
Universidade Federal de Mato Grosso do Sul
Campus do Pantanal
Curso de Sistemas de Informação

**UFMS Mobile: Ferramenta para auxílio acadêmico na
Universidade Federal de Mato Grosso do Sul**

Luiz Henrique Quevedo Lima

Me. Luciano Édipo Pereira da Silva(Orientador)

Março de 2016

UFMS Mobile: Ferramenta para auxílio acadêmico na Universidade Federal de Mato Grosso do Sul

Luiz Henrique Quevedo Lima

Este exemplar corresponde à redação final da monografia da disciplina Trabalho de Conclusão de Curso II devidamente corrigida e defendida por Luiz Henrique Quevedo Lima e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Sistemas de Informação.

Corumbá, 29 de Março de 2016.

Me. Luciano Édipo Pereira da Silva
(Orientador)

UFMS Mobile: Ferramenta para auxílio acadêmico na Universidade Federal de Mato Grosso do Sul

Luiz Henrique Quevedo Lima

Março de 2016

Banca Examinadora:

- Me. Luciano Édipo Pereira da Silva (Orientador)
- Ma. Lucineide Rodrigues da Silva
- Me. Anderson Pereira das Neves

Resumo

O Android é o sistema operacional para smartphones que mais cresce no mundo. O Android está presente em mais smartphones do que qualquer outra plataforma atualmente. Por ser uma plataforma aberta, o desenvolvimento de aplicações para Android torna-se muito vantajoso e desafiador, pois a plataforma cresce em ritmo acelerado. Este trabalho apresenta o desenvolvimento de um aplicativo Android para comunicação entre professores e alunos da Universidade Federal de Mato Grosso do Sul (UFMS), sincronizado com uma aplicação web que gerencia o conteúdo e um *web service* que permite comunicação entre plataformas. Este projeto baseia-se nas mais rígidas métricas do Material Design e implementação propostas pelo Google, além de apresentar uma série de bibliotecas e APIs que contribuem na criação de uma interface de usuário com qualidade.

Palavras-chave: Android, smartphones, UFMS, material design.

Abstract

Android is the leading Operating System for smartphones in the world. Android is present in more smartphones than any other platform today. Being an open source platform, the Android application development has become very advantageous and challenger because the platform grows in a fast pace. This paper presents the development of an application for communication between professors and students of Universidade Federal de Mato Grosso do Sul (UFMS), synchronizing with a content manager web application and a web service that allows cross-platform communication. This project is based on strict Material Design and development guidelines proposed by Google, Inc., in addition to presenting numerous libraries and APIs which contributed on creating a quality user interface.

Keywords: Android, smartphones, UFMS, material design.

Agradecimentos

Agradeço primeiramente a Deus, que tornou todo este processo possível. À minha família que me proporcionou o apoio necessário para que conseguisse realizar esta etapa dos meus estudos. A todos que me auxiliaram de alguma forma para que fosse possível a realização deste trabalho, em especial à minha namorada, por ter me apoiado em todas as fases da minha formação, sempre que precisei, e meu orientador Luciano Édipo Pereira da Silva pelo excelente suporte que me ofereceu no desenvolvimento deste trabalho. Agradeço a vocês.

Tabela de Siglas e Símbolos

ADT	Application Development Tools
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Android Package
AS	Android Studio
ASF	Apache Software Foundation
AVD	Android Virtual Device
DC	Diagrama Comparativo
DDMS	Dalvik Debug Monitor Server
DER	Diagrama Entidade-Relacionamento
DES	Data Encryption Standard
DFD	Diagrama de Fluxo de Dados
DVCS	Distributed Version Control System
DVM	Dalvik Virtual Machine
FTP	File Transfer Protocol
GB	Gigabyte
GPS	Global Position System
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IES	Instituição de Ensino Superior
JDK	Java Development Kit
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MB	Megabyte
MDL	Material Design Lite
MVC	Model View Controller

OHA	Open Handset Alliance
OO	Orientação a Objetos
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
POJO	Plain Old Java Object
REST	Representational State Transfer
SDK	Software Development Kit
SGA	Sistema de Gestão Acadêmica
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistemas de Informação
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TI	Tecnologia da Informação
3G	Third Generation
UI	User Interface
USB	Universal Serial Bus
UML	Unified Modeling Language
VCS	Version Control System
XML	Extensible Markup Language
WIFI	Wireless Fidelity
W3C	World Wide Web Consortium

Sumário

Resumo	iv
Abstract	v
Agradecimentos	vi
Tabela de Siglas e Símbolos	vii
1 Introdução	1
1.1 Justificativa	2
1.2 Objetivos	3
1.2.1 Objetivos Específicos	3
1.3 Organização do Trabalho	5
2 Referencial Teórico	6
2.1 Dispositivos Móveis	6
2.2 Computação em Nuvem	6
2.3 Gerenciamento de Alunos	7
2.4 Aplicativos Relacionados	7
2.4.1 Aplicativo Android para o Ambiente Univates Virtual	7
2.4.2 UFPE Mobile	8
2.4.3 PUC Minas Mobile	8
3 Metodologia	9
3.1 Processo de desenvolvimento unificado	9
3.2 Ferramentas utilizadas	9
3.2.1 Astah Professional	10
3.2.2 Android Studio	10
3.2.3 JetBrains PhpStorm	10
3.3 Bibliotecas e APIs	11
3.3.1 Google Play Services	11
3.4 Linguagens Utilizadas	12
3.4.1 Java	12

3.4.2	PHP: Hypertext Preprocessor (PHP)	13
3.4.3	Extensible Markup Language (XML)	14
3.4.4	JavaScript Object Notation (JSON)	14
3.5	Model, View e Controller (MVC)	14
3.6	Open Handset Alliance	15
3.7	Android	15
3.7.1	Arquitetura	16
3.8	Dalvik Virtual Machine	17
3.9	Notificações	17
3.10	Activity e Intents	18
3.10.1	Activity	18
3.10.2	Ciclo de vida das activities	18
3.10.3	Intents	20
3.11	Fragments	21
3.12	Armazenamento	22
3.12.1	Banco de dados SQLite 3	22
3.12.2	MySQL	23
3.12.3	Shared Preferences	23
3.12.4	Preferences Framework	23
3.13	Compatibilidade	24
3.13.1	Support Library	24
3.14	Layouts	25
3.14.1	LinearLayout	25
3.14.2	RelativeLayout	26
3.14.3	FrameLayout	26
3.15	Material Design	26
3.15.1	CardView	27
3.15.2	RecyclerView	27
3.15.3	TextInputLayout	27
3.15.4	Snackbar	28
3.15.5	AppBar	28
3.15.6	CoordinatorLayout	28
3.16	Navegação	29
3.16.1	Navigation View	29
3.16.2	Up Navigation	29
3.16.3	Layout com abas usando TabLayout	30
3.17	Rede	30
3.17.1	Web Services	30
3.17.2	Padrão REST	31
3.17.3	RESTful Web Service com PHP	32
3.17.4	Acesso à rede eficiente com Volley	32

3.17.5	Tarefas assíncronas	33
3.18	Segurança	33
3.18.1	Encrytação de dados armazenados	34
3.18.2	Protegendo consultas no banco de dados	35
3.18.3	Permissões do Android	35
3.18.4	Permissões em tempo de execução	36
3.19	Google Cloud Messaging	36
3.19.1	Arquitetura	37
3.20	Gradle	38
3.20.1	Dependências	39
3.21	Alocação dinâmica de recursos	39
3.21.1	Recursos de Imagem	39
3.21.2	Orientação	40
3.21.3	Versões	41
3.21.4	Suporte a Tablets	42
3.22	Ferramentas para web	42
3.22.1	Material Design Lite (MDL)	42
3.22.2	Materialize	42
3.22.3	Data Table	43
4	UFMS Mobile	44
4.1	Roteiro de Desenvolvimento	44
4.1.1	Visão Geral	44
4.1.2	Requisitos	44
4.1.3	Modelos Criados	45
4.1.4	Protótipos da aplicação	48
4.2	Aplicativo UFMS Mobile	58
4.2.1	Ambiente de Instalação	58
4.2.2	Processo de Desenvolvimento	61
4.2.3	Arquitetura do Projeto	62
4.2.4	Interface Gráfica	62
4.2.5	Dispositivos Utilizados	63
4.2.6	Permissões	64
4.2.7	Preparação da Aplicação Cliente	64
4.2.8	Dependências do projeto	65
4.2.9	Aplicação Web	67
4.2.10	Iterações de design	68
4.3	Funcionalidades da aplicação	68
4.3.1	Inserindo um evento na aplicação	69
4.3.2	Acessando menu principal do aplicativo	77
4.3.3	Avaliando disciplina cursada	78

