

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

**DESENVOLVIMENTO DE UM PROTÓTIPO DE NOTIFICAÇÃO PARA O
PORTAL DO ALUNO DA UFSM**

Jéssica Letícia Félix Mouta

Santa Maria, RS, Brasil

2014

DESENVOLVIMENTO DE UM PROTÓTIPO DE NOTIFICAÇÃO PARA O PORTAL DO ALUNO DA UFSM

Jéssica Letícia Félix Mouta

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da
Universidade Federal de Santa Maria (UFSM, RS), como requisito para
a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: João Carlos Damasceno Lima, Prof. Dr.

Trabalho de Graduação N. 388

Santa Maria, RS, Brasil

2014

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

**DESENVOLVIMENTO DE UM PROTÓTIPO DE NOTIFICAÇÃO PARA O
PORTAL DO ALUNO DA UFSM**

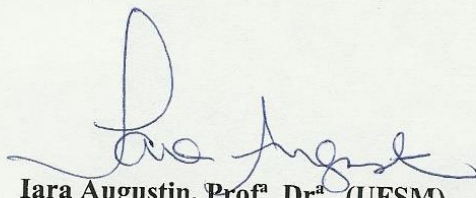
elaborado por

Jéssica Letícia Félix Mouta

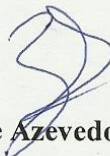
como requisito para obtenção do grau de
Bacharel em Ciência da Computação
COMISSÃO EXAMINADORA:



João Carlos Damasceno Lima, Prof. Dr. (UFSM)
(Presidente/Orientador)



Iara Augustin, Prof^a. Dr^a. (UFSM)



Bruno Romero de Azevedo, Bel. (UFSM)

Santa Maria, 03 de Novembro de 2014.

DEDICATÓRIA

Dedico este trabalho à minha família, que nunca deixou de acreditar em mim.

AGRADECIMENTOS

Dedico meus agradecimentos a todos que de alguma forma colaboraram com a realização deste trabalho:

- aos meus familiares, pelo apoio em minhas decisões, especialmente a minha mãe;
- ao professor João Carlos Damasceno Lima, pela orientação neste trabalho;
- aos colegas de graduação, em especial a Liza, Felipe Maia, Ricardo, Vanderlan, Tiago, Giane e Andressa, que sempre me ajudaram e me acompanharam durante o curso;
- aos colegas do grupo PET - Ciência da Computação, que me proporcionaram grandes aprendizados;
- à coordenação e aos demais professores do curso de Ciência da Computação, que colaboraram na minha formação acadêmica.
- e principalmente ao meu marido José que sempre me apoiou e incentivou estando do meu lado em todos os momentos.

“O estudo da ciência da Computação não consegue transformar qualquer um em um excelente programador, da mesma forma que o estudo de tintas e pinceis não transforma qualquer um em um excelente pintor.” - Eric S. Raymond

RESUMO

Trabalho de Graduação

Curso de Ciência da Computação

Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM PROTÓTIPO DE NOTIFICAÇÃO PARA O PORTAL DO ALUNO DA UFSM

AUTOR: JÉSSICA LETICIA FELIX MOUTA

ORIENTADOR: JOÃO CARLOS DAMASCENO LIMA

Local da Defesa e Data: Santa Maria, 05 de Dezembro de 2014.

Com o crescente aumento da popularidade dos dispositivos móveis, é cada vez maior a necessidade do desenvolvimento de aplicativos que facilitem o acesso a dados remotos para prover consulta de informações. Essa comunicação pode ser feita de diversas formas, utilizando o protocolo HTTP, *sockets* ou *web services*. Neste contexto de provimento de informações, temos o *site* do Portal do Aluno da Universidade Federal de Santa Maria (UFSM) que disponibiliza informações, como notas das provas, faltas, horários das aulas, a situação do Restaurante Universitário, solicita requerimentos, entre outros. No entanto, para ter acesso a essas informações é necessário realizar a autenticação do usuário com a matrícula e senha do aluno. Desta forma, este trabalho se propõe a construir um aplicativo para dispositivos móveis que possibilite o acesso aos dados e informações do *site* do Portal do Aluno, criando um sistema de notificações.

O aplicativo oferece duas funcionalidades principais: a primeira funcionalidade refere-se ao quadro de notas do aluno e permitirá ao usuário receber notificações quando as notas das disciplinas, definidas previamente, estiverem disponíveis no portal. A segunda funcionalidade está relacionada ao Restaurante Universitário, o aplicativo realizará uma consulta à situação do saldo do Restaurante Universitário e emitirá uma notificação com o valor do saldo.

O aplicativo foi desenvolvido para dispositivos com sistema operacional Android utilizando a linguagem de programação Java e a IDE Eclipse, em conjunto com as ferramentas disponibilizadas no SDK do Android. A validação é realizada com testes para verificar a usabilidade do aplicativo.

Palavras-chave: Computação Móvel. Comunicação de rede. Protocolo HTTP. Android. Dispositivos Móveis. Mobilidade.

ABSTRACT

Undergraduate Final Work

Undergraduate Program in Computer Science

Federal University of Santa Maria

DEVELOPMENT OF A PROTOTYPE OF NOTIFICATION FOR STUDENT PORTAL OF UFSM

AUTHOR: JÉSSICA LETICIA FELIX MOUTA

ADVISOR: JOÃO CARLOS DAMASCENO LIMA

Defense Place and Date: Santa Maria, December 5, 2014.

The growing popularity of mobile devices is causing a growing demand on the development of applications that ease access to remote service providers. Traditionally, the communication is made via HTTP protocol, sockets or web services. In the information providing context we have the Student Portal of the Federal University of Santa Maria (UFSM), which offers information such as test assessment, attendance, timetable, access to the university restaurant balance, among other features. However, the Portal access requires user authentication every time, this authentication is done with the login and password of the student. We present a mobile application that allows access to the Portal's information, creating a notification system. There are two main features: the first regards student assessment and feedback when they are available on the Portal. The second feature regards query about the current balance on the university restaurant.

The application runs on Android devices and was developed in Java, using Eclipse IDE, as well as the Android SDK tools. Finally, the validation was carried out tests to verify the usability of the application.

Keywords: Mobile Computing. Network communication. Protocol HTTP. Android. Mobile Devices. Mobility.

LISTA DE FIGURAS

Figura 1 - Página inicial do portal do aluno da UFSM.....	19
Figura 2 - Gráfico de desempenho das Plataformas Móveis.....	20
Figura 3 - Distribuição das versões do Android entre os usuários [DEVELOPERS, 2014].....	21
Figura 4 - Comunicação Cliente/Servidor.	24
Figura 5 - Diagrama de Caso de Uso.....	28
Figura 6 - Diagrama de Atividades.....	28
Figura 7 - Diagrama de Classes.....	34
Figura 8 - Tela de login do aplicativo.....	39
Figura 9 - Tela de Preferências.....	39
Figura 10 - Tela de opções.....	40
Figura 11 - Tela de histórico de notificações do saldo do RU.....	40
Figura 12 - Tela de histórico de notificações do quadro de notas.....	41
Figura 13 - Tela de notificações.....	41
Figura 14 - Tela que detalha a disciplina selecionada.....	41

LISTA DE TABELAS

Tabela 1 – Requisitos Funcionais e Não Funcionais do Sistema.....	30
---	----

LISTA DE ABREVIATURAS

ADT	Android Development Tools
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
iOS	iPhone Operational System
SDK	Software Development Kit
UFSM	Universidade Federal de Santa Maria
UML	Unified Modeling Language
OHA	Open Handset Alliance
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
HTML	HyperText Markup Language
DAO	Data Access Object
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO	14
1.1 Objetivos.....	15
1.1.1 Objetivos Específicos	16
1.2 Justificativa	16
1.3 Estrutura do Texto.....	16
2 FUNDAMENTAÇÃO E REVISÃO TEÓRICA	18
2.1 Portal do Aluno da UFSM	18
2.2 Plataformas Móveis	20
2.2.1 Android	21
2.3 Sistema de Notificação	22
2.4 Comunicação com a rede	23
2.4.1 Clientes e Servidores	23
2.4.2 <i>Sockets</i>	24
2.4.3 <i>Web Services</i>	24
2.4.4 Protocolo HTTP.....	25
2.5 Métodos de comunicação utilizado no aplicativo.....	26
3 DESENVOLVIMENTO DO APLICATIVO PARA ACESSAR O PORTAL DO ALUNO DA UFSM	27
3.1 Definição da aplicação.....	27
3.1.1 Requisitos do Sistemas	29
3.2 Experiências Preliminares	30
3.2.1 Preferências	30
3.2.2 Implementar serviços de background.....	31
3.2.3 Criar notificação na barra de <i>status</i>	31
3.3 Implementação do aplicativo	31
3.3.1 <i>Parser</i> HTML	34
3.3.2 Controle de Notificações	36
3.4 Descrição das Interfaces.....	37
3.5 Testes de Funcionalidades	42
3.5.1 Teste para notificação de saldo	42
3.5.2 Teste para notificação de notas	42

3.5.3 Testes de autenticação.....	43
3.5.4 Testes de conectividade	43
4 CONCLUSÃO.....	44
5 REFERÊNCIAS	45

1 INTRODUÇÃO

A evolução nos métodos de comunicação sem fio, ampliando o conceito de computação distribuída, possibilitou o crescimento no desenvolvimento de tecnologias para a Computação móvel. Com a disseminação dessas tecnologias, foi se permitindo, cada vez mais, o acesso às informações remotas onde quer que se esteja, tornando possível o surgimento de inúmeras facilidades, sistemas e serviços para os usuários, possibilitando o desenvolvimento de novas ferramentas para auxiliarem as pessoas, seja no âmbito pessoal ou profissional [FIGUEIREDO, CARLOS M. S. NAKAMURA, EDUARDO, 2014].

Com a crescente popularização dos dispositivos móveis, cada vez mais aumenta o número de pessoas que estão utilizando esses dispositivos com capacidade de comunicação entre si, com a internet e com redes fixas tradicionais.

