidas, ao tocar, a tela principal deverá ser mostrada.
RF009	Enviar mensagem	Exibir a tela de envio de mensagem onde o usuário poderá informar um título e o conteúdo. Ao enviar, direcionar para a tela principal.

Quadro 5 - Listagem dos requisitos funcionais

Identificação	Nome	Descrição
RNF001	Primeiro acesso	Ao realizar o primeiro login pelo aplicativo mobile, registrar o GCM ID para o dispositivo e armazenar no bando de dados. Após isso, baixar todas as mensagens não lidas do servidor.
RNF002	Status da mensagem	Criar um campo no bando de dados para armazenar o status da mensagem que serão 3: baixada, notificada ou pendente.
RNF003	Job	Criar um job para verificar as novas mensagens não notificadas. Esse job irá ler o banco de dados e enviar para os usuários uma mensagem indicando que existem mensagens novas no servidor para que então o aplicativo possa baixa-las.
RNF004	Banco de Dados Mobile	O banco de dados deverá ser criado de forma automática, na primeira execução do aplicativo.
RNF005	Banco de Dados Web	Conectar o servidor ao banco de dados MySql do sistema Web.
RNF006	Background	Todas as requisições feitas pelo aplicativo deverão executar em segundo plano.
RNF007	Mensagem direta	Ao enviar uma mensagem do mobile para outro usuário que também possua mobile, a notificação deverá ser enviada imediatamente, não dependendo do Job (RNF003)

Quadro 6 - Listagem dos requisitos não funcionais

Com base nos requisitos funcionais e não funcionais, foram desenvolvidos os diagramas de caso de uso, conforme Figura 20.

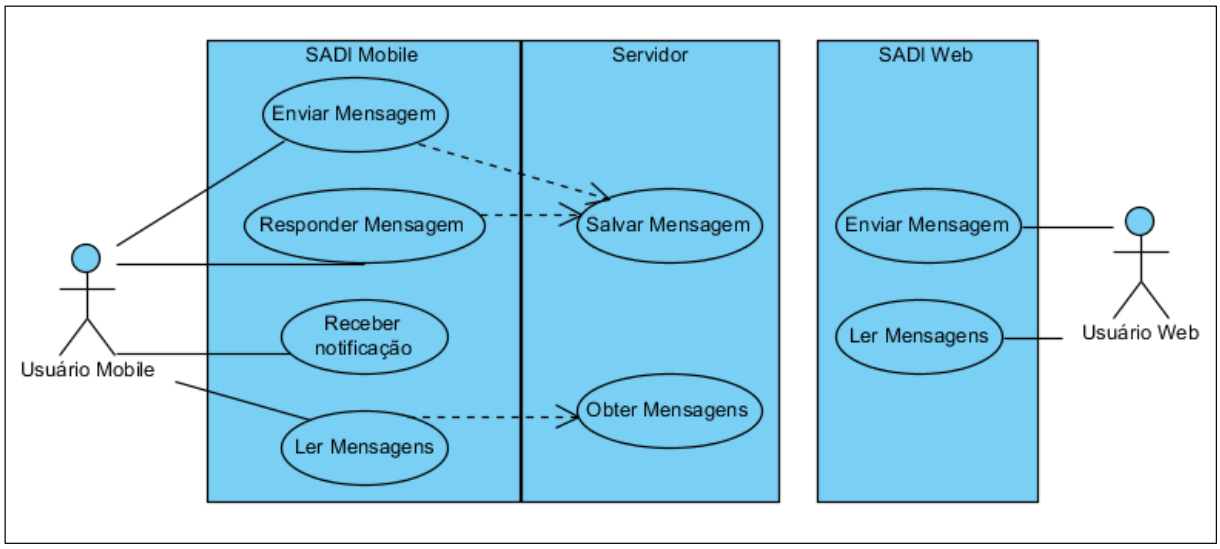

Figura 20 - Diagrama de Caso de Uso

4.2.1 Persistência dos dados

Para o funcionamento do sistema, as informações serão armazenadas em bancos de dados distintos. O sistema Web já possui um banco de dados definido, este é o MySql, onde estão todas as informações do sistema, no mobile por sua vez, o banco de dados é SQlite e possui uma estrutura de tabelas diferente do sistema Web, onde as informações são mais sucintas, cabendo apenas o que é necessário. Ambos os bancos, são descritos a seguir.

4.2.1.1 Persistência SADI Mobile

As informações do usuário do aplicativo, por serem apenas um registro, serão armazenadas no SharedPreferences do Android. Esta é uma técnica alternativa de persistência baseada em arquivo texto. Estas informações serão obtidas no primeiro acesso para que o usuário obtenha os contatos e as mensagens recebidas ainda não lidas. A Tabela 7 demonstra os dados do usuário.