4.3.4	Visualizando notas disponíveis	82
4.3.5	Acessando configurações do aplicativo	85
4.3.6	Exibindo lista de eventos	87
4.3.7	Licença de Utilização	91
4.3.8	Repositório de Código	91
4.4	Considerações e Validação	92
4.4.1	Adaptando para tablets	92
4.4.2	Suportando diversas versões do Android	93
4.4.3	Desempenho no acesso à rede	96
4.4.4	Validando design da aplicação	96
5	Considerações Finais	98
5.1	Contribuições	98
5.2	Limitações	99
5.3	Trabalhos Futuros	99
A	Engenharia de requisitos	104
A.1	Regras de negócios	104
A.2	Lista de requisitos	105
A.3	Documento de requisitos	107
A.3.1	Escopo	107
A.3.2	Restrições	108
A.3.3	Dependências	108
A.3.4	Descrição geral do sistema	109
A.3.5	Requisitos de software	109
B	Diagramas	127
B.0.6	Casos de Uso	127
B.0.7	Diagrama de Classe	128
B.0.8	Diagrama Relacional	129
B.0.9	Pacotes do projeto	130
C	Desenvolvimento Iterativo	131
D	Bibliotecas e Ferramentas Utilizadas	136
D.1	Picasso	136
D.2	DSpec	136
D.3	WilliamChart	137
D.4	Material Calendar View	137
D.5	Text Drawable	137
D.6	Adobe Photoshop	137
D.7	Justinmind Prototyper	137

Lista de Tabelas

3.1	Dependências da biblioteca de Support.	25
3.2	Densidades de tela.	40
3.3	Diretórios de layouts e descrição.	40
3.4	Pastas values para diferentes versões da aplicação.	41
3.5	Pasta contendo recursos para tablets.	42
4.1	Dispositivos usados no desenvolvimento.	64
4.2	Lista das permissões utilizadas.	64
4.3	Lista de dependências das bibliotecas utilizadas.	66
A.1	Escopo funcional da aplicação.	107
A.2	Fora do escopo deste trabalho.	108
A.3	Restrições do projeto.	108
A.4	Dependências do projeto.	109
A.5	Requisito funcional RFN01.	110
A.6	Requisito funcional RFN02.	110
A.7	Requisito funcional RFN03.	111
A.8	Requisito funcional RFN04.	111
A.9	Requisito funcional RFN05.	112
A.10	Requisito funcional RFN06.	112
A.11	Requisito funcional RFN07.	113
A.12	Requisito funcional RFN08.	113
A.13	Requisito funcional RFN09.	114
A.14	Requisito funcional RFN10.	114
A.15	Requisito funcional RFN11.	115
A.16	Requisito funcional RFN12.	115
A.17	Requisito funcional RFN13.	116
A.18	Requisito funcional RFN14.	116
A.19	Requisito funcional RFN15.	117
A.20	Requisito funcional RFN16.	117
A.21	Requisito funcional RFN17.	118
A.22	Requisito funcional RFN18.	118
A.23	Requisito funcional RFN19.	119

A.24 Requisito funcional RFN20.	119
A.25 Requisito funcional RFN21.	120
A.26 Requisito funcional RFN22.	120
A.27 Requisito funcional RFN23.	121
A.28 Requisito funcional RFN24.	121
A.29 Requisito funcional RFN25.	122
A.30 Requisito funcional RFN26.	122
A.31 Requisito funcional RFN27.	123
A.32 Requisito funcional RFN28.	123
A.33 Requisito funcional RFN29.	124
A.34 Requisito funcional RFN30.	124
A.35 Requisito funcional RFN31.	125
A.36 Requisitos não funcionais de usabilidade.	125
A.37 Requisitos não funcionais de segurança.	126
C.1 Atividades executadas na iteração 1.	131
C.2 Atividades executadas na iteração 2.	132
C.3 Atividades executadas na iteração 3.	132
C.4 Atividades executadas na iteração 4.	132
C.5 Atividades executadas na iteração 5.	133
C.6 Atividades executadas na iteração 6.	133
C.7 Atividades executadas na iteração 7.	133
C.8 Atividades executadas na iteração 8.	134
C.9 Atividades executadas na iteração 9.	134
C.10 Atividades executadas na iteração 10.	134
C.11 Atividades executadas na iteração 11.	135
C.12 Atividades executadas na iteração 12.	135
C.13 Atividades executadas na iteração 13.	135

Lista de Figuras

1.1	Fatia do mercado de smartphones utilizando o Android.	2
3.1	Linguagens de programação mais utilizadas.	12
3.2	Funcionamento da linguagem PHP.	13
3.3	Arquitetura do Android.	16
3.4	Ciclos de vida de um activity.	19
3.5	Esquema exemplificando o RESTful Web Service.	31
3.6	Arquitetura do Google Cloud Messaging.	38
3.7	<i>Ripple effect</i> ao clicar em um item, disponível a partir da versão 5.0.	41
4.1	Casos de uso dos alunos.	45
4.2	Casos de uso dos professores.	46
4.3	Trecho do diagrama relacional – tabela evento e suas ligações.	47
4.4	Trecho do diagrama relacional – tabela disciplina e suas ligações.	47
4.5	Protótipos das telas de acesso à aplicação Android: Login e Cadastrar.	48
4.6	Protótipos das telas de detalhes da disciplina.	49
4.7	Primeira e segunda iteração dos protótipos da tela de detalhes de eventos.	50
4.8	Protótipos das telas de lista de disciplinas e eventos.	51
4.9	Protótipos das telas de lista de disciplinas quando estiverem vazias.	52
4.10	Protótipos das telas de lista de eventos quando estiverem vazias.	52
4.11	Protótipos das telas de acesso às notas do aluno.	53
4.12	Protótipos das telas inicial e menu de opções.	54
4.13	Protótipos da tela de configuração da aplicação.	55
4.14	Protótipos da página inicial da aplicação web.	56
4.15	Protótipos da página de eventos da aplicação web.	57
4.16	Protótipos da tela de adicionar evento.	58
4.17	Tela de configuração do SDK Manager.	59
4.18	Configurações disponíveis de aparelhos Android.	60
4.19	Imagens do sistema Android disponíveis para uso.	61
4.20	Arquitetura do projeto.	62
4.21	Logomarca da aplicação Android em diferentes resoluções.	66
4.22	Página inicial da aplicação web.	67
4.23	Lista de eventos disponíveis na aplicação web.	68

4.24	Página para adicionar eventos na aplicação web.	69
4.25	Notificação recebida pelo aplicativo Android.	71
4.26	Tela com detalhes do evento recém criado.	72
4.27	Tela com documento de anexo do evento recém criado.	73
4.28	Permissão para gravar no armazenamento externo do telefone.	74
4.29	Fazendo download do anexo.	75
4.30	Abrindo documento baixado.	76
4.31	Fazendo download de anexo via dados móveis.	77
4.32	Botão “sanduíche” da barra de tarefas para abrir o menu principal.	77
4.33	Menu principal da aplicação posicionado no lado esquerdo da tela.	78
4.34	Avaliando disciplina.	79
4.35	Confirmando e enviando nota – 1 a 5 – para servidor.	80
4.36	Editar avaliação de disciplina realizada anteriormente.	81
4.37	Média de nota das avaliações realizadas pelos usuários.	82
4.38	Disciplinas em que o aluno está matriculado.	83
4.39	Atividades com notas disponíveis em determinada disciplina.	84
4.40	Modo de visualização de notas para análise de desempenho com gráficos.	85
4.41	Monitorando alterações nas preferências do usuário na tela de configurações.	86
4.42	Aplicando preferências do usuário no comportamento da aplicação.	87
4.43	Lista de eventos disponíveis na aplicação Android.	89
4.44	Buscando um evento específico na lista de eventos.	90
4.45	Exibindo eventos em modo calendário.	91
4.46	Aplicação não adaptada executando em tablet de 9 polegadas.	93
4.47	Layout da aplicação adaptado para executar em tablet.	93
4.48	Diagrama Comparativo 1 – Lista de eventos API 19 e API 23.	94
4.49	Diagrama Comparativo 2 – Modo desempenho do aluno API 19 e API 23.	95
4.50	Diagrama Comparativo 3 – Detalhes do evento API 19 e API 23.	95
4.51	Desempenho da aplicação ao fazer sincronia com o servidor através da internet.	96
4.52	Design da aplicação em diferentes plataformas.	97
B.1	Diagrama de casos de uso do projeto.	127
B.2	Diagrama de classes do projeto.	128
B.3	Diagrama relacional de banco de dados do projeto.	129
B.4	Pacotes criados no desenvolvimento deste projeto.	130
E.1	Iterações de design da tela de configurações.	139
E.2	Iterações de design da tela de lista de disciplinas.	140
E.3	Iterações de design da tela de detalhes do evento.	140
E.4	Iterações de design da tela inicial (Explore).	141
E.5	Iterações de design da tela de acesso: login.	141
E.6	Iterações de design do menu lateral da aplicação.	142