Os dispositivos móveis permitem aos usuários, ter acesso a serviços independentemente de sua localização, podendo inclusive estar em movimento. Desta forma, a Computação Móvel está diretamente relacionada ao conceito de mobilidade das informações. A mobilidade pode ser definida como a capacidade de acessar informações a partir de qualquer lugar e a qualquer momento. Para isso, é preciso contar com dispositivos móveis como *notebooks*, *tablets* e *smartphones*, e com uma conexão *wireless* eficiente, para acessar os sistemas remotos e à Internet [INTEL CORPORATION, 2014].

Com o constante aumento de informações espalhadas na internet, algo que tem se tornado comum é que as aplicações, tanto *web* como *mobile* e até mesmo aplicações *desktop*, utilizem serviços remotos disponibilizados por outras aplicações, consumindo informações. Os usuários esperam cada vez mais centralização e integração de dados, com o intuito de trazer acesso rápido e comodidade, tendo acesso às informações, literalmente, na ponta dos dedos. Grandes empresas como Facebook, Twitter, Google e Yahoo! Disponibilizam acesso remoto a seus serviços através de uma API [LECHETA, 2009].

Neste contexto, onde é cada vez maior a necessidade do desenvolvimento de aplicativos que facilitem o acesso a dados remotos, temos o *site* do Portal do Aluno da Universidade Federal de Santa Maria (UFSM) que disponibiliza informações, como notas das provas, faltas, horários das aulas, a situação do Restaurante Universitário, entre outros. Este trabalho propõe desenvolver um

aplicativo para acessar os dados do portal do aluno da UFSM e disponibilizar o acesso às informações por meio de dispositivos móveis, criando um sistema de notificações, buscando fornecer algumas funcionalidades úteis ao dia-a-dia de seus usuários.

Por necessitar de autenticação para ter acesso ao seu conteúdo, o site do Portal do Aluno pode ser caracterizado como uma *Deep Web*. Deep web ou web oculta ou profunda é uma parte da internet constituída por um conjunto de sites, fóruns e comunidades que não podem ser detectados pelos tradicionais motores de busca, como o Google, por exemplo. Os sites que estão na *Deep Web* buscam uma coisa em comum: a privacidade, uma vez que a única maneira de obter acesso a dados na Deep Web é através da submissão de formulário. Um dos desafios encontrados é como preencher os formulários da *web* automaticamente.

No entanto, uma grande parte das informações sobre a *Deep Web* está disponível em bases de dados online e só pode ser acessada por meio do preenchimento de formulários web. Esses formulários geralmente estão em formato HTML, contendo campos a serem preenchidos para que a submissão dos mesmos possa recuperar dados, acessando o banco de dados online escondido por trás do formulário. Os campos podem ser caixas de texto, listas de seleção, caixas de seleção, botões de *radio*, e botões de *submit*. Eles podem ser classificados em dois grupos: os campos com domínios finitos (como listas de seleção) e campos com domínios infinitos (tais como caixas de texto, em que um usuário pode digitar qualquer valor). Uma vez que os valores sejam preenchidos, o formulário pode ser enviado. A submissão de um formulário pode ser feita por dois métodos, GET ou POST [KANTORKI, Gustavo Z., MORAIS, Tiago G., MOREIRA, Viviane P., HEUSER, Carlos A, 2013].

Um dos desafios é como preencher automaticamente esses formulários para ter acesso aos dados. Esta tarefa não é trivial, uma vez que esses formulários foram projetados para serem utilizados pelos seres humanos. Se o formulário possui campos de texto de preenchimento obrigatório, a intervenção do usuário é necessária.

1.1 Objetivos

O objetivo geral deste trabalho é explorar o conceito de Computação Móvel que possibilitem a implementação de uma aplicação ágil, com interface amigável e de fácil interação com o usuário. O objetivo principal é criar um aplicativo que permita a conexão com os dados do

portal do aluno da UFSM para a implementação de um sistema de notificações e analisar técnicas para capturar informações da rede.

1.1.1 Objetivos Específicos

Este trabalho englobará os seguintes objetivos específicos:

- Escolha do método de acesso à rede para realizar a comunicação entre a aplicação e o servidor;
- Aprofundar os conhecimentos específicos sobre a plataforma Android;
- Criação de duas funcionalidades: uma das funcionalidades será relacionada ao quadro de notas do aluno, onde o aplicativo tem a função de emitir notificações quando as notas de disciplinas, pré-definidas no sistema, estiverem disponíveis para visualização no portal do aluno. A segunda funcionalidade tem como objetivo realizar uma consulta à situação do saldo referente ao Restaurante Universitário, emitindo uma notificação para informar ao usuário o valor atual do saldo;
- Desenvolver um aplicativo para a plataforma Android para o fim desejado, utilizando os conhecimentos adquiridos.

1.2 Justificativa

Para acessar os dados do Portal do Aluno da UFSM é necessário ser um usuário autenticado, desta forma, sempre que se desejar obter quaisquer informações do Portal é necessário realizar a autenticação no sistema. A principal vantagem da criação de um aplicativo para Portal do Aluno está em proporcionar praticidade para o usuário, não havendo a necessidade de fazer a autenticação toda vez que desejar obter informações referentes ao saldo do Restaurante Universitário e ao quadro de nota. Além de criar um sistema de notificações, no qual o dispositivo irá avisá-lo toda vez que for encontrada uma nova informação.

1.3 Estrutura do Texto

Este trabalho está estruturado da seguinte forma: no Capítulo 2 é realizada a revisão bibliográfica sobre conceitos e tecnologias úteis para o desenvolvimento do trabalho. O Capítulo 3 detalha a criação do aplicativo, apresentando as ferramentas usadas para seu desenvolvimento e as funcionalidades disponíveis. Por último, o capítulo 4 conclui o trabalho.

2 FUNDAMENTAÇÃO E REVISÃO TEÓRICA

Este capítulo destina-se à definição de conceitos teóricos acerca dos assuntos abordados no trabalho. Apresentando uma análise sobre o funcionamento do Portal do Aluno da UFSM, um estudo sobre as plataformas móveis atuais, a importância de sistemas de notificações nas aplicações móveis atuais, e um estudo sobre os diferentes tipos de comunicação com a rede entre dispositivos móveis.

2.1 Portal do Aluno da UFSM

O *site* do Portal do Aluno da UFSM é um *site* de interação do aluno com a instituição de ensino que fornece diversas funcionalidades. Com o Portal do Aluno, o aluno controla e acompanha seus dados referentes aos cursos realizados na instituição, como notas das provas, faltas e horários das aulas, solicita requerimentos, solicitação de matrícula, situação do Restaurante Universitário, entre outros. Também é utilizado para exibir notícias sobre a instituição.

Para utilizar os recursos do portal é necessário realizar a autenticação com *login* e senha. O *login* de acesso é a matrícula fornecida pela instituição e a senha é criada pelo aluno. Após a autenticação o aluno tem acesso a um menu de opções, que pode ser visualizado na Figura 1, com as seguintes funcionalidades:

- Página inicial: retorna para a página de *login*;
- Relatórios: possibilita a geração de atestados, certificados, comprovante de matrícula, histórico escolar simplificado, índice de desempenho acadêmico, quadro de horários e quadro de notas;
- Dados cadastrais: possibilita que o aluno altere seus dados cadastrados;
- Alteração curricular: possibilita solicitação de trancamento ou requerimento de dispensa;
- Matrícula: possibilita solicitação de matrícula e acesso ao comprovante de matrícula;
- Outros: possibilita acesso ao calendário letivo, à biblioteca e ao Restaurante Universitário;

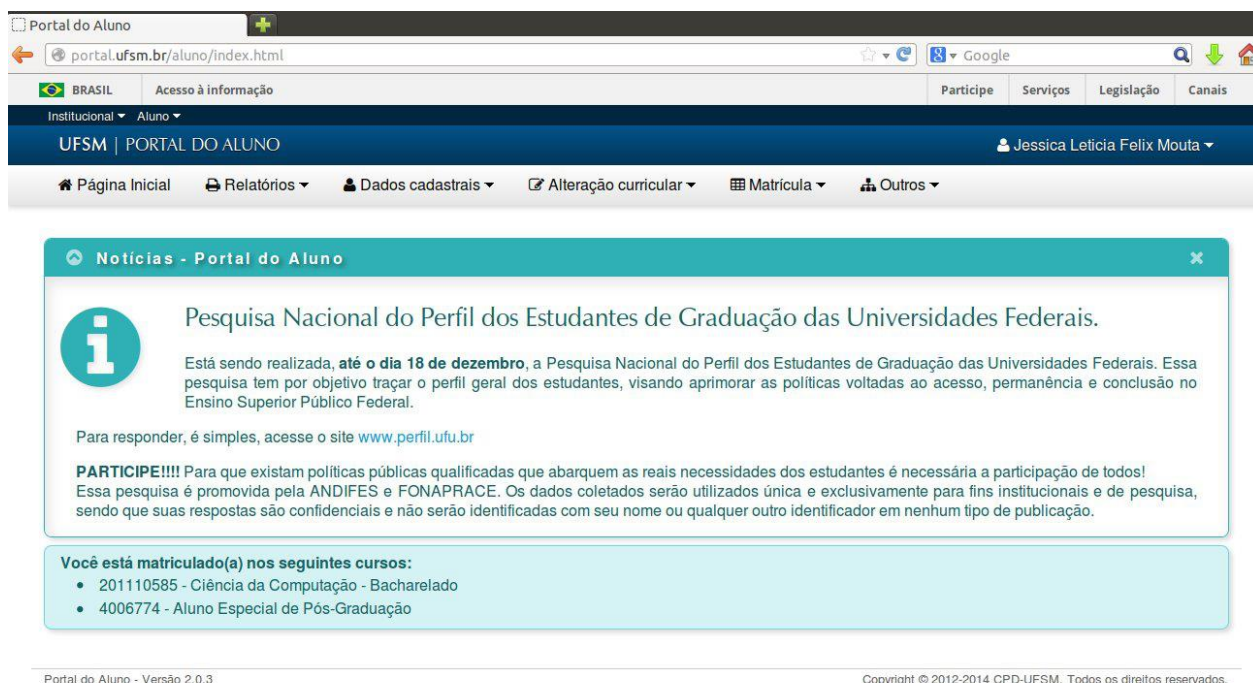


Figura 1 - Página inicial do portal do aluno da UFSC

Para desenvolver essa aplicação serão consideradas duas funcionalidades: exibir o quadro de notas do aluno e exibir o saldo do Restaurante Universitário.