Usuário Shared Preferences	
Parâmetro	**Tipo de dado**
SHARED_USUARIO_LOGIN	Texto
SHARED_USUARIO_ID	Inteiro
SHARED_USUARIO_SENHA	Texto
SHARED_USUARIO_IS_LOGADO	Boleano
SHARED_GCM_ID	Texto

Quadro 7 - Tabela do Usuário

Os contatos, para enviar as mensagens, serão armazenados na tabela usuário, composta

por um identificador que será o mesmo do sistema Web, o tipo (responsável, professor, diretor, pedagogo e administrador) e o login para identificação, conforme descritos na Quadro 8.

USUARIO	
Coluna	**Tipo de dado**
Id	Inteiro
Tipo	Texto
Login	Texto

Quadro 8 - Tabela de Contatos

As mensagens do aplicativo, tanto enviadas como recebidas, serão armazenadas na tabela mensagem, esta, que dentre as colunas, possui um identificador local, que será incremental, o identificador do servidor, que indica o id da mensagem no servidor. O Quadro 9 representa a tabela mensagem.

MENSAGEM	
Campo	**Tipo de dado**
Id	Inteiro
IdServer	Inteiro
Tipo	Texto
Lida	Booleano
Usuario	Inteiro
Mensagem	Texto
Titulo	Texto
Data	Numérico

Quadro 9 - Tabela de Mensagens

4.2.1.2 Servidor

No servidor, que irá comportar as mensagens e realizar a comunicação entre o aplicativo mobile e o bando de dados Web, as tabelas que foram mapeadas são: mensagem (Quadro 10), mensagem_destinatario (Quadro 11) e usuário (Quadro 12). Todos os campos já existiam, com exceção do campo status na tabela mensagem_destinatario, que terá os valores pendente, notificada ou baixada, e do campo gcmid na tabela usuário, que irá armazenar o código gerado pelo GCM da Google.

MENSAGEM	
Campo	**Tipo de Dado**
Id_mensagem	Inteiro
Id_remetente	Inteiro
Id_resposta	Inteiro
Titulo_mensagem	Texto
Texto_mensagem	Texto
Data_mensagem	Data
Excluida	Enumerado

Quadro 10 - Tabela de Mensagens no servidor

MENSAGEMDESTINATARIO	
Campo	**Tipo de Dado**
Id_mensagem_destinatario	Inteiro
Id_mensagem	Inteiro
Id_destinatario	Inteiro
Lida	Enumerado
Status	Enumerado

Quadro 11 - Tabela de Mensagens por Destinatário

USUARIO	
Campo	**Tipo de Dado**
Id_usuario	Inteiro
Id_acesso	Inteiro
Tipo_usuario	Enumerado
Login_usuario	Texto
Data_cadastro	Data
Senha_usuario	String
Senha_alterada	Enumerado
GcmId	Texto

Quadro **12 -** Tabela de Usuários no servidor

Como mencionado anteriormente, essas tabelas são do banco MySql, utilizado pelo sistema Web. Para o sistema mobile, elas são lidas e as informações extraídas para então serem convertidas e persistidas no SQlite, ou seja, no banco de dados mobile.

4.3 SISTEMA DESENVOLVIDO

O sistema de envio e recebimento de mensagens, SADI Mobile, desenvolvido neste trabalho deve funcionar de forma sincronizada, estendendo as funcionalidades do sistema SADI Web.

O projeto é dividido em 2 sistemas principais, o servidor e o aplicativo móvel, ambos serão descritos na sequência.

4.3.1 Servidor

O aplicativo servidor desenvolvido com o Spring Boot, é o responsável por fazer a comunicação e a interligação entre o sistema SADI Web e SADI Mobile, isso é realizado através do banco de dados. Toda a comunicação com o aplicativo mobile é feito através dos serviços REST que se utilizam do formato JSON (*JavaScript Object Notation*).

As funções que o servidor disponibiliza através de serviços REST são:

- Usuário: login, lista de usuários e registro de GCM;
- Mensagem: envio de mensagens, lista de mensagens, atualização de mensagens.

O sistema possui um Job, que funciona como uma tarefa agendada, para a leitura de mensagens enviadas dentro do sistema Web, em um intervalo de 60 segundos é verificado no banco de dados se existem mensagens que não foram notificadas. A partir disso, uma a uma, as mensagens são selecionadas e uma notificação é enviada para o usuário destino, caso este possua o registro de GCM.

As mensagens enviadas a partir do SADI Mobile, são notificadas automaticamente para o usuário destino, novamente, caso o mesmo possua registro de GCM.

O servidor não possui interface gráfica, ele pode ser executado como um aplicativo desktop, empacotado em um jar ou *Java Archive*, que contém as classes compiladas e metadados relacionados, pois o Spring Boot tem como um dos recursos o empacotamento de aplicações com o servidor TomCat embutido.