Lista de Algoritmos

1	Trecho de código do arquivo de configuração AndroidManifest.xml	65
2	Trecho do array de JSON com os eventos disponíveis retornado pelo servidor, disponível em: http://www.henriqueweb.com.br/webservice/list/listEventos.php	88

Capítulo 1

Introdução

Com o avanço da tecnologia e a adesão dos dispositivos móveis como meio de comunicação, surgiu uma nova possibilidade de criação de sistemas mais flexíveis e uma nova experiência do usuário final. Anteriormente os processos envolvendo professores e acadêmicos de universidades eram realizados através do computador, com portais na internet, ou diretamente via e-mail. As aplicações web não garantem o acesso do usuário, uma vez que necessitam ser alertados por uma aplicação de terceiros sempre que houver atualização no sistema ou que estes usuários estejam constantemente acessando o portal. Os e-mails apresentam as informações sem uma forma de filtro eficiente ou lotam a “caixa de entrada” com informações indesejadas.

Este trabalho tem como objetivo desenvolver uma ferramenta de auxílio na comunicação entre professores e alunos da Universidade Federal de Mato Grosso do Sul (UFMS) inicialmente, mas podendo ser configurada para ser utilizada em outras instituições de ensino. Através da aplicação, os professores poderão disponibilizar conteúdo de forma rápida através de uma aplicação web, facilitando a interação com os acadêmicos, que terão acesso a partir de um aplicativo para dispositivos Android. Os acadêmicos poderão ser informados de atualizações relacionadas às suas disciplinas cursadas no momento e anteriores, além da possibilidade de filtrar o conteúdo.

Esta aplicação possibilita que alunos recebam conteúdo acerca das disciplinas em que estão matriculados de qualquer lugar, através de um smartphone ou tablet. Também permite que os professores mantenham o controle das disciplinas que lecionam, além de manter os alunos conectados e atualizados.

Por meio deste trabalho pretende-se criar uma aplicação padrão para acesso a conteúdo de disciplinas das instituições de ensino. Para os professores será útil para manter os alunos engajados em suas disciplinas e receber feedback da qualidade das suas aulas ministradas através do mecanismo de avaliação de disciplinas. Para os alunos, poder receber em seus dispositivos móveis, com o conforto que a mobilidade oferece, atualizações do seu curso ou instituição de ensino.

Para as instituições em geral seria muito vantajoso usufruir de uma aplicação customizável e gratuita, que pode ser aplicada em qualquer universidade. Este projeto facilitaria

a comunicação entre aluno e professor daquelas universidades que não possuem ferramenta padronizada para tal, e tornaria uma opção de baixo custo para outras que desejem migrar para um sistema configurável.

1.1 Justificativa

O Android é o sistema operacional que mais cresce no mundo todo. Presente em 8 a cada 10 smartphones, este fenômeno tecnológico nos apresenta um leque de possibilidades. Segundo Vincent (2015), o Android oficialmente ultrapassou a marca dos 1,4 bilhões de usuários, sendo 300 milhões utilizando versão 5.0 (Lollipop) do sistema. Este número representa 82% dos smartphones no mundo utilizando dispositivos com Android segundo a International Data Corporation (2015), como mostrado na figura 1.1.

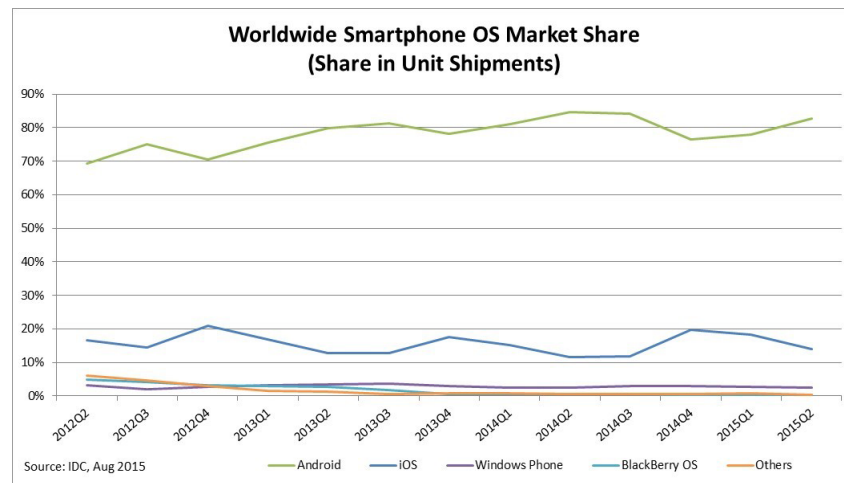


Figura 1.1: Fatia do mercado de smartphones utilizando o Android.

Fonte: International Data Corporation (2015)

Outro dado interessante é o número de aplicações baixadas no Google Play Store¹. Segundo Lorenzoni (2016), em 2015 cerca de 200 milhões de aplicativos foram baixados na loja do Android. Ainda segundo a pesquisa, o principal motivo deste crescimento é a popularização do Android em mercados emergentes “Brasil, Índia, Indonésia, México e Turquia somam praticamente metade dos downloads de Android em todo o mundo (100 milhões), sendo locais fundamentais para o aumento mundial”.

O acesso à internet é um dos fatores que impulsiona esta popularização, pois segundo pesquisa do governo federal, os brasileiros ficam mais tempo conectados do que assistindo

¹Google Play Store é a loja oficial de aplicativos do Android :<https://play.google.com/>.

televisão. Segundo o Portal Brasil (2014)² “a escolaridade e a idade dos entrevistados são os fatores que impulsionam a frequência e a intensidade do uso da internet no Brasil”. Dentre os usuários com ensino superior, 72% acessam a internet todos os dias.

Como acadêmico do curso de Sistemas de Informação da Universidade Federal de Mato Grosso do Sul (UFMS), percebeu-se a oportunidade de tornar simples a relação entre professores e alunos do curso da instituição através de um aplicativo para Android, uma vez que smartphones estão sendo cada vez mais utilizados e aceitos pelos usuários finais que o utilizam cada vez mais como meio principal de comunicação. A aplicação visa simplificar os processos rotineiros da instituição, além de propor serviços que não estão disponíveis atualmente na mesma. Além de estreitar a comunicação de atividades importantes – eventos – para os alunos, disponibilizar informações sobre disciplinas, professores, etc.

Dispositivos móveis são cada vez mais utilizados e aceitos pelo público em geral. O sistema operacional Android executa em smartphones de diversas fabricantes (ex.: Motorola, Samsung, LG) e categorias, tendo como atrativos: desempenho, facilidade no uso, multitarefa, etc. A aplicação visa inicialmente atender a um público específico, do curso de Sistemas de Informação da UFMS, porém posteriormente atender a um público mais abrangente.

Este trabalho tem como foco oferecer uma aplicação móvel centralizada, que ao mesmo tempo irá oferecer funcionalidades úteis aos usuários. Trata-se de uma aplicação que irá contemplar o gerenciamento das disciplinas do curso de Sistemas de Informação (SI) da UFMS, pelo qual os professores poderão interagir diretamente com os alunos, lançando eventos específicos para suas disciplinas ou gerais, como: avaliações, trabalhos e seminários, ementa das disciplinas, carga horária das mesmas, etc. de forma rápida e simples.