Ao navegar no *site*, percebe-se que para acessar a situação do aluno referente ao Restaurante Universitário é necessário fazer a autenticação duas vezes. No entanto, se utilizarmos o link direto [Portal RU, 2014] para consultar a situação a autenticação é solicitada apenas uma vez, mas não permite acesso às outras informações do Portal do Aluno.

Atualmente, existe um aplicativo, chamado UFSC Mobile, disponível para *download* na Google Play Store [Google Play Store, 2014]. Este aplicativo permite acessar os conteúdos disponíveis no Portal da UFSC, como consultar os horários dos ônibus, o cardápio do Restaurante Universitário, acesso ao Portal do Aluno, as notícias e os destaques da UFSC. No entanto, este aplicativo possui apenas *links* que redirecionam para o *site* da UFSC, com as mesmas informações da página da UFSC no navegador, sendo necessário realizar a autenticação no sistema todas as vezes que se desejar buscar informações.

2.2 Plataformas Móveis

Os principais sistemas operacionais comercializados na área de Computação Móvel, atualmente, são o Android e o iOS, presentes na maioria dos dispositivos móveis [NETMARKETSHARE, 2014]. No entanto, a popularidade do sistema Android é visivelmente superior, por estar presente em centenas de milhões de dispositivos móveis em mais de 190 países ao redor do mundo. É a maior base instalada de qualquer plataforma móvel e vem crescendo rapidamente. Todos os dias, mais de 1 milhão de novos dispositivos Android são ativados em todo o mundo [DEVELOPERS, 2014]. Devido à possibilidade de ser executado em *smartphones* e *tablets* de diversos fabricantes e não possuir restrições de hardware, o Android vem se tornando uma opção muito rentável.

A Figura 2 mostra que de acordo com dados da NetApplications, em uma pesquisa realizada em julho de 2014, o Android aparece como o sistema operacional móvel mais usado do mundo, com 44,62% do mercado seguido do iOS, que caiu para 44,19% [NETMARKETSHARE,2014].

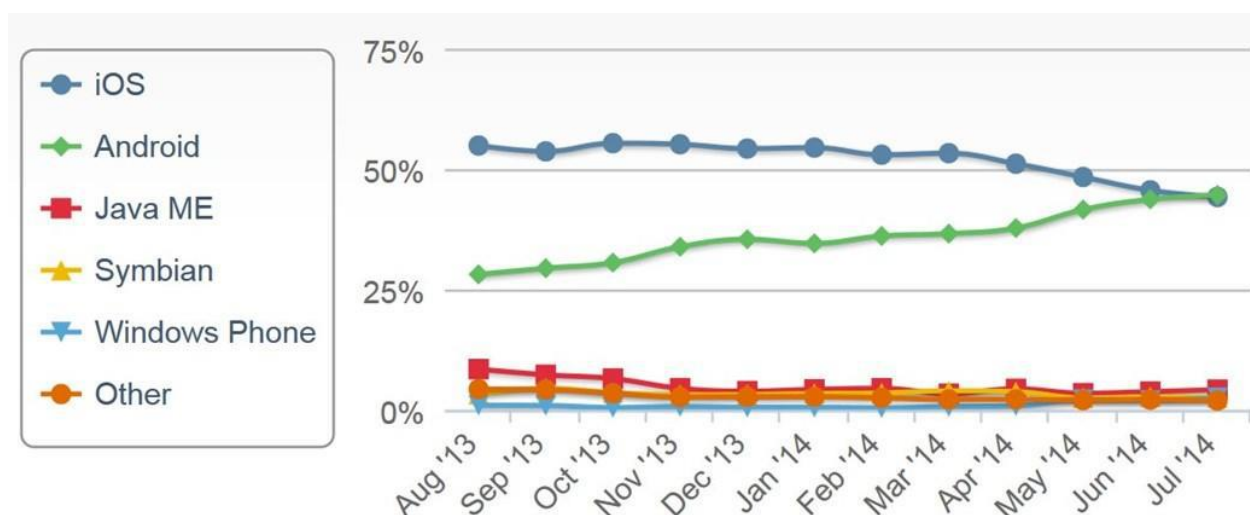


Figura 2 - Gráfico de desempenho das Plataformas Móveis

A plataforma móvel Android se mostrou mais adequada para o desenvolvimento do aplicativo devido ao número de usuários superior à plataforma iOS e a inexistência de restrições de hardware, aumentando a diversidade de aparelhos que poderão executar o aplicativo.

2.2.1 Android

O Android é um sistema operacional para dispositivos móveis baseado em Linux, sendo uma plataforma *open source*. Este sistema operacional foi desenvolvido pela *Open Handset Alliance* (OHA), um grupo formado por uma aliança entre empresas de desenvolvimento de software líderes do mercado da tecnologia móvel [OHA et al., 2014]. O sistema operacional Android teve sua primeira versão, chamada *Cupcake*, lançada em 2008 e, desde então, o número de usuários cresceu significativamente. Atualmente, possui as versões *Jelly Bean*, lançada em 2012 e a última versão 4.4 chamada *KitKat* foi lançada em outubro de 2013 [DEVELOPERS, 2014].

Para o desenvolvimento de aplicativos para essa plataforma, a Google disponibilizou um *Software Development Kit* (SDK) do Android, ou Kit de Desenvolvimento de Software, com emulador para teste, possuindo uma plataforma de desenvolvimento robusta que utiliza Java como linguagem, facilitando assim, o desenvolvimento de aplicações [BENJAMIN, 2008]. Além disso, os aplicativos desenvolvidos para o sistema operacional Android podem ser disponibilizados na loja online Google Play Store [GOOGLE PLAY STORE, 2014].

Segundo dados de distribuição das versões do Android mostrados na Figura 3 [DEVELOPERS, 2014], mais da metade dos usuários utilizam a versão Jelly Bean ou superior. Por esse motivo, Aplicativo será desenvolvido na versão Jelly Bean 4.2.2 para garantir a compatibilidade com a maior parte dos usuários, utilizando os recursos dessa versão.

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%

Figura 3 - Distribuição das versões do Android entre os usuários [DEVELOPERS, 2014]

2.3 Sistema de Notificação

Uma aplicação executando em segundo plano nunca deve exibir um alerta para o usuário ou abrir uma tela sem a permissão do mesmo, pois ele pode estar executando outra aplicação. Desta forma, uma aplicação não pode atrapalhar a atividade que o usuário está executando no momento. Uma alternativa é criar uma notificação, nesse caso o usuário pode decidir visualizar seu conteúdo ou simplesmente ignorar a mensagem [LECHETA, 2009].

O uso de notificações torna o aplicativo mais inteligente e mais atraente para o usuário. Isto porque ele possibilita a definição de uma nova forma de interação entre aplicativo e usuário. Basicamente, um sistema de notificação visa informar o usuário de algo importante sem que o mesmo precise buscar por esta informação ou fazer esta pesquisa em *sites*.

O Android foi um dos pioneiros na criação de sistemas de notificações, trazendo-os desde suas primeiras versões. Além da facilidade que o desenvolvedor encontra para utilizar o mesmo. A notificação é mostrada no topo do aparelho assim que ela é lançada. E ela pode ter som e fazer o dispositivo vibrar. Ou seja, fazendo com que o aplicativo se comunique com o usuário, invertendo a ordem presente nos primeiros sistemas operacionais móveis, onde o usuário precisava abrir o aplicativo, obrigatoriamente, para ver as informações.

Além disso, uma notificação pode ser a porta de saída de um complexo sistema que roda em *background*. Devido à possibilidade de *multithreading* do Android, pode-se fazer diversas consultas em uma quantidade muito grande de informações. Esta característica torna o aplicativo mais inteligente para o usuário.

Notificações podem ser úteis em várias aplicações, tais como aplicativos de e-mails, mensagens, redes sociais, como Facebook, Twitter, Whatsapp, entre outros, possibilitando que o próprio dispositivo avise o usuário sempre que surgir alguma nova informação, em vez de abrir o aplicativo em busca dessas informações. Com isso ganhamos tempo e praticidade em atividades diárias. As notificações também são importantes em sistemas críticos, onde o usuário precisa receber uma informação, esse tipo de notificação pode ser utilizado em sistemas *Home Care*, sistemas de monitoramento ou sistemas de alertas.

Há casos em que receber notificações pode ser um incômodo ao usuário, como notificações de jogos nas redes sociais, por exemplo. Nesses casos o aplicativo deve ter a opção de desabilitar

as notificações. Apesar disso, um bom aplicativo Android faz extenso uso de notificações, isso porque é uma forma aprimorada de interação com os usuários [LECHETA, 2009].

2.4 Comunicação com a rede

Para coletar os dados necessários para as funcionalidades desenvolvidas, o necessita estabelecer uma comunicação com o servidor da UFSM. Para isso, foi utilizada uma arquitetura baseada em Cliente-Servidor, onde o cliente requisita os dados, estando estes em um ou mais servidores.

Portanto, a fim que o aplicativo alcance seus objetivos, precisamos realizar uma comunicação com o servidor, enviando uma requisição e recuperando os resultados para, em seguida, exibi-los. O Android fornece vários métodos para lidar com a comunicação com a rede, os principais são *Socket*, *web services* e solicitações por *Hypertext Transfer Protocol* (HTTP).

2.4.1 Clientes e Servidores

Hoje em dia, qualquer um que já usou um navegador está familiarizado com o modelo Cliente/Servidor. Para conseguir abrir páginas *web*, é necessário a utilização de um navegador, portanto o navegador é o cliente e o provedor do *site* é o servidor, pois o primeiro acessa informações disponibilizadas pelo segundo. A Figura 4 mostra como é feita esta comunicação.