Ainda, pode-se gerar um arquivo war, ou seja, um arquivo de aplicação Web que contém páginas, classes, metadados, bibliotecas e diversos arquivos, para implantar em outros servidores de aplicação, como Glassfish ou Wildfly, por exemplo. Essas opções são definidas em um arquivo chamado pom, onde estão as configurações do Maven.

A Figura 21 demonstra o processo básico de início da aplicação do servidor. No item 1, é executada a classe principal do programa, onde estão todas as anotações referentes a

aplicação do SpringBoot, assim como todas as demais classes anotadas que são lidas no item 2, como a classe responsável pelo Job, que é iniciado no item 3. No item 4, é realizado o mapeamento das tabelas do banco de dados, ou seja, o SpringBoot verifica se todas as tabelas e colunas mapeadas existem, caso contrário, são criadas. O item 5 inicia os serviços, que são as classes anotadas com @Service e que detém a camada de regra de negócio e por fim, no item 6 a aplicação está on-line e disponível para receber requisições.

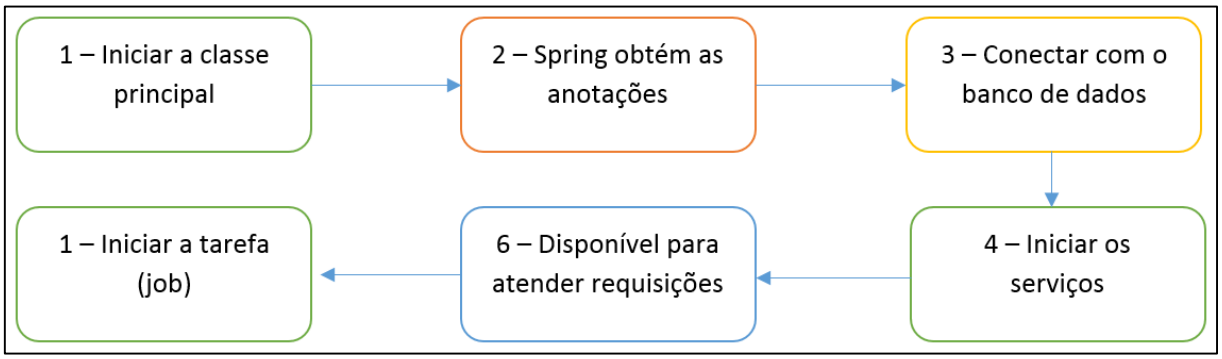

Figura - Inicialização do servidor

4.3.2 SADI Mobile

O aplicativo SADI mobile disponível para os usuários do sistema SADI Web, tem como objetivo facilitar a troca de mensagens entre estes usuários, sendo então, uma extensão das funcionalidades do sistema principal.

No primeiro acesso, após instalar o aplicativo no Android, será exibida a tela de login com os campos de usuário e senha, estes, que são os mesmos utilizados no sistema Web.

Após a validação do login, é realizado o registro do dispositivo no Google Cloud Messaging e realizado o download de todos os contatos e mensagens não lidas para o usuário.

A Figura 22 apresenta três telas do aplicativo, a tela de login na esquerda, no centro o processamento onde é feito o registro no GCM e baixados os contatos e, na imagem da direita, a tela inicial com as mensagens recebidas e não lidas pelo usuário.