1.2 **Objetivos**

O objetivo do projeto é oferecer aos alunos do curso de Sistemas de Informação da UFMS uma aplicação intuitiva para Android, além de projetar uma interface web para gerenciamento de conteúdo, facilitando a interação dos docentes com os estudantes.

1.2.1 **Objetivos Específicos**

O aplicativo tem como proposta dar suporte aos professores do curso, permitindo a disponibilização de conteúdo para os alunos, como: ementa das disciplinas, carga horária, notas, etc., permitindo maior interação entre eles e disseminação rápida de informação. Os alunos serão eventualmente informados – através de notificações do sistema – quando novas informações forem disponibilizadas pelos professores da instituição.

1. **Cadastro de eventos para disciplinas do curso:** Através do aplicativo Android será possível criar e gerenciar eventos relacionados às disciplinas que o usuário está

²Pesquisa realizada pela Secretaria de Comunicação Social da Presidência da República intitulada “Pesquisa Brasileira de Mídia 2015”.

cursando. Isto é, algumas atividades rotineiras, onde os acadêmicos da instituição poderão ser informados e notificados a respeito de acontecimentos nas disciplinas em curso. Alguns exemplos de eventos que poderão ser disponibilizados pelos professores: atividades, provas, trabalhos, apresentações, palestras, exercícios, avisos, comunicados, e outros;

2. **Integridade da aplicação:** Os eventos criados e informações a respeito das disciplinas, por exemplo, deverão ser gerenciadas pelos professores, a fim de manter a confiabilidade do sistema. Além de essas informações serem analisadas periodicamente, os estudantes não poderão excluir qualquer item cadastrado no aplicativo, deixando essa funcionalidade competente exclusivamente aos usuários responsáveis por gerenciar o conteúdo da aplicação;
3. **Anexo de arquivos:** Os professores do curso de Sistemas de Informação poderão anexar arquivos na aplicação. Desta forma, os professores poderão agilizar o processo de comunicação, levando rapidamente aos alunos interessados, por exemplo, atividades, cronogramas, etc;
4. **Desempenho acadêmico:** Os usuários poderão visualizar o seu desempenho acadêmico (em relação às notas) através de gráficos dinâmicos e explicativos;
5. **Avaliação de disciplinas cursadas:** Os alunos poderão emitir feedback para os professores avaliando sua experiência na disciplina lecionada.

Este projeto também contará com uma interface web para disponibilização do conteúdo pelos professores. Embora não seja o foco principal deste trabalho, iremos abordar seus pontos principais. Dentre suas funcionalidades, temos:

1. **Cadastro das disciplinas:** As disciplinas inseridas no aplicativo deverão ser atualizadas por professores da instituição, visando manter a confiabilidade da aplicação;
2. **Cadastro de eventos:** Os eventos também deverão ser criados pelos professores. Neste caso, cada disciplina terá o seu professor responsável, que ficará encarregado de atualizar as informações dos eventos disponíveis;
3. **Disponibilização de notas:** Os professores poderão lançar notas para suas disciplinas através da aplicação web. Essas notas serão disponibilizadas para os alunos através do aplicativo Android;
4. **Matrícula de alunos:** Os professores irão fazer a matrícula dos alunos cadastrados no sistema nas suas respectivas disciplinas através do sistema web.

1.3 Organização do Trabalho

No capítulo 2 será feito levantamento a respeito do referencial teórico, as tecnologias que serão abordadas neste trabalho e aplicações similares.

No capítulo 3 os conceitos teóricos envolvidos no desenvolvimento deste projeto serão abordados. Ferramentas como IDEs e bibliotecas utilizadas no decorrer deste trabalho são detalhadas.

O capítulo 4 trata do desenvolvimento desta aplicação proposta e detalhada nos capítulos anteriores. O capítulo de desenvolvimento detalhe o processo de desenvolvimento de software aplicado neste trabalho. Também serão apresentados modelos e artefatos de códigos criados, bem como a utilização de ferramentas de produtividade.

O capítulo 5 traz reflexões a respeito de contribuições que este trabalho tem no ambiente específico em que está inserido. As limitações experienciadas e futuras melhorias no projeto em geral.

Ao final deste trabalho serão disponibilizados como apêndices os artefatos gerados no processo de desenvolvimento deste projeto.

Capítulo 2

Referencial Teórico

Neste capítulo serão abordados os conceitos teóricos que envolvem este trabalho, comparar ferramentas semelhantes e mostrar o que foi observado de melhor em alguns trabalhos similares no desenvolvimento deste projeto.

2.1 Dispositivos Móveis

A crescente adesão dos usuários a smartphones trouxe uma infinidade de possibilidades no que se trata de inovações e criação de diferentes tipos de soluções que auxiliam na tomada de decisão, ajudam a resolver problemas do dia-a-dia e facilita tarefas. Smartphones e sistemas operacionais multitarefas impulsionaram uma nova era de experiência do usuário, onde diversas aplicações executam ao mesmo tempo e de forma rápida. A tecnologia móvel (do inglês, *mobile technology*) proporciona aos seus usuários uma forma mais flexível de executar tarefas rotineiras, como: acesso à internet, e-mails, etc.

Dispositivos móveis que executam o sistema operacional Android possuem alguns aspectos importantes a serem ressaltados, tais como: são multitarefas, ou seja, executam diversas operações simultaneamente, são populares atualmente, possuem baixo custo de utilização – uma vez que o sistema é aberto – e desenvolvimento, já que a construção de aplicações Android é gratuita.

2.2 Computação em Nuvem

Segundo Taurion (2009), pode-se dizer que a computação em nuvem é um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na internet. Ou seja, é possível acessar os mesmos arquivos de diferentes locais, através de diferentes dispositivos, isto porque os seus dados não se encontram em um local específico, e sim na rede, na “nuvem”.

Através da computação em nuvem e do conceito de armazenamento em nuvem, é possível criar uma base de dados, por exemplo, que pode ser acessada de qualquer dispositivo que

tenha acesso à rede de internet. As alterações feitas nesse banco de dados são repassadas para os outros usuários que têm acesso ao conteúdo.

A computação em nuvem oferece uma forma simples de acessar servidores, armazenamento, bancos de dados e um conjunto amplo de serviços de aplicativo pela Internet Amazon Web Services, Inc. (2016). O armazenamento em nuvem auxilia na redução de custos, pois evita armazenar arquivos diretamente no dispositivo, além de centralizar a disponibilidade de informações para diversos usuários, evitando redundância e atingindo um número maior de utilizadores.

2.3 Gerenciamento de Alunos

Sistemas de gerenciamento de alunos se tornaram frequentes e cada vez mais necessários, visto que a utilização da Tecnologia da Informação (TI) facilita tarefas rotineiras e permite o rápido compartilhamento de informações. Além disso, outros benefícios são notados e requeridos destes sistemas, tais como: lançamento de atividades, descrição de disciplinas e professores que compõe um curso de uma universidade, por exemplo.

2.4 Aplicativos Relacionados

Com a crescente popularização da tecnologia móvel, temos a possibilidade de criar soluções que aproximem os professores e alunos. Nesta seção iremos abordar trabalhos relacionados a este e destacar suas funcionalidades e diferenças.

2.4.1 Aplicativo Android para o Ambiente Univates Virtual

Este aplicativo foi desenvolvido por Marcel Dall'Oglio pelo Centro Universitário Univates em 2013¹. O Objetivo deste projeto segundo Dall'Oglio (2013, p.15), “é desenvolver um software (aplicativo) nativo para sistemas móveis Android, que permitirá aos usuários do ambiente de aprendizagem Univates virtual acessar o conteúdo e as atividades a partir dispositivos móveis”.

Dentre as principais funcionalidades do aplicativo Univates estão:

- Listar tarefas das disciplinas que o aluno está cursando;
- Listar e-mails recebidos no endereço eletrônico da instituição;
- Mostrar recursos adicionados nas disciplinas;
- Receber notificações para cada item listado acima.

¹Trabalho de conclusão de curso desenvolvido por Malcel Dall'Oglio na Universidade Univates em Lajeado, Rio Grande do Sul.

Este projeto possui algumas características divergentes das apresentadas acima. O aplicativo da Univates utiliza o servidor central da instituição, sendo disponibilizado através de um *web service*. A aplicação **UFMS Mobile** apresenta uma estrutura facilmente expansível para se adequar em qualquer instituição. Além disso, este projeto possui algumas funcionalidades ausentes no aplicativo Univates como, por exemplo, lista de alunos cursando determinada disciplina, download de materiais e acesso às disciplinas antigas cursadas.