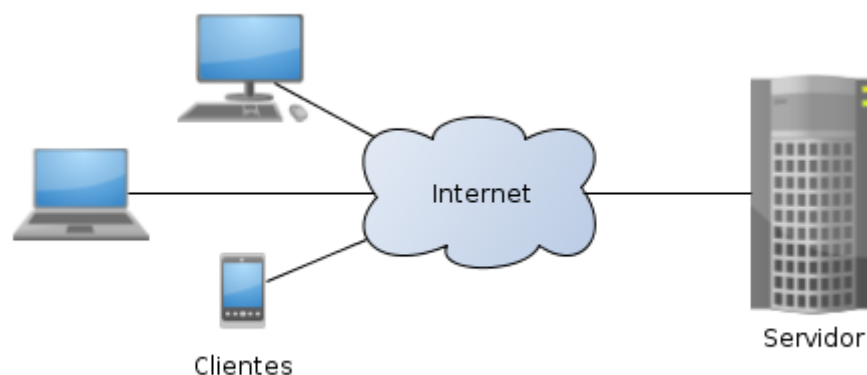


Figura 4 - Comunicação Cliente/Servidor.

Desta forma, as redes que utilizam servidores são chamadas do tipo Cliente/Servidor, onde o cliente faz um tipo de solicitação para o servidor, usando um protocolo designado, como o HTTP, e o servidor responde. Este modelo é o mesmo que a maioria das aplicações Android geralmente seguem. Os aplicativos do Android tipicamente são os clientes [ABLESON et al., 2012].

2.4.2 Sockets

Um *socket* é um ponto de conexão de comunicações em que você pode nomear e endereçar em uma rede. Os processos que usam *sockets* podem estar no mesmo sistema ou em sistemas diferentes e redes diferentes. Eles são úteis, tanto para aplicações *stand-alone*, quanto para aplicações de rede, pois permitem a troca de informações entre os processos na mesma máquina ou em uma rede [IBM, 2013].

Para utilizar uma comunicação com *socket*, é necessário que um computador sirva como servidor. Quando um *socket* é criado ele deve estar associado a uma porta específica que fica esperando as conexões. O dispositivo móvel atua como cliente, chamando o *socket* servidor para iniciar a conexão.

2.4.3 Web Services

Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

O Android não possui nenhuma API nativa para acessar um *web service*, mas é possível utilizar qualquer outra biblioteca e incorporá-la ao projeto normalmente, desde que seja compacta, pois o poder de processamento de um dispositivo móvel é inferior ao de um computador normal [LECHETA, 2009]. Os *web services* SOAP ou REST são os mais comumente utilizados [ABLESON, 2012]. Ambos possuem vantagens e desvantagens e fica a cargo do desenvolvedor determinar a melhor abordagem para cada caso em particular.

2.4.4 Protocolo HTTP

Outra forma para realizar a comunicação em aplicações móveis é utilizando o protocolo HTTP, onde um servidor *web* na internet é utilizado para receber as requisições e enviar a resposta para o cliente. Dessa forma, o dispositivo móvel atua como cliente do servidor *web*, como se fosse um browser [LECHETA, 2009].

O protocolo HTTP permite a clientes e servidores interagir e trocar informações de uma maneira simples e confiável, com a utilização de URIs (*Uniform Resource Identifier*) para identificar dados na Internet. Estes identificadores, aliados às localizações dos documentos no servidor, são chamados de URLs (*Uniform Resource Locators*), que fazem referência a arquivos, diretórios ou objetos situados em um servidor *web* e que serão acessados a partir de um navegador ou diretamente a partir de outras aplicações [DEVMEDIA, 2014].

O protocolo HTTP é *stateless*, ou seja, quando uma conexão é estabelecida entre o cliente e o servidor, os dados são enviados pelo cliente, processados pelo servidor e devolvidos ao cliente. Depois disso, a conexão é interrompida. Para poder armazenar dados sobre a troca de mensagens, devem ser utilizados recursos como Sessão, *Cookies*, entre outros.

Os principais métodos de requisição HTTP são o GET e o POST. Na requisição GET, é passado parâmetros para o servidor HTTP através de uma URL. O método GET deve ser usado para acessar recursos que recuperem as informações presentes no servidor, como buscas, filtros, listagens, etc. O método POST deve ser usado em ações que modifiquem as informações presentes

no servidor, como em alteração, edição e remoção em banco de dados. Além do GET e do POST, o protocolo HTTP possui ainda mais 6 métodos: PUT, DELETE, HEAD, TRACE, CONNECT e OPTIONS.

2.5 Métodos de comunicação utilizado no aplicativo

Após a análise das características de cada método de comunicação e dos aspectos envolvidos no desenvolvimento, e devido à impossibilidade da criação de qualquer serviço do lado servidor, para agilizar o desenvolvimento da aplicação o acesso aos dados portal do aluno será realizado por meio do protocolo HTTP com requisições GET e POST para acesso à base de dados do servidor remoto, com a utilização de uma técnica adequado para essa finalidade.

3 DESENVOLVIMENTO DO APLICATIVO PARA ACESSAR O PORTAL DO ALUNO DA UFSM

Neste capítulo é detalhado o processo de desenvolvimento do aplicativo para acesso às informações do Portal do Aluno da UFSM. O aplicativo desenvolvido provê uma ferramenta para que o usuário, tenha acesso às informações relacionadas ao quadro de notas e ao saldo do Restaurante Universitário. Primeiramente, é descrita uma definição da aplicação, com o objetivo e os seus requisitos. A seguir, é detalhado o projeto do mesmo e, para concluir o capítulo, é descrita a implementação em si, além de uma exibição das interfaces finais obtidas.

3.1 Definição da aplicação

O aplicativo tem como objetivo comunicar-se com um servidor web para capturar dados, essa comunicação é realizada por meio do protocolo HTTP com requisições GET e POST para acesso à base de dados do servidor. O aplicativo funciona como um *browser*, que irá “navegar” pelo site do Portal do Aluno e recuperar páginas no formato HTML como resposta as suas requisições. Após recuperação da página solicitada, é necessário interpretá-la e utilizar um *parser* para extrair informações necessárias para implementar as funcionalidades do aplicativo.

Primeiramente, quando o usuário entrar no sistema será exibido a tela inicial de *login*. Após realizar o *login* aparecerá a tela de preferências do usuário, na qual ele poderá escolher se deseja ou não receber notificações e definir os parâmetros necessários. Para ativar as notificações de notas é obrigatório definir os parâmetros de curso, ano e período de busca por disciplinas.

Após a realização do *login* e a configuração das notificações, o sistema mostrará a tela de relatórios, na qual o usuário poderá acompanhar o histórico de suas notificações.

Para modelar o sistema foram levantadas as principais funcionalidades que o aplicativo deveria oferecer, para isso foram utilizados alguns diagramas UML, de forma a proporcionar uma melhor visualização e compreensão das atividades envolvidas no uso do aplicativo. A Figura 5 mostra os principais casos de uso realizados pela aplicação, são eles: realizar o *login*, configurar as preferências, visualizar relatórios das notificações de notas e do RU e visualizar os detalhes das disciplinas.

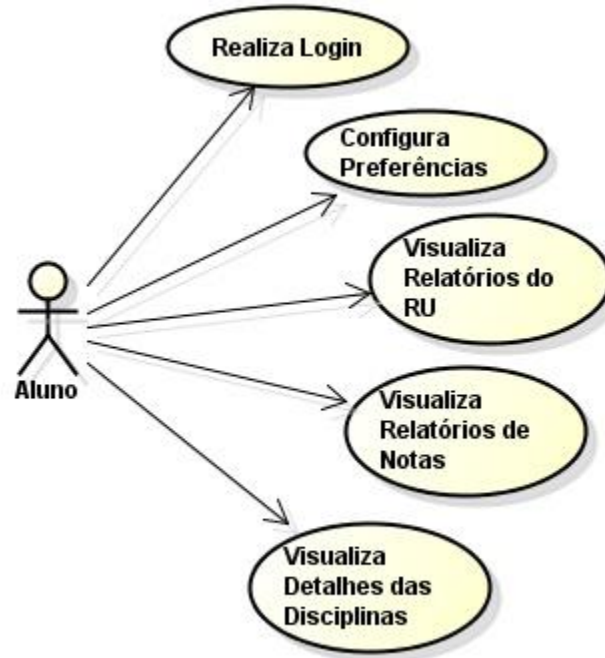


Figura 5 - Diagrama de Caso de Uso

É possível, ainda, representar o fluxo dos processos envolvidos nas tarefas executadas pelo aplicativo através de um Diagrama de Atividades. O aplicativo desenvolvido baseia-se em uma sequência de passos definida e exibida na Figura 6.

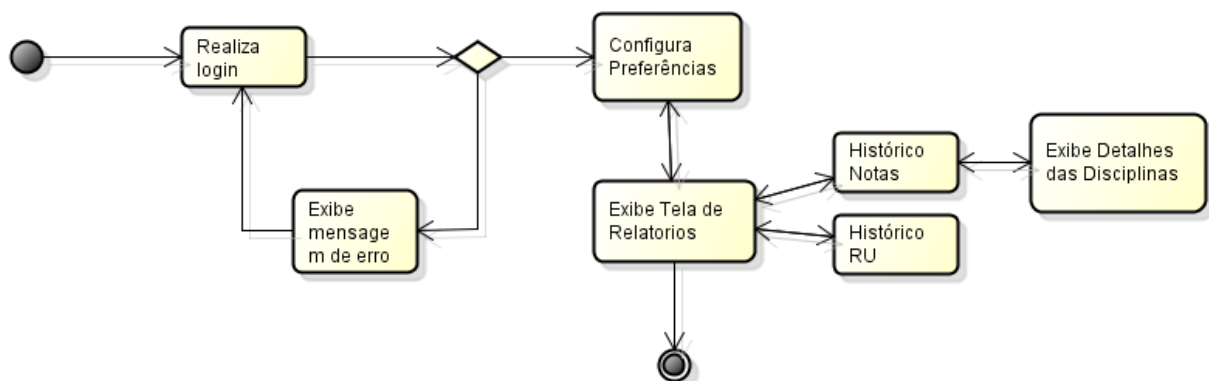


Figura 6 - Diagrama de Atividades

3.1.1 Requisitos do Sistemas

Uma importante etapa para o desenvolvimento de *softwares* é a análise e definição dos requisitos. Esta seção apresenta os requisitos básicos necessários ao sistema que servirão de base ao seu planejamento. Para isso, é feita a definição de quais requisitos são funcionais e quais requisitos não são. Por requisito entende-se uma condição ou capacidade com a qual o sistema deve estar de acordo. Abaixo explicamos a diferença entre eles.