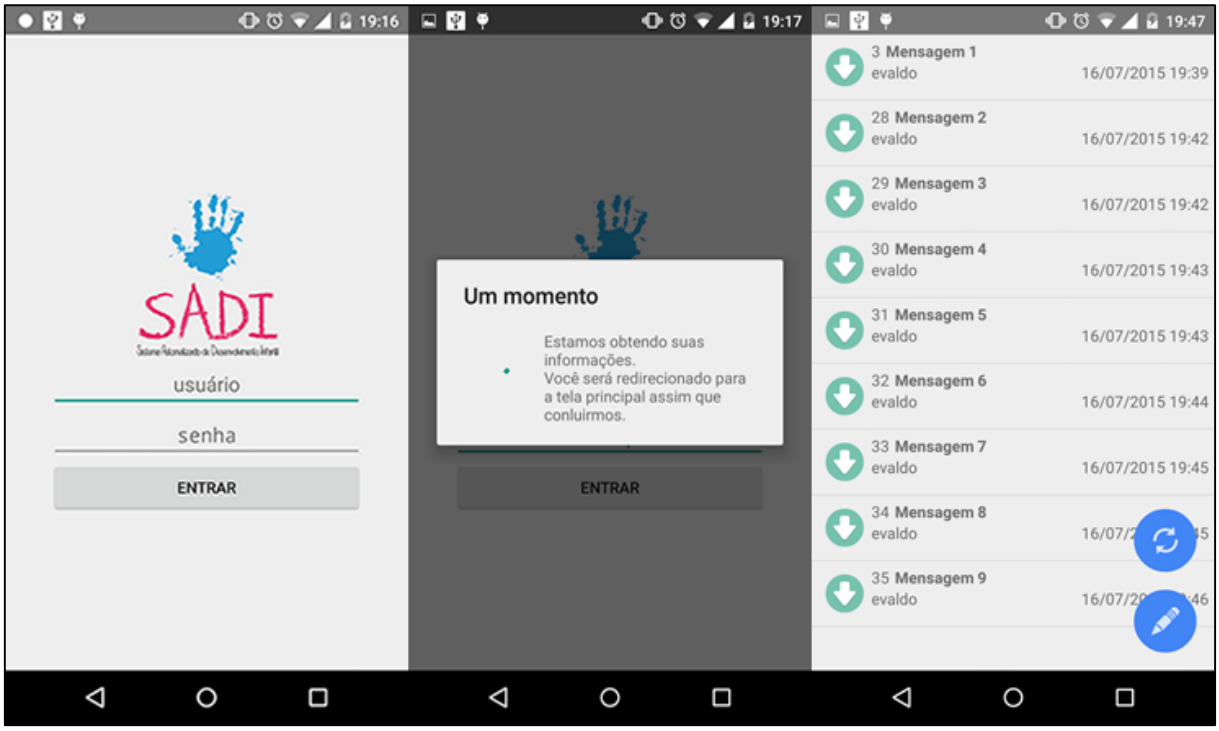

Figura 21 - SADI Mobile (login, primeiro acesso e tela inicial)

Na tela inicial existe ainda um botão para atualizar, que irá receber as novas mensagens do usuário imediatamente, e também um botão que permite escrever novas mensagens.

Ao pressionar o botão para novas mensagens, o usuário é direcionado para a tela de contatos, exibido na Figura 23 (imagem da esquerda), tocando sobre o contato, será exibida a tela para digitar a mensagem, imagem do centro, com o campo de título e conteúdo. Por fim, se o usuário tocar em uma mensagem recebida, como mostra a imagem da direita, ele poderá visualizar a mensagem e um botão para responder também está disponível, este que irá direcionar para a mesma tela de envio de novas mensagens.

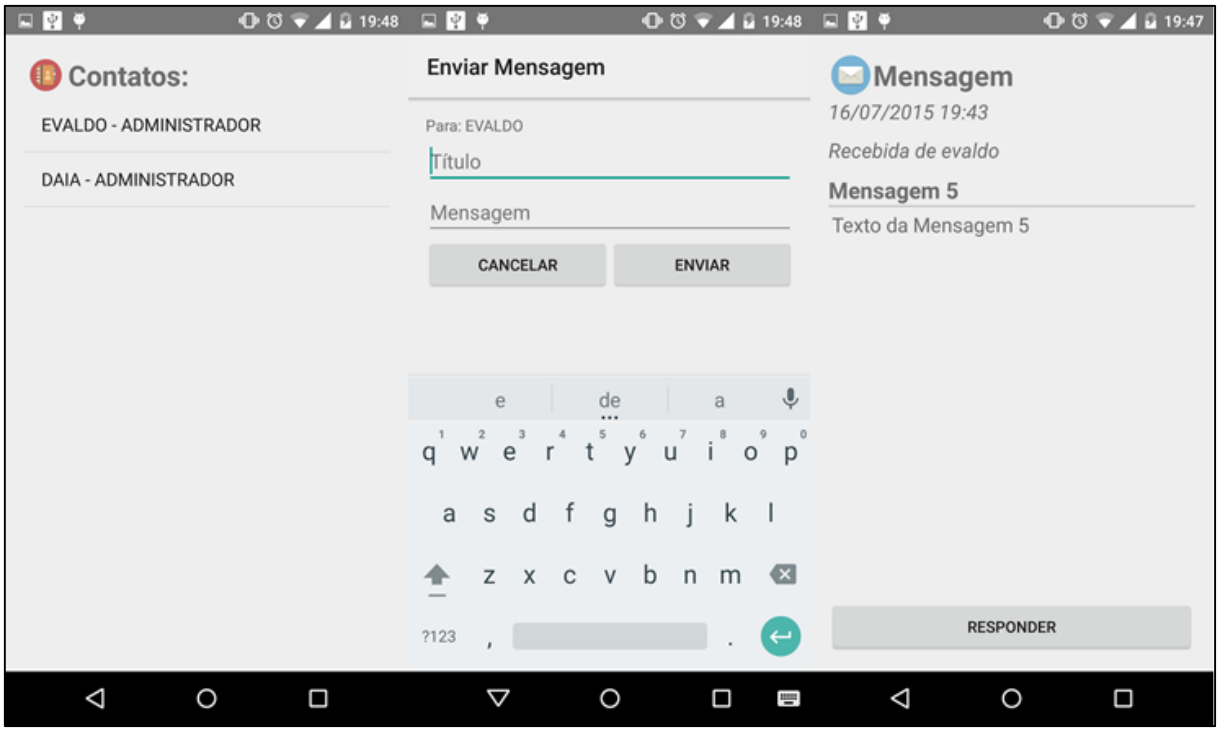

Figura 22 - SADI Mobile (contatos, enviar mensagem e mensagem recebida)

Quando existirem novas mensagens no servidor, uma notificação é enviada para o usuário, como mostra a Figura 24 na imagem da esquerda. Na imagem da direita, é possível observar novamente a lista de mensagens diferenciando as enviadas e recebidas com um ícone de seta.