2.4.2 UFPE Mobile

Segundo o UFPE (2015), “O UFPE Mobile é um aplicativo que reúne um conjunto de informações relevantes sobre a Universidade Federal de Pernambuco”². Esta aplicação tem como principais características: ver eventos, telefones, e-mails, mapa da universidade e informações do restaurante universitário.

O aplicativo UFPE Mobile apresenta algumas características informativas a respeito da IES (Instituição de Ensino Superior) em questão, como mapa da universidade e formas de contatos dos seus servidores. Já neste trabalho, a interação entre aluno e professor é mais evidente, já que os professores poderão comunicar os alunos de diferentes eventos. Além disso, funcionalidades de acesso à biblioteca e informações de restaurante universitário não estão presentes no aplicativo **UFMS Mobile**.

2.4.3 PUC Minas Mobile

Segundo o PUC Minas (2015), “com o aplicativo oficial da PUC Minas, você pode acessar o SGA³ e todas as informações institucionais da universidade, diretamente do seu celular ou tablet”⁴. O PUC Minas Mobile é semelhante ao sistema citado anteriormente, por trazer diversas informações sobre a instituição. Por outro lado, diferentemente do sistema apresentado anteriormente e deste trabalho, o aplicativo da PUC é integrado com o servidor da universidade.

Embora seja integrado com o SGA da PUC Minas, este aplicativo disponibiliza apenas a consulta de notas e faltas pelos alunos. A aplicação UFMS Mobile apresentada neste trabalho disponibiliza informações detalhadas sobre eventos disponíveis, disciplinas que estão sendo cursadas as já concluídas, além de gerar gráficos interativos para acompanhamento do desempenho acadêmico, funcionalidade que não foi identificada em nenhum destes trabalhos citados.

No capítulo seguinte será abordada a metodologia utilizada neste projeto, as tecnologias e linguagens de desenvolvimento, bem como os *frameworks* e bibliotecas nativas e de terceiros que auxiliaram a implementação deste trabalho.

²Disponível em: <https://goo.gl/JuPh1K>.

³Sistema de Gestão Acadêmica é o sistema web utilizado pela PUC Minas para acesso de alunos e professores.

⁴Disponível em: <https://goo.gl/3vhZqx>.

Capítulo 3

Metodologia

O desenvolvimento de uma aplicação Android envolve a utilização de diversas ferramentas, bibliotecas e APIs na fase de implementação. Felizmente, grande parte destes recursos são disponibilizados pelo Google ou por terceiros. Neste capítulo iremos abordar as metodologias e ferramentas necessárias no processo de desenvolvimento.

3.1 Processo de desenvolvimento unificado

O processo de desenvolvimento de software unificado (Unified Process) é uma metodologia de desenvolvimento de sistemas que visa diminuir a complexidade e riscos de um projeto, subdividindo-os em módulos. Este modelo é iterativo, ou seja, a cada etapa um sub-sistema é gerado. O processo unificado foi introduzido para ser uma alternativa ao modelo Cascata¹, já que este atrai grande risco para projetos.

O processo unificado permite feedback constante com o usuário e a diminuição do impacto dos riscos em um projeto. Cada iteração pode ser apresentada ao usuário final, garantindo que a mudança de requisitos ao decorrer do projeto seja gerenciada sem prejudicar o andamento do mesmo.

Neste trabalho será utilizada a metodologia do processo unificado, garantindo que nosso sistema seja subdividido em mini-projetos a cada etapa, diminuindo a complexidade do mesmo, aprimorando a detecção e correção de erros e minimizando o impacto de mudanças.

3.2 Ferramentas utilizadas

No desenvolvimento do projeto, diversos softwares foram utilizados. Na modelagem do aplicativo, utilizamos o Astah Professional. A IDE (*Integrated Development Kit*) utilizada no desenvolvimento da aplicação Android será o Android Studio. Para desenvolvimento da

¹Método de desenvolvimento em que as seguintes fases devem ser executadas, nesta ordem: requisitos, análise, arquitetura, design, implementação e testes. Uma característica deste modelo é que para passar para a próxima fase, a anterior deve estar 100% concluída.

plataforma web, utilizaremos a ferramenta PhpStorm. Nesta seção iremos as ferramentas utilizadas no processo de desenvolvimento deste trabalho.

3.2.1 Astah Professional

Segundo seu próprio site (<http://astah.net/editions/professional>) “Astah Professional é uma ferramenta de design de software que suporta UML, DER, DFD, diagrama conceitual e muito mais”. Neste projeto foi utilizada a versão profissional da ferramenta, que é comercial. Para fins de desenvolvimento, uma versão para estudante foi adquirida através do endereço (<http://astah.net/student-license-request>). Para solicitar uma versão de estudante, basta preencher o formulário com um e-mail válido de uma instituição de ensino superior ou enviar um documento de comprovação de matrícula junto à universidade. Dentro os diagramas criados utilizando este software estão: Diagrama de Casos de Uso e Diagrama de Classes.

3.2.2 Android Studio

Até meados de 2013 o Google disponibilizava duas opções de IDE para desenvolvedores: o *Android Development Tools* (ADT) e o Android Studio (Glauber, 2015, p.22).

O ADT pode ser integrado a IDE Eclipse de desenvolvimento para diversas funcionalidades para desenvolvimento de aplicações Android ao Eclipse, como: editor visual de layouts, emuladores Android e lint (um *framework* de verificação de erros).

Segundo Glauber (2015, p.22), o Android Studio, por sua vez, é uma personalização do IntelliJ IDEA, e foi lançado no Google I/O de 2013. Embora em versão beta quando esta aplicação foi proposta, atualmente o Android Studio é a ferramenta oficial e recomendada pelo Google para o desenvolvimento de aplicações Android.

O Android Studio (<http://developer.android.com/sdk/index.html>) traz diversas ferramentas integradas para auxiliar o desenvolvimento e aprimoramentos em relação ao ADT. Algumas ferramentas são: *Android Device Monitor* (para analisar comportamento de uma aplicação), criação de emuladores e integração com sistemas de versionamento – Git, por exemplo.

Atualmente o Eclipse e o ADT estão obsoletos, ou seja, não recebem mais atualizações para o desenvolvimento Android. Sendo assim, é preciso migrar projetos antigos para o Android Studio, função que está disponível desde as primeiras versões da IDE.

3.2.3 JetBrains PhpStorm

JetBrains PhpStorm (<https://www.jetbrains.com/phpstorm/>) é uma IDE robusta para desenvolvimento *back-end* e *front-end*² para a web, com suporte a diversas linguagens e

²*Back-End* define a camada de acesso aos dados no servidor, enquanto *front-end* representa a camada de apresentação – para o usuário final.

tecnologias. Segundo JetBrains s.r.o. (2016) “O editor entra no código e entende a fundo sua estrutura, que suporta todos os recursos da linguagem PHP, de projetos modernos até versões mais antigas”. Nesta IDE, iremos desenvolver a estrutura do *web service* em PHP e sincronizar com o banco de dados remoto e o sistema de arquivos do servidor.

Este é um software comercial para uso individual e, neste projeto, foi obtida uma licença para estudante, através de (<https://www.jetbrains.com/shop/eform/students>). É necessário preencher formulário com um e-mail acadêmico válido e sua licença dá acesso a todas as ferramentas da JetBrains pelo período de um ano.

O PhpStorm permite transferir arquivos com o nosso servidor de hospedagem através do protocolo FTP (*File Transfer Protocol*) de forma nativa, além de sincronizar os arquivos e identificar as alterações nos mesmos. Outra funcionalidade interessante é a sincronização com o banco de dados hospedado no nosso servidor remoto. Isso nos permite evitar erros de digitação ao criar os scripts, uma vez que a função de auto completar nos ajuda. Além disso, é possível analisar o banco de dados através de diagramas, gerados a partir das tabelas.

3.3 Bibliotecas e APIs

Existem diversas bibliotecas de terceiros e APIs nativas que visam facilitar o desenvolvimento de aplicações Android. Algumas introduzem recursos que não existiam ainda na plataforma, outras facilitam tarefas manuais de criação de componentes customizados e outras são utilizadas para consumir recursos disponibilizados pelo Google. Estas bibliotecas e APIs têm o intuito de tornar as aplicações Android mais robustas, customizáveis e homogêneas. Neste tópico iremos abordar as principais bibliotecas utilizadas neste projeto.