Requisitos funcionais são fundamentais e definem as funções de um software ou parte dele. É o conjunto de entradas, seu comportamento e sua saída, ou seja, envolve cálculos, lógicas de trabalho, manipulação e processamento de dados, entre outros. Dentro dos requisitos funcionais também encontram-se a arquitetura do aplicativo, diferentemente da arquitetura técnica, que pertence aos requisitos não funcionais.

Requisitos não funcionais são relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, disponibilidade, segurança e tecnologias envolvidas. Muitas vezes, os requisitos não funcionais acabam gerando restrições aos funcionais. Com o crescente número de usuários vindos de dispositivos móveis, requisitos não funcionais como portabilidade e mobilidade tornam-se fundamentais em praticamente todos os sistemas web.

A Tabela 1 mostra os principais requisitos funcionais e não funcionais necessários para o bom funcionamento do sistema.

Tabela 1 – Requisitos Funcionais e Não Funcionais do Sistema

Requisitos Funcionais	Requisitos Não Funcionais
<ul style="list-style-type: none"> • Permitir a autenticação por meio de Login e senha; • Permitir que o usuário escolha as preferências para as notificações; • Permitir que o usuário exclua os históricos de notificações; • Permitir ao usuário a visualização do histórico das notificações. 	<ul style="list-style-type: none"> • Evitar erro por falta de conexão com a rede; • Autenticação do usuário.

Após a definição do sistema e dos requisitos, partiu-se para a implementação das funcionalidades básicas necessárias para o bom funcionamento da aplicação.

3.2 Experiências Preliminares

Para o desenvolvimento do aplicativo Android, optou-se pela IDE do Eclipse, pois possibilita que as aplicações sejam facilmente desenvolvidas com a ajuda de um *plug-in* chamado *Android Development Tools* (ADT).

A partir da escolha das ferramentas para o desenvolvimento do aplicativo, iniciaram-se estudos sobre criação e implementação de projetos na IDE de desenvolvimento. Partiu-se, então para a implementação e testes de funcionalidades básicas, onde se teve uma melhor compreensão de como funciona uma aplicação Android.

O objetivo desta seção é mostrar uma análise das funcionalidades básicas necessárias para o bom funcionamento do aplicativo.

3.2.1 Preferências

Muitas vezes é interessante que o usuário configure o comportamento do aplicativo de acordo com a sua preferência. Isso torna o aplicativo muito mais atrativo e personalizado.

Na aplicação é necessário usar algumas preferências com relação ao quadro de notas do aluno, como curso, ano e período. Também é necessário usar preferências para definir com que frequências o serviço de notificações será executado e se estará ativada ou não.

3.2.2 Implementar serviços de background

Para deixar o aplicativo mais interativo é necessário criar um serviço de background que faz pesquisas periódicas no servidor. Quando a nota de uma determinada disciplina for encontrada, ou quando o saldo do restaurante universitário for informado, será criada uma notificação na barra de status do dispositivo para alertar o usuário.

O *Service* é um componente da plataforma Android que pode executar tarefas de longa duração em segundo plano e que não possui interface gráfica, funcionando como uma *thread*. Ele pode ser iniciado por qualquer outro tipo de componente (uma *Activity* ou outro *Service*, por exemplo) e pode continuar em execução mesmo que o componente que o iniciou seja destruído. Alguns exemplos de uso de *Service* incluem fazer downloads, tocar uma música e acessar um serviço remoto [LECHETA, 2009].

3.2.3 Criar notificação na barra de *status*

A barra de status do Android é o local onde as notificações são exibidas. Essas notificações podem ser do próprio sistema, como, por exemplo, um aviso sobre o descarregamento da bateria, ou podem ter sido criadas por qualquer aplicativo. Quando o usuário selecionar alguma das notificações, devemos mostrá-la em uma nova *Activity* da aplicação. Uma *Activity* representa uma tela da aplicação e é responsável por tratar os eventos gerados nessa tela.

3.3 Implementação do aplicativo

Para implementação, foram utilizados todos os conceitos da sessão 3.1 e 3.2, criando classes para realizar operações que atendessem às necessidades do sistema.

Para permitir que a aplicação armazene informações do usuário, como o *login*, senha e histórico de cada notificação foi criado um banco de dados local para armazenar essas informações.

A classe que manipula o banco de dados se chama PortalAlunoDAO, e pode ser acessada por todas as outras classes quando necessário. O banco é utilizado principalmente para a verificação de atualizações através de comparações com as informações do *site*.

O padrão DAO (*Data Access Object*) é um padrão de projeto que abstrai e encapsula os mecanismos de acesso aos dados escondendo os detalhes da execução e a origem dos mesmos, criando uma interface de cliente genérica para fazer o acesso aos dados permitindo que os mecanismos de acesso sejam alterados independentemente do código que os utilizará. Esse padrão é muito usado em aplicações com banco de dados relacionais.

Existem diversas implementações do padrão DAO mas em geral pode-se relacionar algumas características de uma implementação:

- Todo o acesso aos dados deve ser feito através das classes DAO de forma a se ter o encapsulamento;
- Cada instância da DAO é responsável por um objeto de domínio;
- O DAO deve ser responsável pelas operações CRUD no domínio;

As classes necessárias para a implementação do aplicativo desenvolvido nesse trabalho, bem como sua estrutura, são descritas abaixo:

- **MainActivity:** é a classe principal da aplicação mostra a tela de login e chama a classe Consulta login para autenticar o usuário, se o login estiver correto chama a classe ConfiguracoesActivity para setar os parâmetros das preferências do aplicativo. Na próxima vez que o aplicativo for iniciado, a tela de login será omitida e a classe RelatoriosActivity será chamada;
- **ConsultaLogin:** estende a classe *AsyncTask*, que cria uma *thread* para realiza uma requisição *post* ao servidor, passando como parâmetro o login e a senha informados pelo usuário, salvando o login e a senha no banco de dados local para futuras interações;
- **ConfiguracoesActivity:** estende a classe *PreferenceActivity*, chama a classe CarregaParametros e retorna os parâmetros encontrados para a tela de preferências;

- **CarregaParametros:** estende a classe *AsyncTask*, que cria uma *thread* para realiza uma requisição get da página do quadro de notas ao servido e executa um *parser* para extrair as informações dos parâmetros pré-definidos, na página, como curso e período.
- **NotasService:** classe que estende a classe *Service*, realiza o serviço de notificações de notas e salva a nova nota no banco de dados local para exibir o histórico posteriormente;
- **RUService:** classe que estende a classe *Service*, realiza o serviço de notificações do saldo do RU e salva o novo saldo no banco de dados local para exibir o histórico posteriormente;
- **NotificaçãoTask:** classe que implementa *Runnable*, realiza uma requisição post ou get para recuperar uma página e realizar um *parser* para extrair as informações solicitadas;
- **RelatoriosActivity:** exibe uma tela com duas opções, acessar a lista do histórico das notificações do saldo do RU ou das notas. Também possibilita ao usuário editar as preferências ou sair do sistema;
- **RUActivity:** exibe uma tela com a lista do histórico das notificações do saldo do RU;
- **NotasActivity:** exibe uma tela com a lista do histórico das notificações de notas;
- **DetalhesNotasActivity:** essa classe exibe uma tela com os detalhes da notificação, referente ao quadro de notas, selecionada na barra de *status*, ou na lista de histórico das notificações.

A seguir, a Figura 7 mostra um diagrama de classes com as principais classes que ativam as funcionalidades do sistema.