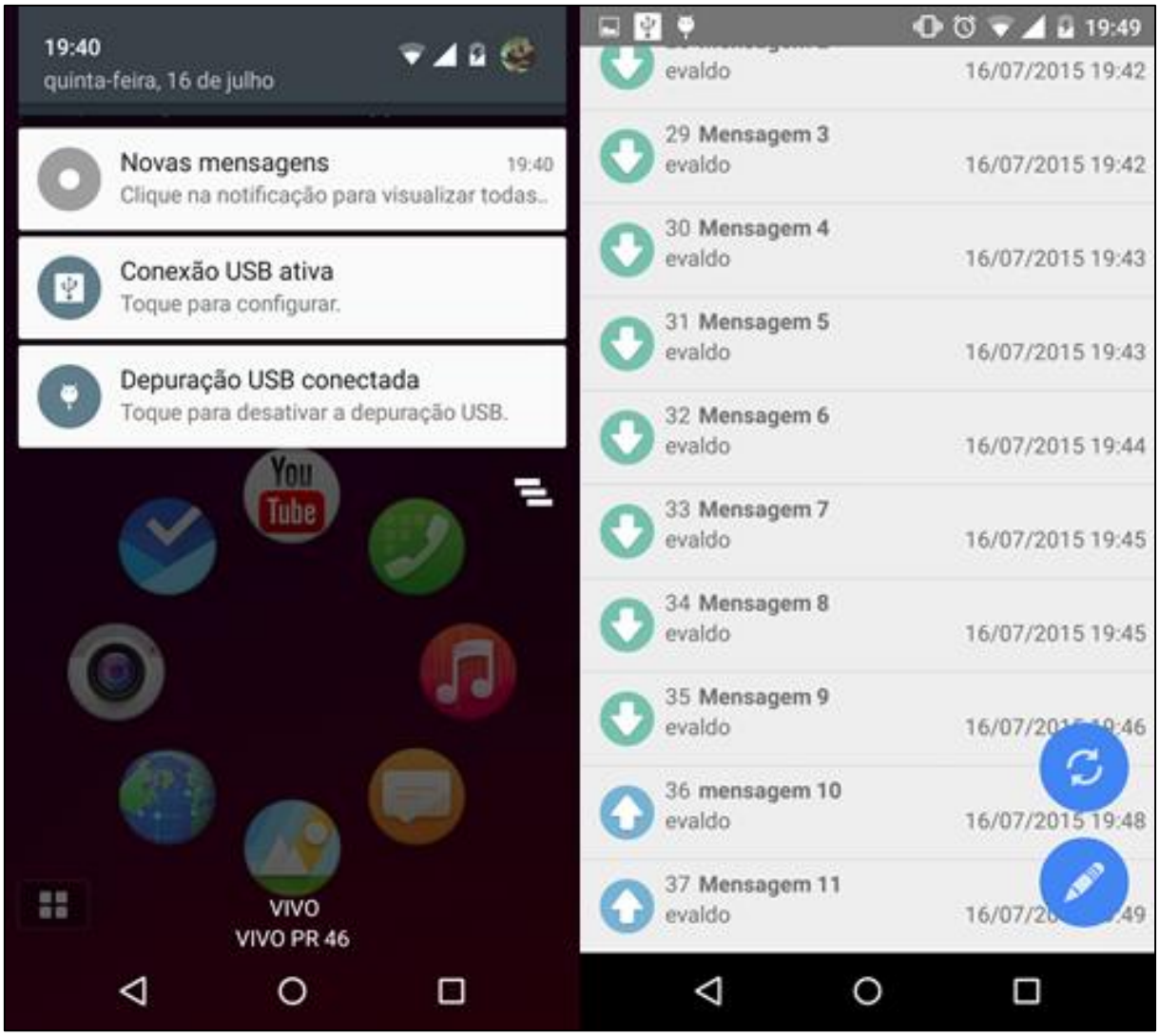

Figura 23 - SADI Mobile (notificação e tela inicial)

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta sessão serão apresentados os principais códigos para exemplificar o desenvolvimento do sistema deste trabalho, em especial, a utilização das tecnologias Spring Boot e GCM.

Por ser desenvolvido com o Spring Boot, o início da aplicação se dá por meio de uma classe principal, que através das anotações, irá mapear todas as classes do projeto e iniciar o serviço para começar a receber as requisições. A Listagem 1 demostra a classe responsável por iniciar todo o serviço.