3.3.1 Google Play Services

O Google Play Services é um serviço que disponibiliza APIs do Google interessantes para serem incorporadas em uma aplicação Android, como Google Plus API, Youtube API, Ad-MobAPI e Google Cloud Messaging. Estas APIs permitem integrar serviços do Google em sua aplicação, tirando proveito de serviços prontos para uso. Nesta aplicação, vamos utilizar o Google Cloud Messaging. Glauber (2015, p.462) descreve “O Google Cloud Messaging, ou simplesmente GCM, é um serviço que permite que aplicações servidoras enviem mensagens para aplicações Android”. O GCM posteriormente será abordado com mais detalhes na seção 3.19.

As demais bibliotecas e ferramentas utilizadas neste trabalho podem ser encontradas no apêndice D deste documento.

3.4 Linguagens Utilizadas

Este projeto apresenta o desenvolvimento de uma aplicação móvel para o sistema operacional Android, além da implementação de uma ferramenta web para interagir com o aplicativo. A implementação deste trabalho utiliza linguagens de programação tanto no ambiente *mobile* quanto no ambiente web, que serão abordadas nas seções seguintes.

3.4.1 Java

Java é uma linguagem de programação orientada a objetos criada pela Sun Microsystems em 1995 e atualmente mantida pela Oracle inc. Milhares de aplicações de diferentes gêneros e alcances utilizam Java, tais como: *e-banks*, softwares para desktop, aplicativos para Android, etc. A linguagem Java é muito difundida, sendo a linguagem mais utilizadas por programadores no mundo inteiro³, como mostrado na figura 3.1.

Mar 2016	Mar 2015	Change	Programming Language	Ratings	Change
1	2	▲	Java	20.528%	+4.95%
2	1	▼	C	14.600%	-2.04%
3	4	▲	C++	6.721%	+0.09%
4	5	▲	C#	4.271%	-0.65%
5	8	▲	Python	4.257%	+1.64%
6	6		PHP	2.768%	-1.23%
7	9	▲	Visual Basic .NET	2.561%	+0.24%
8	7	▼	JavaScript	2.333%	-1.30%
9	12	▲	Perl	2.251%	+0.92%
10	18	▲	Ruby	2.238%	+1.21%
11	13	▲	Delphi/Object Pascal	2.005%	+0.85%
12	28	▲	Assembly language	1.847%	+1.23%
13	10	▼	Visual Basic	1.674%	-0.28%
14	23	▲	Swift	1.587%	+0.77%
15	3	▼	Objective-C	1.461%	-5.23%
16	20	▲	R	1.285%	+0.33%
17	36	▲	Groovy	1.193%	+0.78%
18	19	▲	MATLAB	1.193%	+0.19%
19	17	▼	PL/SQL	1.193%	+0.16%
20	31	▲	D	1.139%	+0.64%

Figura 3.1: Linguagens de programação mais utilizadas.

Fonte: TIOBE software BV (2016)

A plataforma Android utiliza a linguagem de programação Java como tecnologia oficial de desenvolvimento de aplicações. Através da programação é possível interagir com os

³Originalmente na proposta deste trabalho, a linguagem Java era a segunda mais utilizada no mundo, atrás da linguagem C.

dispositivos móveis de diversas formas, desde aplicativos que interagem diretamente com a resposta do usuário até manipulação de sensores dos aparelhos.

Orientação a objetos é um paradigma que representa toda a filosofia para construção de sistemas. Na orientação a objetos lidamos com objetos e estruturas que já estamos acostumados a lidar no nosso dia-dia e sobre as quais possuímos maior compreensão Dall'Oglio (2009). Decidimos utilizar uma linguagem orientada a objetos pela tendência de utilização dessa tecnologia e por algumas vantagens significativas desse tipo de abordagem, como: herança, polimorfismo, redução de linhas de código e reutilização de código.

3.4.2 PHP: Hypertext Preprocessor (PHP)

PHP (*PHP: Hypertext Preprocessor*) é uma linguagem de programação servidora utilizada no desenvolvimento de sistemas web dinâmicos, permitindo maior interação com o usuário. Uma característica desta linguagem é que os códigos escritos em PHP são executados no lado servidor da aplicação, ou seja, o usuário recebe somente o código HTML, uma vez que o processamento é realizado no lado servidor, conforme esquema da figura 3.2. O PHP se consolida ano após ano como uma das linguagens de programação orientada a objetos que mais crescem mundo Dall'Oglio (2009).



Figura 3.2: Funcionamento da linguagem PHP.

Fonte: Rahman (2011)

A utilização da linguagem PHP para desenvolvimento de soluções web está se popularizando, uma vez que possui detalhada documentação, além de ser abordada por diversas comunidades de desenvolvedores. Aliado a isso, sua flexibilidade e integração com outras tecnologias web torna o seu uso muito vantajoso.

A linguagem PHP será utilizada para a implementação do *web service* – que fará a comunicação entre plataformas –, do servidor Google Cloud Messaging, que irá enviar mensagens para os dispositivos Android a partir do servidor, e desenvolvimento da aplicação web, que será responsável pelo gerenciamento do conteúdo do aplicativo móvel.

3.4.3 Extensible Markup Language (XML)

XML, do inglês *eXtensible Markup Language*, é uma linguagem de marcação recomendada pela W3C⁴ para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais Pereira (2009). É considerada extensível porque possibilita que elementos de marcação sejam definidos.

A linguagem de marcação XML possui o mesmo conceito da linguagem HTML. Porém, enquanto a HTML é utilizada para formatar e organizar a estrutura de websites, a XML é utilizada para separar e organizar sequências de dados.

3.4.4 JavaScript Object Notation (JSON)

JSON (*JavaScript Object Notation*) é um formato de texto para a serialização de dados estruturados Simões (2007). Este formato é utilizado para transmitir e receber dados do servidor através do protocolo HTTP, é leve e utiliza o padrão chave-valor na sua construção.

As linguagens de programação utilizadas tanto no desenvolvimento da aplicação Android (Java), quanto no desenvolvimento da aplicação web (PHP) suportam a troca de informações via JSON, viabilizando sua utilização neste trabalho.

3.5 Model, View e Controller (MVC)

O MVC (*Model, View e Controller*) é uma arquitetura ou padrão de engenharia de software que lhe permite dividir as funcionalidades de seu sistema em camadas. Essa divisão é realizada para facilitar resolução de um problema maior Bastos (2014).

Segundo Sampaio (2007, p. 78) o “MVC é uma estratégia de separação de camadas de software que visa desacoplar a interface de seu tratamento e de seu estado. Suas camadas são: Modelo, Controle e Visão”.

O padrão MVC há muito tempo tem sido usado para desenvolvimento de aplicações. A ideia de camadas permite ao desenvolvedor de determinada aplicação maior controle do sistema em desenvolvimento, uma vez que esse padrão possibilita encapsular as diferentes atividades da aplicação.

O padrão MVC facilita eventuais manutenções à aplicação, além de desacoplar as partes integrantes de um sistema, auxiliando na identificação e resolução de problemas.

⁴W3C é a principal organização de padronização web.

3.6 Open Handset Alliance

Open Handset Alliance é um consórcio entre empresas de tecnologia que ajudam a desenvolver e manter a plataforma Android, criando um padrão aberto para telefonia móvel. Segundo Open Handset Alliance (2007), OHA “é um grupo de 84 empresas de tecnologia que se uniram para acelerar a inovação dos dispositivos móveis e oferecer aos consumidores uma rica e melhor experiência”. Empresas de telecomunicação, fabricantes de smartphones, semicondutores e software participam desta aliança.

Um dos fatores que auxiliaram a popularização do Android é o fato da plataforma ser código aberto. Além disso, empresas fabricantes de smartphones (como Samsung, Motorola e LG) participam da OHA e utilizam o Android em seus aparelhos. Geralmente estas fabricantes desenvolvem uma versão modificada do sistema com diversos adicionais como meio de obter diferencial competitivo.

3.7 Android

O Android é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middlewares*, aplicativos e interface do usuário Pereira, Lúcio Camilo O.; da Silva, Michel L. (2009). O sistema operacional tem como base o kernel do Linux, que é responsável pelo gerenciamento de processos, drivers, memória e energia Glauber (2015). Por ser baseado em Linux, o Android herda algumas vantagens do consagrado sistema, como: presença de multitarefa e gerenciamento eficiente de memória.