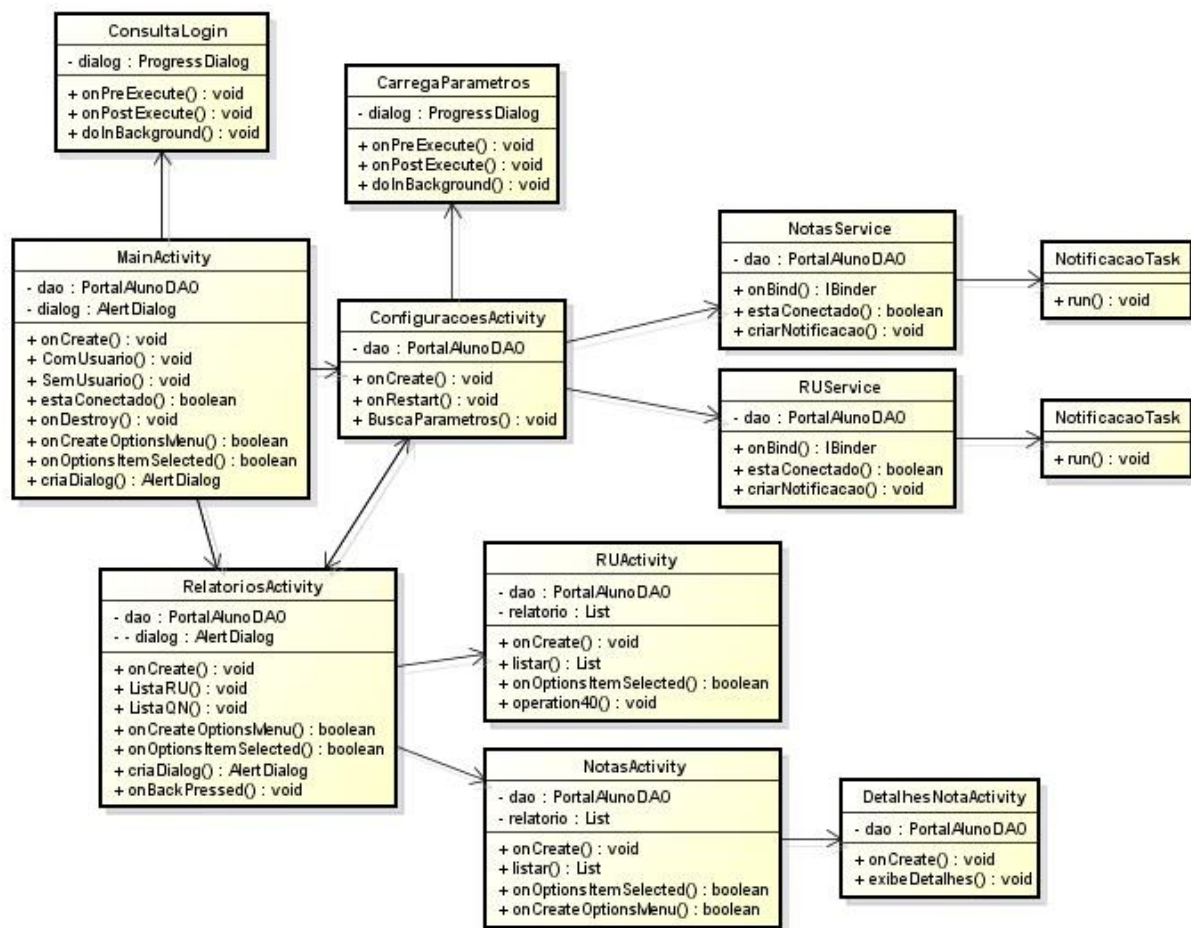


Figura 7 - Diagrama de Classes

Para desenvolver o aplicativo e realizar a captura de informações do servidor foi necessário entender como funciona a estrutura da linguagem HTML, pois essa é a linguagem utilizada na construção das páginas web do Portal do Aluno da UFSM.

3.3.1 Parser HTML

HTML (*Hypertext Markup Language*) não é uma linguagem de programação, mas sim uma linguagem especializada para a formatação de texto. O texto produzido fica entre símbolos especiais, denominados *tags*, que indicam sua aparência, e dentro destas *tags* são inseridos os comandos para fazer a formatação do texto. Tudo o que está fora da *tag* é um texto, portanto é exibido no corpo da página. A grande maioria das *tags* ocorre aos pares, delimitando o texto a ser

formatado, podendo ocorrer “aninhamento” de *tags*. Uma *tag* tem o seguinte formato: <nome da tag>Texto de Exemplo </nome da tag> [PILGRIM, 2011].

O código HTML é uma série de elementos em árvore onde alguns elementos são filhos de outros e assim por diante. O elemento principal dessa grande árvore é sempre a *tag* HTML (<html>) [PILGRIM, 2011].

Para realizar o *parser* da página HTML foi utilizado uma biblioteca chamada Jsoup, a qual foi incorporada ao projeto. Jsoup é um projeto *open source* distribuído sob a licença MIT e seu código fonte está disponível no GitHub. Esta é uma biblioteca Java que permite a leitura de uma página HTML e inclusive do novo HTML 5. O Jsoup trabalha como uma biblioteca XML (*Extensible Markup Language*) do tipo DOM, lendo as *tags* pelo tipo, podendo capturar além do conteúdo o valor de seus atributos [MBallem, 2014].

A página que contém o saldo do RU está estruturada em forma de tabela. A *tag* <table> define uma tabela em HTML e pode possuir um ou mais elementos <tr>, <th>, e <td>. O <tr> define uma linha da tabela, o elemento <th> define um cabeçalho da tabela, e o elemento <td> define uma célula da tabela.

O código utilizado para realizar o *parser* na página e capturar o do valor atual do saldo do RU foi o seguinte:

```
Document doc = Jsoup.parse(a.toString());
Elements elementos = doc.getElementsByTag("td");
String saldo = elementos.get(3).text();
ru = new RelatorioRU (data,saldo);
```

Nesse trecho de código são capturados todas as *tag* <td> existentes na página e guardadas em uma lista. Como o conteúdo que desejamos está na posição 3 dessa lista, somente esse valor é guardado em uma String, para ser utilizado posteriormente.

A página que contém o conteúdo relacionado ao quadro de notas do aluno está separada por *divs*, cada *div* possui uma classe diferente. A *tag* <div> define uma divisão ou uma seção em um documento HTML e é usada para agrupar blocos de elementos para formatá-los com CSS. A classe “accordion-group” define a divisão onde encontram-se as informações referentes a cada uma das disciplinas do aluno. Dentro dessa *div* encontram-se outros elementos como o nome, situação e notas de cada disciplina separados por *divs* de diferentes classes. A classe “span8” define as *divs*

com os nomes de cada disciplina. A classe “span4” define a situação de cada disciplina. Dentro da div pertencente a classe “accordion-group” tem uma tag <tbody>, onde estão as informações sobre as notas daquela disciplina. Essa *tag* é usado para agrupar o conteúdo do corpo em uma tabela HTML. O <tbody> elemento é usado em conjunto com os <thead> e <tfoot> elementos para especificar cada parte de uma tabela (corpo, cabeçalho, rodapé). Em cada célula dessa tabela estão as informações sobre as notas.

O código utilizado para realizar a captura das informações referentes ao quadro de notas de cada disciplina é o seguinte:

```
Document doc = Jsoup.parse(a.toString());
Elements elementos = doc.select("div[class=accordion-group]");
for(int x = 0; x < elementos.size(); x++){
    Elements e = elementos.get(x).select("div[class=span8]");
    String c = e.get(0).text();
    e = elementos.get(x).select("div[class=span4]");
    String s = e.get(0).text();
    e = elementos.get(x).select("tr");
    for(int y = 0; y < e.size(); y++){
        Elements e1 = e.get(y).select("td");
        for(int f = 0; f < e1.size(); f++){
            String n = e1.get(f).text();
        }
    }
}
```

A informações capturadas são salvas em estruturas e guardadas no banco de dados local da aplicação. Essas informações são utilizadas para serem comparas futuramente quando forem realizadas novas consultas ao servidor do site do Portal do Aluno, essas comparações definirão se uma nova notificação deve ser acionada ou não.

3.3.2 Controle de Notificações

Uma das preocupações ao utilizar aplicativos que necessitam de acesso constante à rede é o consumo excessivo de bateria do dispositivo móvel. A aplicação em questão necessita acessar os dados disponíveis num servidor remoto de tempos em tempos, em busca de novas atualizações. Uma alternativa encontrada para essa aplicação foi o uso de alarmes para programar as buscas por atualizações que necessitam de acesso à rede.

O controle de notificações é feito pelo usuário, ele pode definir nas preferências do aplicativo a frequência em que o serviço de notificações será executado. O código utilizado para programar esses acessos é o seguinte:

```
private void agenda (String service, int freq,){  
    Intent s = new Intent(service);  
    p = PendingIntent.getService(this, 0, s, 0);  
    Calendar c = Calendar.getInstance();  
    c.setTimeInMillis(System.currentTimeMillis());  
    c.add(Calendar.HOUR, freq);  
    alarme = (AlarmManager) getSystemService(ALARM_SERVICE);  
    long time = c.getTimeInMillis();  
    alarme.setRepeating(AlarmManager.RTC_WAKEUP, time, 60000, p);  
}
```

A função agenda é chamada na classe **RelatoriosActivity** passando como parâmetro uma *string* correspondente ao nome serviço e um inteiro que corresponde a frequência com que esse serviço será chamado. Desse modo o consumo de energia utilizada pela aplicação é reduzido.

3.4 Descrição das Interfaces

Após a finalização da codificação, obteve-se diversas telas como resultado do aplicativo criado e estas serão exibidas e detalhadas nesta sessão.

A Figura 8 mostra a tela de *login* do usuário, essa tela permite mostrar ao usuário informações acerca do aplicativo, como versão e desenvolvedor. O usuário deve inserir o *login* e a senha para autenticação, logo após, o sistema realiza a validação dos dados e guarda o *login* e a senha, caso corretos, em um banco de dados local para acessos futuros. Se os campos de *login* e a

senha estiverem em branco ou forem incorretos, o sistema emitirá um aviso informando ao usuário. A tela de login só será mostrada novamente se o usuário se desconectar do sistema.

Se a autenticação estiver correta o sistema exibirá a tela de preferências, como mostra a Figura 9. Nesta tela será possível definir alguns parâmetros para as notificações, como: escolher o curso, período e o ano, nos quais deseja-se consultar as disciplinas e suas respectivas notas, a frequência em que as notificações são emitidas e também possui a opção de ativar ou desativar as notificações.

Após a configuração dos parâmetros, será exibida a tela de opções, conforme a Figura 10. Esta será a tela exibida ao iniciar o aplicativo novamente, depois da autenticação. Nesta tela são mostrados dois botões, um para exibir os históricos de notificações de saldo e outro para exibir os históricos de notificações sobre as notas. Essa tela possui um menu que permite ao usuário voltar a tela de preferências, ver informações sobre o aplicativo e sair ou se desconectar do sistema.

Ao clicar o botão do Restaurante Universitário na tela de opções, será exibido um histórico das notificações sobre o saldo do RU, conforme a Figura 11. O mesmo acontece ao clicar no botão do Quadro de Notas, o histórico sobre as notificações de notas é exibido, como na Figura 12. Essas telas possuem um menu com a opção de limpar o histórico.

A Figura 13 mostra as notificações emitidas pelo sistema, e a Figura 14 mostra os detalhes de cada disciplina. Essa tela é mostrada quando o usuário clica em uma notificação ou seleciona alguma no histórico de notificações do quadro de notas.