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class MainServer extends SpringBootServletInitializer {
	public static void main(String[] args) {
		Object[] obj =  {MainServer.class, MensagemJob.class};
		SpringApplication.run(obj , args);
	}
}
```

Listagem 1 - Classe principal do Servidor

A seguir, a Listagem 2 demostra o funcionamento da tarefa, este que é executado a cada 60 segundos, conforme configurado na anotação @Scheduled, e que faz uma busca no banco de dados pelas mensagens que não foram notificadas, monta uma fila em memória e envia as notificações, uma a uma, caso o destinatário possua o GcmId.

```
@EnableScheduling
public class MensagemJob {
	@Autowired private MensagemDestinatarioService
mensagemDestinatarioService;
	@Autowired private NotificacaoService notificacaoService;

	@Scheduled(cron = "0/60 * * * * ?")
	public void execute() {
		List<MensagemDestinatario> mensagens =
mensagemDestinatarioService.obterMensagensPendentesNotificacao();

		for (MensagemDestinatario m : mensagens) {

			if ( m.getDestinatario() != null &&
m.getDestinatario().getGcmId() != null ) {
				notificacaoService.notificarMensagem(m);
			}
		}
	}
}
```

Listagem 2 - Classe MensagemJob

O código da Listagem 3, demonstra o envio da notificação. Nessa classe, existe uma variável com o código gerado pelo Google para identificar o projeto, uma chave para evitar duplicidade de notificações para o mesmo aparelho e um nome para a mensagem de notificação.

O método “notificarMensagem” recebe uma mensagem e busca no banco de dados pelos dispositivos registrados para o usuário destinatário, em seguida envia a notificação e atualiza a mensagem como notificada, caso obtenha êxito. Na listagem, também são utilizadas algumas classes do Android para realizar o envio, são o *Sender*, *Message* e *MultcastResult*.

```
@Service
public class NotificacaoServiceImpl implements NotificacaoService {
	private static final String SENDER_ID =
"AIzaSyBd047SWgQUgIOPIdpy28gq3URCDfNC-sY";

	private static final String COLLAPSE_KEY = "SADI_MENSAGEM";
	private static final String MESSAGE = "NEW_MESSAGE";

	@Autowired private MensagemDestinatarioService
mensagemDestinatarioService;

	public void notificarMensagem(MensagemDestinatario mensagem) {

		List<String> androidTargets = new ArrayList<String>();
		androidTargets.add(mensagem.getDestinatario().getGcmId());

		Sender sender = new Sender(SENDER_ID);

		Message message = new Message.Builder()
		.collapseKey(COLLAPSE_KEY)
		.timeToLive(30)
		.delayWhileIdle(true)
		.addData("message", MESSAGE)
		.build();

		try {
			MulticastResult result = sender.send(message,
androidTargets, 3);

			if (result.getResults() != null) {
				mensagem.setStatus(StatusMensagem.NOTIFICADA);

	mensagemDestinatarioService.saveMensagemDestinatario(mensagem);

			}

		} catch (Exception e) {…}
	}
}
```

Listagem 3 - Envio de notificação GCM

Para a utilização do serviço do GCM, no projeto Android são necessárias algumas configurações no arquivo android_manifest.xml, estas exibidas na Listagem 4. É adicionada a

permissão para o receiver, adicionada a biblioteca do Google Play Services, criada a declaração da classe receiver e o serviço que irá baixar as mensagens do servidor.

```
<permission
        android:name="com.ecda.sadimobile.view.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="com.ecda.sadimobile.view.permission.C2D_MESSAGE" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"
/>

<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <receiver
            android:name="com.ecda.sadimobile.gcm.GcmBroadcastReceiver"
            android:permission="com.google.android.c2dm.permission.SEND" >
            <intent-filter>

                <action android:name="com.google.android.c2dm.intent.RECEIVE"
/>

                <category android:name="com.ecda.sadimobile.view" />
            </intent-filter>
        </receiver>

        <service android:name="com.ecda.sadimobile.gcm.GcmIntentService" />
```

Listagem 4 - Configurações do manifest Android

Como foi citado anteriormente, ao acessar a aplicação mobile pela primeira vez e realizar o login, será feito o registro do dispositivo para obter o gcmId e poder receber as notificações do servidor. O código da Listagem 5, demonstra como é realizado esse processo.

No código listado existe a chave gerada no site do Google para o registro, a verificação do Google Play Services e o registro propriamente dito, que retorna um código único, armazenado no mobile e enviado para o servidor na sequência.