Além das características aproveitadas do Linux, desenvolver para Android traz alguns benefícios:

- A plataforma é open source;
- Ambiente de desenvolvimento é disponibilizado pelo Google de forma gratuita;
- Extensa documentação, guias, livros, etc;
- Presente em diversas tecnologias;
- Frequentemente atualizada.

Exatamente por ser código aberto, diversas fabricantes de aparelhos celulares utilizam este sistema – já que possui alto grau de customização – ajudando na sua popularização. A IDE oficial de desenvolvimento e suas bibliotecas também são gratuitas, além dos serviços disponibilizados pelo Google.

Presente em mais de um bilhão de dispositivos, principalmente em smartphones e tablets, o Android também pode ser encontrado em muitos outros dispositivos, como automóveis, TVs, relógios etc Glauber (2015). Esta disponibilidade em diversas plataformas possibilita

enorme flexibilidade no desenvolvimento de aplicações *mobile*, pois torna possível a integração entre diversas tecnologias através de um sistema operacional em comum.

3.7.1 Arquitetura

A arquitetura do sistema operacional Android é dividida em camadas, onde cada parte é responsável por gerenciar os seus respectivos processos Lecheta (2010). Como mostrado na figura 3.3, a arquitetura do Android é subdividida nas seguintes camadas:

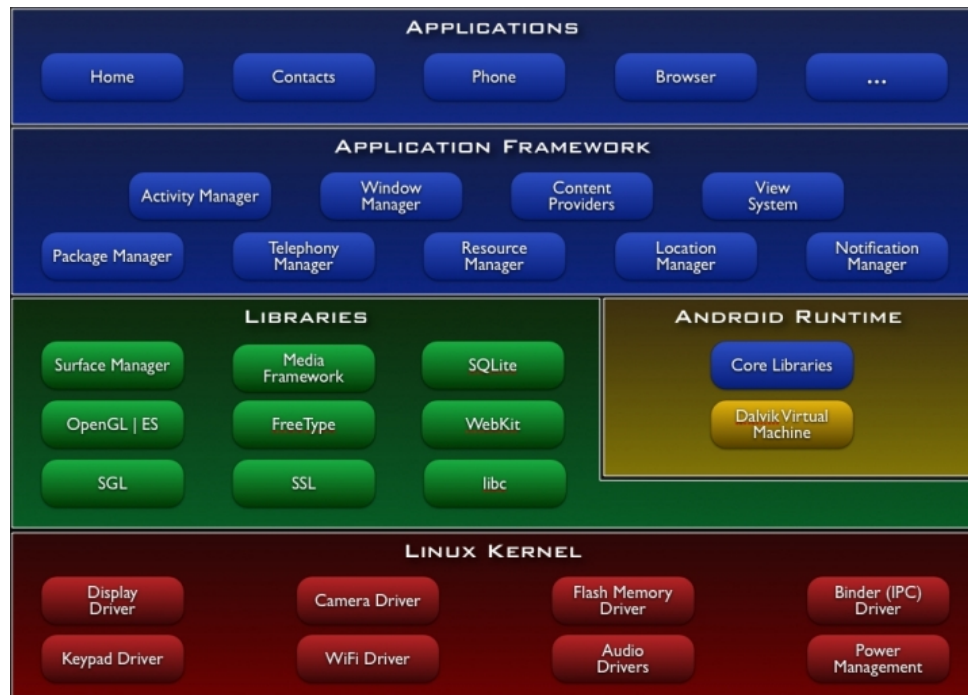


Figura 3.3: Arquitetura do Android.

Fonte: eLinux, Org. (2011)

- **Linux Kernel** – Responsável por gerenciar memória, controlar processos, drivers, gerenciar energia, etc.
- **Libraries & Android Runtime** – A camada de bibliotecas contém as bibliotecas em C que compõe o sistema, disponibilizando recursos de multimídia, gráficos 2D e 3D e banco de dados; a camada *runtime* é responsável pelo gerenciamento da DVM (*Dalvik Virtual Machine*), criada para executar aplicações Android no ambiente de desenvolvimento;
- **Applications Framework** – Esta camada representa serviços em alto nível disponíveis no sistema, como gerenciador de localização, notificações, telefone, etc;

- **Applications** – É nesta camada onde ficam as aplicações que executam sobre o sistema operacional. Note que as aplicações desenvolvidas por terceiros ficam no mesmo nível que as aplicações nativas do Android.

3.8 Dalvik Virtual Machine

A *Dalvik Virtual Machine* é uma máquina virtual baseada na JVM (*Java Virtual Machine*) responsável por executar as aplicações programadas em Android. Um dos pontos fortes da Dalvik é o gerenciamento de processos e *threads*⁵.

A Dalvik executa diversas versões disponibilizadas do sistema operacional Android, permitindo que sua aplicação seja executada em diferentes sistemas, garantido maior compatibilidade e desempenho.

A máquina virtual do Android é um *framework* que integra a IDE, permitindo maior produtividade e execução de testes dentro do ambiente de desenvolvimento. A Dalvik permite que sua aplicação seja realmente testada em um ambiente idêntico ao que teria em um dispositivo físico, trazendo desde funcionalidades básicas até algumas avançadas:

- **Instalação e execução de aplicações** – Os aplicativos desenvolvidos para Android são instalados na máquina virtual, possibilitando que esses sejam executados quantas vezes for necessário;
- **Envio e recebimento de mensagens de texto** – Aplicações que utilizam do envio e recebimento de mensagens de texto também estão disponíveis, através da perspectiva *Dalvik Debug Monitor Server* (DDMS) que permite simular o envio de mensagens de uma máquina virtual para outra, por exemplo;
- **Obter localização a partir do GPS (*Global Position System*)** – Ainda na perspectiva de DDMS, é possível utilizar os recursos de GPS do aparelho. Aplicações que utilizam esta funcionalidade podem ser facilmente testadas através deste servidor.

3.9 Notificações

O serviço de notificação no Android é um recurso muito usado e altamente recomendado quando uma aplicação executa em *background* (segundo plano) e deseja notificar o usuário a respeito de algum acontecimento. Ao invés de abrir uma tela ou uma caixa de diálogo – que por sinal pode não ser oportuno caso o usuário esteja utilizando o aparelho para alguma outra finalidade – a aplicação utiliza o serviço de notificação para que o usuário seja avisado de forma que não interfira na sua atividade naquele momento. Um exemplo amplamente conhecido deste recurso é o recebimento de mensagens de texto nos dispositivos móveis no Android, onde o usuário recebe um aviso em forma de texto atrelado a um aviso sonoro,

⁵*Threads* são sub-processos do sistema operacional.

permitindo que o utilizador do dispositivo possa posteriormente visualizar a mensagem, em um momento mais oportuno.

3.10 Activity e Intents

Nesta seção veremos como activities e intents são utilizadas em conjunto para abrir uma tela de sua aplicação, de aplicação de terceiros e até transferir dados entre activities da aplicação. Também iremos detalhar o ciclo de vida de uma activity, como são gerenciados e o impacto que têm na execução do aplicativo.

3.10.1 Activity

Quando se trata de interface gráfica no Android, uma tela é representada por um componente denominado activity. Uma activity (atividade) é uma classe que gerencia UI (*User Interface*) da aplicação e é composta por um grupo de views, que são componentes que podem ser exibidos na interface gráfica. Activities são estritamente ligadas a intents, pois para iniciar uma tela em Android, utilizamos um objeto intent, que representa a ação de abrir a tela, e efetuamos a operação como a chamada do método **startActivity(intent)**.

3.10.2 Ciclo de vida das activities

Activities em Android possuem um ciclo de vida gerenciado pelo próprio sistema operacional pelo qual todo processo de criação até a destruição de uma activity são tratados. *Você já se perguntou por que uma aplicação Android geralmente não possui um botão “fechar” como existe em softwares em outras plataformas, por exemplo?* Isso é possível pois as activities – que representam a interface gráfica do aplicativo – têm um comportamento baseado em ciclos de vida, responsáveis por controlar quando uma tela deve ser mostrada ou destruída (fechada).