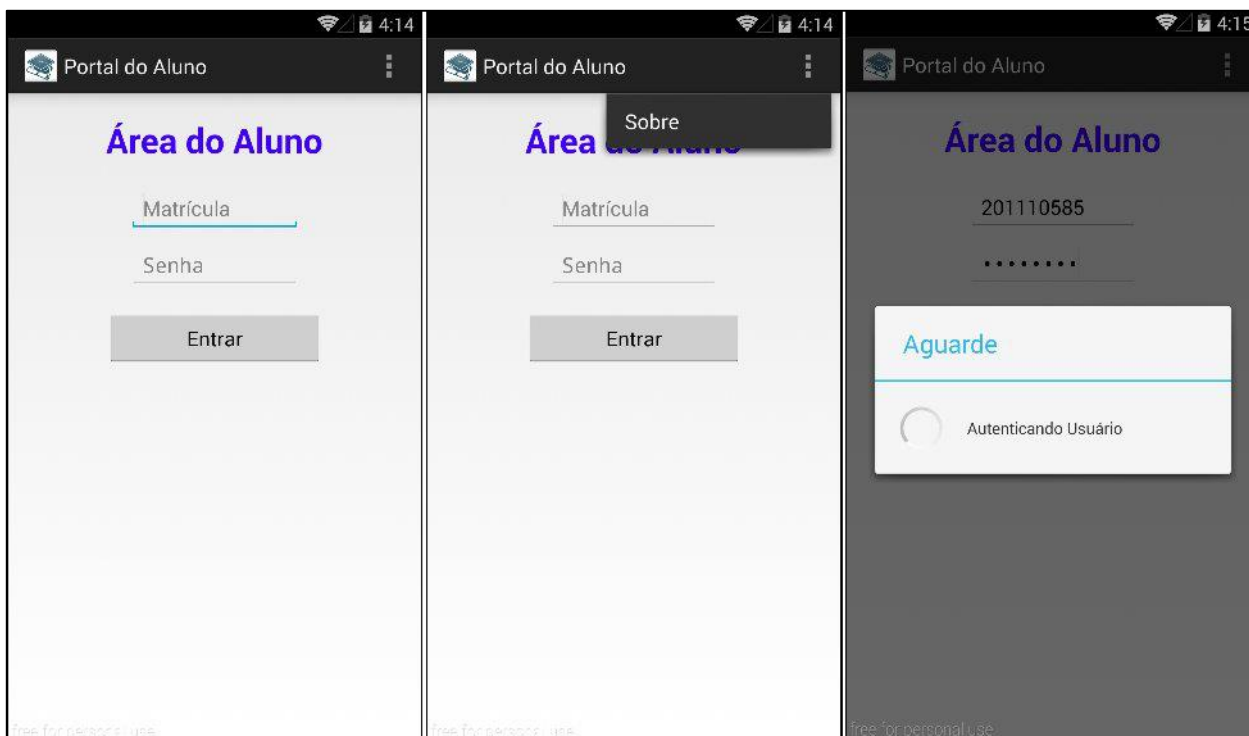


Figura 8 - Tela de login do aplicativo

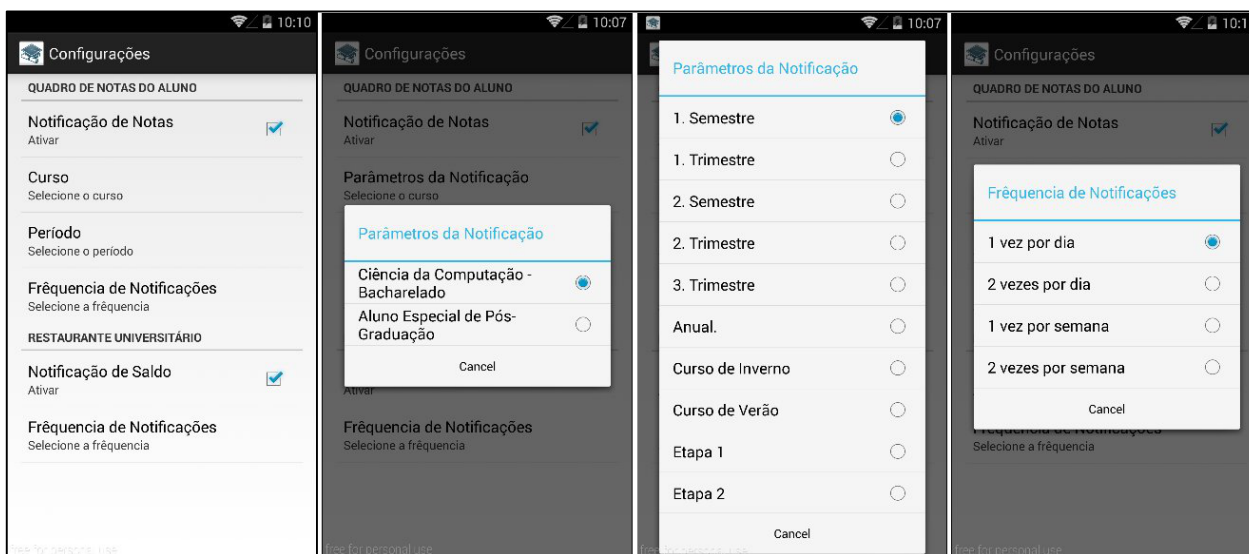


Figura 9 - Tela de Preferências

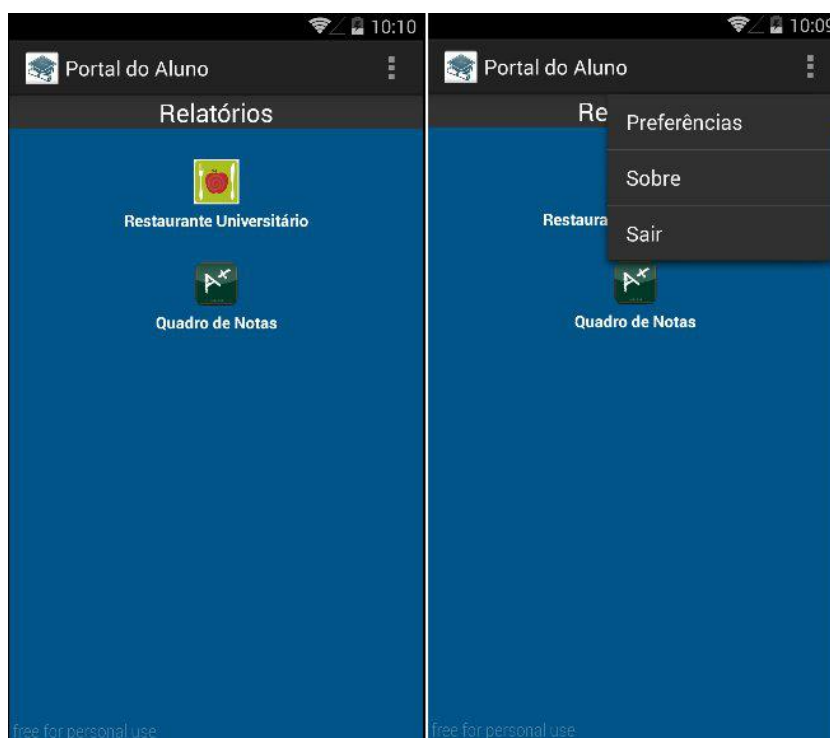


Figura 10 - Tela de opções

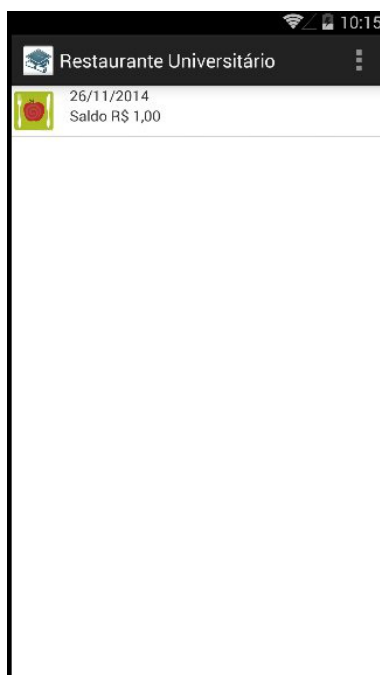


Figura 11 - Tela de histórico de notificações do saldo do RU

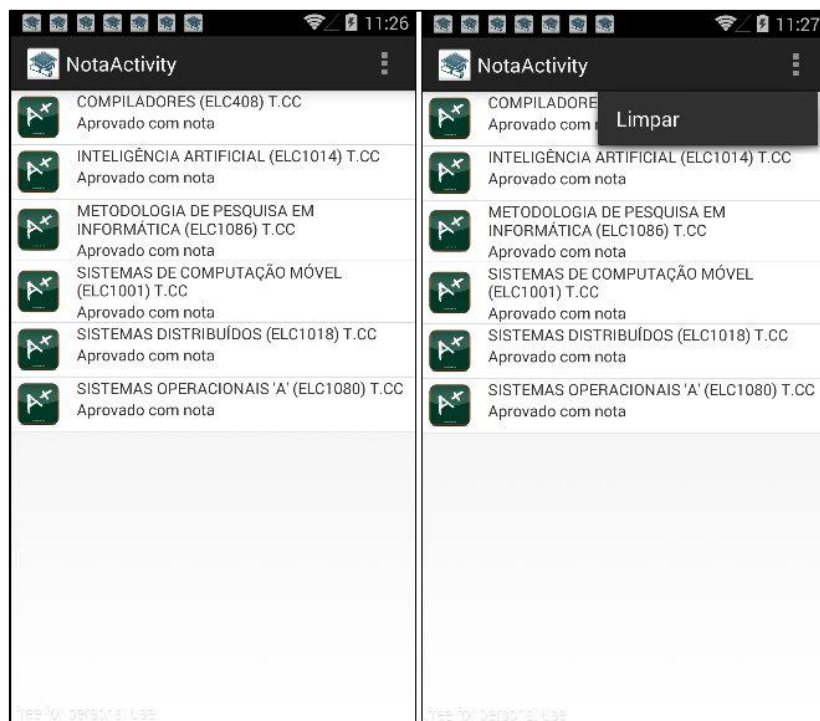


Figura 12 - Tela de histórico de notificações do quadro de notas

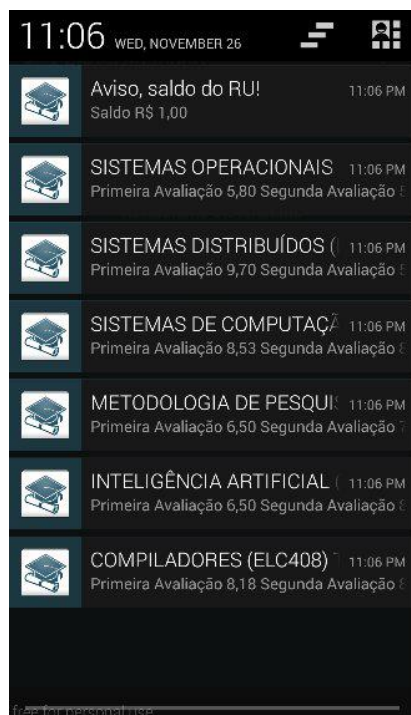


Figura 13 - Tela de notificações

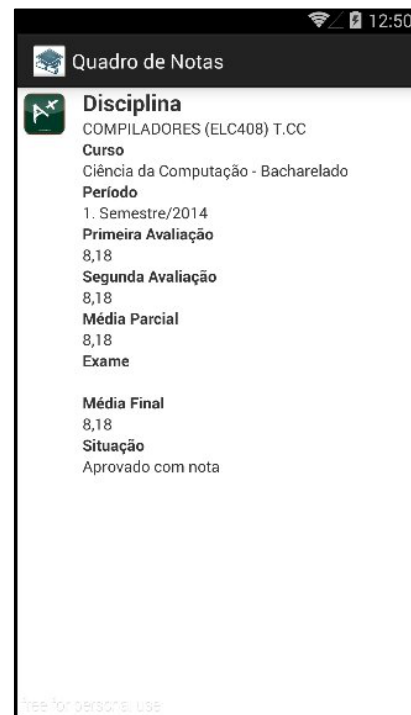


Figura 14 - Tela que detalha a disciplina selecionada

Após a criação as interfaces e o desenvolvimento das funcionalidades do sistema, iniciou-se uma série de testes validar a aplicação.

3.5 Testes de Funcionalidades

Após a conclusão da etapa de desenvolvimento do aplicativo, procurou-se investigar o desempenho e usabilidade do software, através de alguns testes, fazendo várias interações com a aplicação para testar a usabilidade do aplicativo.

Como o sistema está programado para emitir notificações a longo prazo, ou seja, conforme os eventos que modifiquem os valores do site forem ocorrendo, como a mudança do saldo após uma refeição ou compra de créditos, ou a adição de uma nova nota de uma determinada disciplina pelo professor, foram criados ambientes de testes para simular essas situações

3.5.1 Teste para notificação de saldo

Para testar a funcionalidade de emitir notificações sobre o saldo RU, o serviço responsável por essa operação foi modificado para simular em curto prazo as operações realizadas pelo sistema. Para isso, a frequência de verificações que o sistema faz ao site foi alterada no próprio código, para realizar verificações de 30 em 30 segundos.

Para simular os eventos de mudança de saldo, o sistema decrementa o valor informado pelo site antes de salvar as informações no banco local. Dessa forma, sempre que ele for verificar o saldo encontrará um valor diferente e emitirá uma nova notificação.

3.5.2 Teste para notificação de notas

O mesmo procedimento realizado para as notificações de saldo foi realizado para a testar a funcionalidade de verificação de notas.

Os valores encontrados no site foram modificados antes de serem salvos no banco de dados local, desta forma, sempre que for verificado novamente, será encontrado uma informação diferente da que está armazenada no banco, fazendo com que o sistema mostre uma nova notificação ao usuário.

3.5.3 Testes de autenticação

Para ter acesso às informações do site pelo aplicativo, também é necessário realizar a autenticação do usuário. O sistema faz a autenticação enviando uma solicitação para o servidor web, se a solicitação for bem-sucedida sua execução continua normalmente. Caso o usuário digite alguma informação errada, impedindo o acesso às informações do site, ou deixe algum campo em branco, uma mensagem de erro é exibida na tela.

3.5.4 Testes de conectividade

Antes de realizar qualquer tipo de comunicação que necessite do uso da rede, o sistema verifica se o dispositivo está conectado, se estiver, as requisições são executadas normalmente, caso contrário uma mensagem é exibida na tela informando ao usuário que ele precisa se conectar. Essa mensagem é mostrada apenas quando o usuário deseja realizar o *login*, mas não possui conexão, na execução dos serviços a mensagem não é exibida.

4 CONCLUSÃO

Neste trabalho foram apresentados conceitos sobre o desenvolvimento de aplicativos para dispositivos móveis, mais especificamente para a plataforma móvel Android. Acompanhou-se o desenvolvimento e implementação de um protótipo de notificações focado no acesso aos dados do Portal do Aluno da UFSM, mais especificamente do quadro de notas e do saldo do RU.

Durante a implementação do aplicativo, os desenvolvedores do *site* do Portal do Aluno da UFSM realizaram algumas mudanças no *layout* do *site* e na forma como este recuperava as informações da base de dados no servidor. Essas mudanças ocasionaram problemas no desempenho do aplicativo, pois o *parser* utilizado para filtrar as informações da página carregada, não atendia as necessidades da nova implementação do *site*. Para resolver esse problema, foi necessário refazer o *parser*, de modo a se adequar ao novo formato do *site*.