```
public class GCMService {
    private static final String SENDER_ID = "685897362213";
    private String regid;

    public String registraGCM() {
        if (checkPlayServices()) {
            gcm = GoogleCloudMessaging.getInstance(context);
            if (gcm == null) {
                gcm = GoogleCloudMessaging.getInstance(context);
            }
            try {
             regid = gcm.register(SENDER_ID);
            } catch (IOException ex) {…}
        }
        return regid;
      }
      private boolean checkPlayServices() {…}}
```

Listagem 5 - Registro de GCM

O código da classe responsável por receber as mensagens de push e iniciar o serviço para receber as mensagens do servidor, declarada no manifesto do projeto da listagem 4, é demonstrado na Listagem 6.

```
public class GcmBroadcastReceiver extends WakefulBroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ComponentName comp = new ComponentName(context.getPackageName(),
GcmIntentService.class.getName());
        startWakefulService(context, (intent.setComponent(comp)));
        setResultCode(Activity.RESULT_OK);
    }
}
```

Listagem 6 - Receptor de mensagens GCM

A classe do serviço, também declarada no manifesto do projeto, é demonstrada na Listagem 7. No processo, é verificado o tipo da mensagem, se estiver de acordo, é chamado o método para baixar as mensagens novas do servidor e após o término, o método "sendNotification" é chamado, este é o momento que o usuário recebe a notificação na tela e ao tocar, abre a lista de mensagens.

```
public class GcmIntentService extends IntentService {
    public GcmIntentService() {
        super("GcmIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        Bundle extras = intent.getExtras();
        GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(this);
        String messageType = gcm.getMessageType(intent);
        if (!extras.isEmpty()) {
       if (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equals(messageType)) {

             obterNovasMensagens();

                sendNotification("Received: " + extras.toString());
            }
        }
        GcmBroadcastReceiver.completeWakefulIntent(intent);
    }

      private void obterNovasMensagens() {…}


    private void sendNotification(String msg) {…}
}
```

Listagem 7 - Download de mensagens novas do Servidor

Por fim, utilizando a biblioteca Spring para Android, na Listagem 8, pode-se observar como é realizado o envio de uma mensagem do mobile para o servidor, todas as demais interações são desenvolvidas de forma semelhante.

Neste, o método “enviarMensagem” recebe uma mensagem por parâmetro, é criada a URL utilizando uma constante concatenada com “mensagem/”, em seguida é declarado o modelo do Spring (RestTemplate), adicionado o conversor para o formato JSON e por fim enviada, tento como retorno outra mensagem que será tratada de acordo com o que retornar.

```
public MensagemMobile enviarMensagem(MensagemMobile mensagem) throws
NoSuchAlgorithmException {

        StringBuilder url = new
StringBuilder().append(Constantes.urlBase).append("mensagem/");

        RestTemplate restTemplate = new RestTemplate();

        restTemplate.getMessageConverters().add(new
MappingJackson2HttpMessageConverter());

             return restTemplate.postForObject(url.toString(), mensagem,
MensagemMobile.class );
    }
```

Listagem 8 - Envio de mensagem para o Servidor

5 CONCLUSÃO

Este trabalho teve como objetivo demonstrar o desenvolvimento de um sistema mobile para o envio e recebimento de mensagens integrado a um sistema Web pré-existente, demonstrando a utilização do serviço de GCM e aplicando o framework Spring Boot, de forma a facilitar o desenvolvimento de aplicações Java Web e o framework Spring Android para a comunicação entre os sistemas.

Foi possível entender, de uma forma geral, o funcionamento de um sistema básico de troca de mensagens, o desenvolvimento de um Web service, a integração com um sistema distinto, além de demonstrar a utilização dos recursos disponíveis na plataforma Android como a conectividade entre os dispositivos e o servidor de dados, e o serviço de notificações.

No termino do desenvolvimento, obteve-se um sistema funcional com o aplicativo SADI Mobile e o servidor enviando e recebendo informações de forma competente, expandindo assim as funcionalidades do sistema SADI Web. Obteve-se também uma satisfação com relação ao Spring Boot, dada a facilidade de configuração e desenvolvimento de projetos Java Web, com certeza esse *framework* tem um futuro promissor.

Das dificuldades encontradas durante o desenvolvimento deste trabalho pode-se destacar a dificuldade em encontrar materiais ou publicações sobre sistemas de mensagens, sejam mobile, Web ou desktop, visto que são tecnologias relativamente novas mas tratam-se de sistemas complexos, onde, dependendo do alcance de usuários, pode demandar de uma infraestrutura muito eficiente para manter o serviço funcionando. Durante a análise e codificação, a maior dificuldade foi criar a estrutura de tabelas e a ordem de envios e recebimentos para que o mobile e o Web ficassem sincronizados, este é um aspecto que pode ser melhorado.

O resultado deste trabalho, abre grandes possibilidades para melhorias e desenvolvimentos futuros, como a atualização da lista de contatos caso sejam criados novos ou atualizados os existentes, o desenvolvimento de um serviço que permitiria o sistema Web enviar notificações para mensagens novas, dispensando a utilização do Job no servidor, melhorias de interface e usabilidade além da adição de mais recursos presentes no sistema Web, para aprimorar ainda mais a experiência do usuário.

Por fim, é possível concluir que os objetivos foram alcançados da forma proposta, o desenvolvimento com Spring Boot facilita o processo e o GCM é uma poderosa ferramenta para o desenvolvimento de aplicações mobile.

REFERÊNCIAS

ANDROID ANNOTATIONS. **Android Annotations Wiki**. Disponível em: <https://github.com/excilys/androidannotations/wiki>. Acesso em 20 jun. 2015.

AZEVEDO JUNIOR, Sérgio. **Arquitetura REST com Java: JAX-RS**. Disponível em: <http://blog.caelum.com.br/arquitetura-rest-com-java-jax-rs/>. Acesso em 27 jun. 2015.

DEVELOPER. **Android Developer**. Disponível em: <http://developer.android.com/index.html>. Acesso em: 29 mai. 2015.

DEVELOPERS. **Google Cloud Messaging**. Disponível em <https://developers.google.com/cloud-messaging/>. Acesso em 28 jun. 2015.

ECLIPSE. **Eclipse EE**. Disponível em: <https://www.eclipse.org/>. Acesso em 5 jul. 2015.

FONTOURA, Vidal. **Implementando Web Service REST e Client REST Utilizando Spring**. Disponível em <http://www.matera.com/br/2013/01/25/implementando-web-service-rest-e-client-rest-utilizando-spring/>. Acesso em 1 jul. 2015.

G1. **Google Play tem 700 mil aplicativos, mesmo número da loja da Apple.** Disponível em: <http://g1.globo.com/tecnologia/noticia/2012/10/google-play-tem-700-mil-aplicativos-mesmo-numero-da-apple-diz-agencia.html>. Acesso em: 20 abr. 2015.

GOOGLE PLAY. **Google play**. Disponível em < https://play.google.com/store/apps/details?id=com.whatsapp>. Acesso em 23 jul. 2015.

GUIMARÃES, Saulo Pereira. **Brasil é o quarto país do mundo em número de smartphones**. Disponível em: <http://exame.abril.com.br/tecnologia/noticias/brasil-e-o-quarto-pais-do-mundo-em-numero-de-smartphones>. Acesso em 29 abr. 2015.

MELLO, Joisson J.; PEREIRA Fabio S. **Sistema de Automatização de Processos em Centros Municipais de Educação Infantil – CMEIS**. 2014. 59 f. Trabalho de Conclusão do Curso

(Tecnologia em Análise e Desenvolvimento de Sistema) – Universidade Tecnológica Federal do Paraná, Pato Branco.

ORACLE. **Java SE**. Disponível em: <http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>. Acesso em: 1 mai. 2015.

ORACLE. **MySQL**. Disponível em <http://www.oracle.com/br/products/mysql/index.html>. Acesso em 1 jul. 2015.

PINHEIRO, Amanda. **Conheça o ZapZap, versão brasileira do WhatsApp**. Disponível em <http://www.euviali.com/curiosidades/conheca-o-zap-zap-versao-brasileira-whatssap/>. Acesso em 25 jul. 2015.

PISA, Pedro. **O que é e como usar o MySQL?** Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>. Acesso em: 10 mar. 2015.

RASMUSSEN, Bruna. **Android: o sistema operacional móvel que conquistou o mundo**. Disponível em: <http://www.tecmundo.com.br/infografico/9010-android-o-sistema-operacional-movel-que-conquistou-o-mundo.htm#ixzz2Y2OCqLHN>. Acesso em: 17 mai. 2015.

SQLITE. **SQLite**. Disponível em: <http://www.sqlite.org/index.html>. Acesso em: 5 jun. 2015.

TUDOCELULAR. **WhatsApp ultrapassa mensagens de texto tradicionais com 30 bilhões de envios por dia**. Disponível em: <http://www.tudocelular.com/android/noticias/n52140/WhatsApp-ultrapassa-mensagens-de-texto-tradicionais-com-30-bilhoes-de-envios-por-dia.html>. Acesso em 17 jun. 2015.

VISUAL PARADIGM. **Visual Paradigm for UML 11.0 Community Edition**. Disponível em: <http://www.visual-paradigm.com/product/vpuml/editions/community.jsp>. Acesso em: 20 jun. 2015.

WEISSMANN, Henrique Lobo. **Spring Boot: simplificando Spring**. Disponível em: <http://www.devmedia.com.br/spring-boot-simplificando-spring-revista-java-magazine-135/31979>. Acesso em 21 jun. 2015.

WHATSAPP. **WhatsApp**. Disponível em: <https://www.whatsapp.com/>. Acesso em 15 jun. 2015.

ZAPZAP. **ZapZap**. Disponível em: <http://www.site.zapzap.gratis/>. Acesso em 20 jun. 2015.