Apesar de certos imprevistos no momento de implementação, tendo em vista os objetivos primários deste trabalho, seu resultado pode ser considerado satisfatório, pois o aplicativo implementado possibilita ao seu usuário a busca/consulta de informações acerca das notas disponível no Portal do Aluno da UFSM e sobre o saldo do Restaurante Universitário, sem que o usuário precise iniciar o aplicativo toda vez que quiser obter essas informações.

Espera-se que o aplicativo construído sirva de base para a continuação do desenvolvimento de novas funcionalidades para o acesso aos dados do Portal do Aluno. Além disto, em um futuro próximo, pretende-se refinar a maneira como os dados são capturados, bem como adicionar novas funcionalidades e melhorias na interface do usuário.

Como possíveis trabalhos futuros a serem realizados é importante priorizar a economia dos recursos do sistema utilizados pelo aplicativo, para torná-lo mais eficiente. Isso seria possível com a implementação de notificações do tipo *push*. As notificações do tipo *push* usam um canal de comunicação já existente, e funcionam no caminho contrário. Na aplicação desenvolvida é o celular que contata o servidor. No *push*, é o servidor que avisa o celular que existe uma nova atualização. Uma aplicação que utilize o *push* estará na memória do dispositivo, mas ficará num estado suspenso até que receba alguma mensagem do servidor. Dessa forma não há gasto de bateria, nem uso desnecessário da rede, pois a aplicação só irá contatar o servidor depois que receber uma mensagem avisando que existem novas atualizações, deixando a aplicação muito mais eficiente.

5 REFERÊNCIAS

INTEL CORPORATION. **O conceito de mobilidade.** Disponível em: <http://www.nextgenerationcenter.com/detalle-curso/Mobilidade.aspx>>. Acesso em: setembro de 2014.

BENJAMIN, Speckmann. **The Android Mobile Platform.** A Paper Submitted to the Eastern Michigan University. 2008.

MATEUS, G. R.; Loureiro, A. A. F. **Introdução a Computação Móvel. 11ª Escola de Computação,** Rio de Janeiro, 1998.

POSSER, N. L. **Computação Móvel e Learning: Estudo e Construção de um Protótipo para Smartphone,** Porto Alegre, 2006.

GOOGLE PLAY STORE. Disponível em: <https://play.google.com/store?hl=pt_BR>. Acesso em: julho de 2014.

ABLESON, W. Frank et al. **Android em Ação.** 3. ed. Rio de Janeiro: Campus, 2012.

DEVELOPERS, ANDROID. **Android, the world's most popular mobile platform.** Disponível em: <<http://developer.android.com/about/index.html>>. Acesso em: julho de 2014.

IBM. **Socket programming.** Disponível em: <<http://publib.boulder.ibm.com/iserics/v5r1/ic2924/info/rzab6/rzab6mst.pdf>>. Acesso em: setembro de 2014.

LECHETA, Ricardo R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK,** 2ª Edição. São Paulo: Novatec, 2009.

OHA et. al. **Building a Better phone for consumers**. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: julho de 2014.

DEVMEDIA. **Introdução ao desenvolvimento de aplicações web**. Disponível em: <<http://www.devmedia.com.br/introducao-ao-desenvolvimento-de-aplicacoes-web/29798#ixzz3GAg9TOPf>>. Acesso em: outubro de 2014.

NETMARKETSHARE. **Mobile/Tablet Top Operating System Share Trend**. Disponível em: <<http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1&qpct=4&qpsp=175&qpnnp=12&qptimeframe=M>> Acesso em: outubro de 2014.

APACHE. **The Apache Software Foundation**. Disponível em: <<http://www.apache.org/>> Acesso em: agosto de 2014.

Google Play Store. **Aplicativo UFSM Mobile**. Disponível em: <https://play.google.com/store/apps/details?id=com.multiweb.ufsm.app&hl=pt_BR>, acesso em julho de 2014.

Portal RU. **Página de login do Restaurante Universitário**. Disponível em : <<http://portal.ufsm.br/ru/login.jsp>>, acesso em agosto de 2014.

MBallem. **Programando com Java**. Disponível em: <http://www.mballem.com/post/capturando-conteudo-html-com-jsoup/>, acesso em novembro de 2014.

FIGUEIREDO, Carlos M. S. NAKAMURA, Eduardo. **Computação Móvel: Novas Oportunidades e Novos Desafios**. Disponível em: <https://portal.fucapi.br/tec/imagens/revistas/ed002_016_028.pdf>. Acesso em novembro de 2014.

JOHNSON, Thienne M. **JAVA para Dispositivos Móveis: Desenvolvendo Aplicações com J2ME** Editora Novatec. Acesso em novembro de 2014.

KANTORKI, Gustavo Z., MORAIS, Tiago G., MOREIRA, Viviane P., HEUSER, Carlos A. **Choosing Values for Text Fields in Web Forms**. Porto Alegre, 2013.

MORAES, Tiago Guimarães. **Seleção de Valores para Preenchimento de Formulários Web**. Porto Alegre: PPGC da UFRGS, 2013.

PILGRIM, MARK. **HTML 5 – ENTENDENDO E EXECUTANDO**. Editora Alta Books, Brasil, 2011.