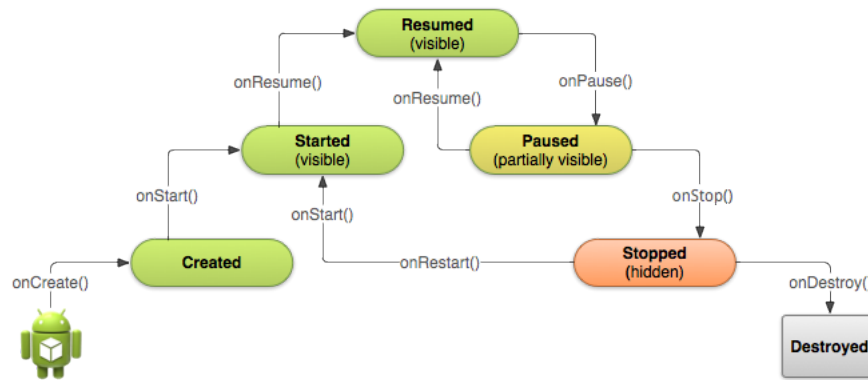


Figura 3.4: Ciclos de vida de um activity.

Fonte: Google, Inc. (2013)

Como mostrado na figura 3.4, os ciclos de vida de uma activity são:

1. onCreate()
2. onStart()
3. onRestart()
4. onResume()
5. onPause()
6. onStop()
7. onDestroy()

Estes ciclos de vida não são necessariamente executados nesta ordem, pois dependem da intervenção do usuário para serem chamados. Porém, alguns ciclos têm ordem bem definida. O método **onCreate()** é sempre o primeiro a ser chamado ao criar a activity, quando a tela é exibida pela primeira vez. Da mesma forma, o método **onDestroy()** é sempre o último a ser chamado a fechar a tela. Quando uma determinada tela é exibida para o usuário, esta já passou por três ciclos de vida: **onCreate()**, **onStart()** e **onResume()**. O método **onResume()** é chamado quando a activity está sendo mostrada para o usuário. Quando o usuário pressiona o botão *home* de seu aparelho, os métodos **onPause()** e **onStop()** são chamados em sequência. Ao voltar para aplicação, o ciclo de vida é retomado através dos métodos **onRestart()** e **onResume()**.

Existem diversas combinações possíveis para chamada destes métodos do ciclo de vida e o mais importante é que este gerenciamento é feito pelo próprio Android. Os ciclos de vida, além de gerenciarem o processo de execução de uma activity, também nos permite

executar operações nos seus respectivos estágios. Por exemplo, quando o método `onStop()` é chamado, a tela da aplicação não está sendo mostrada para o usuário – como mostrado na iteração anterior. Neste ponto, temos a chance de liberar recursos que nossa aplicação eventualmente esteja usando, já que não necessitamos neste momento.

3.10.3 Intents

Intent ou “intenção”, segundo Glauber (2015, p. 60) “representa uma ação que o usuário deseja realizar”. É um objeto que possui diversas características e muito usado por aplicações Android. Dentre suas funções, principalmente um objeto intent pode realizar as seguintes operações: abrir uma activity, se comunicar com aplicações externas e transmitir dados entre activities.

Abrir uma activity da sua aplicação

Ao abrir uma activity da sua própria aplicação, um objeto intent deve ser criado. Este objeto irá representar o seu interesse (intenção) em abrir determinada activity. O objeto intent pode ser de dois tipos: implícito e explícito. Implícito é quando o desenvolvedor não indica qual tela deverá ser aberta em uma determinada ação, e sim, é definido qual a intenção do usuário (por exemplo, abrir uma tela). Uma intent implícita então é disparada no sistema operacional, e somente a activity que puder executar tal intenção será aberta. Já a intent explícita é usada quando o nome da classe da activity que será aberta é explicitamente definida na criação do objeto intent.

Em resumo, a intent implícita informa um interesse em executar uma ação mas não sabe a classe que irá executar ou mesmo se vai executar (no caso de nenhuma intent for encontrada, uma exceção é lançada). A intent explícita indica via código diretamente a classe que será encarregada de executar a ação.

Abrir um aplicativo do sistema Android

Além de abrir uma activity da sua própria aplicação, é possível interagir com aplicações nativas do Android, como calendário, alarme, mapas, etc. Essa interação é feita através de intents. Neste caso, uma intenção (como, por exemplo, abrir um arquivo PDF) é disparada por sua aplicação e propagada no sistema operacional Android. As aplicações que são capazes de “ouvir” este tipo de evento de “abrir um arquivo PDF” serão executadas.

O Android possui diversas aplicações nativas que podem ser utilizadas para executar determinada ação na sua aplicação. É possível até comunicar-se com aplicações de terceiros. Este recurso propicia uma flexibilidade muito grande ao desenvolver aplicações Android, já que existem aplicações nativas (ou de terceiros) que podem executar uma tarefa para sua aplicação e vice-versa.

Transmitir dados de uma tela para outra

O terceiro – e não menos importante – uso de intents no Android é na transmissão de dados de uma activity para outra. Ao abrir uma segunda activity, é possível enviar da primeira activity uma porção de informação que pode ser de tipos primitivos e objetos da linguagem Java. Embora seja possível enviar objetos inteiros para outra activity, este objeto deve implementar a interface **Parcelable** para que isso funcione.

Para atingir este objetivo, basta implementar esta interface em qualquer objeto da aplicação. Feito isso, é possível enviar o objeto da classe inicial e recuperar na classe resultante. Embora exista outra opção de interface que implementa este mesmo comportamento – a interface **Serializable** –, objetos que implementem Parcelable desfrutam do mecanismo mais rápido de transmissão entre activities disponível no Android.

Nesta aplicação iremos implementar a interface Parcelable em nossos objetos visando melhor desempenho possível ao transmitir informações. Exemplos de informações que podem ser transmitidas usando este mecanismo é passando um objeto inteiro de um item clicado na lista para uma activity de detalhes.

3.11 Fragments

Até a versão 3.0 do Android (Honeycomb) a única forma de exibir uma tela de uma aplicação era através do uso de activities. Com o surgimento de dispositivos com telas maiores, principalmente dos tablets, uma nova experiência de uso de aplicações Android foi introduzida.

Os fragments foram introduzidos a partir da versão 3.0 do Android e vieram para suprir uma necessidade dos usuários: melhor gerenciamento do espaço das telas. Como as telas dos dispositivos só aumentavam, as activities padrão se apresentaram ineficientes quando era necessário aproveitar todo o espaço disponível na tela do usuário.

Os fragments (ou fragmentos) são componentes utilizados para dividir as activities em “mini-activities” proporcionando diferentes comportamentos em determinadas partes de uma mesma tela. Segundo Google, Inc. (2014), um fragment representa o comportamento ou uma porção da interface de usuário em uma activity. Uma activity representa uma tela e pode ter vários fragments, e cada fragment possui uma view associada. Por esse motivo a utilização permite exibir diversas views em uma mesma tela de uma aplicação.

Outra característica proporcionada pela utilização dos fragments é a execução da mesma aplicação em dispositivos heterogêneos. Uma mesma aplicação executando em um smartphone e em um tablet. Quando executada em um smartphone, os fragments A e B são mostrados em telas diferentes. Já no tablet, os fragments são mostrados na mesma tela, gerenciando melhor o conteúdo e ocupando os espaços de maneira eficiente, aproveitando o maior tamanho da tela.

3.12 Armazenamento

Existem várias formas de armazenar dados no Android. Desde o armazenamento em arquivo a banco de dados, gravar dados de forma correta é um ponto crucial no sucesso de qualquer projeto. Basicamente é possível ler e escrever dados em arquivos de forma direta e indireta (através das *shared preferences*, que serão abordadas neste tópico), e banco de dados. Embora o Android não suporte acesso a banco de dados remoto de forma nativa, é possível fazê-lo através da implementação de *web services* para comunicação eficiente com a aplicação.

3.12.1 Banco de dados SQLite 3

O SQLite é uma biblioteca disponível em Android – atualmente na versão 3 – que implementa um banco de dados local, compacto e que não necessita de servidor e nem de configurações. Segundo Glauber (2015), com o SQLite podemos usar os diversos recursos existentes em SGBDs, tais como criar tabelas com chaves primárias e estrangeiras, *views*, índices, suporte à transações e até *triggers*.

Esta biblioteca é desenvolvida na linguagem C, porém o desenvolvedor não necessita conhecer esta tecnologia para utilizá-la, já que os métodos desta biblioteca estão disponíveis para acesso em linguagem Java.

O SQLite, como já citado anteriormente, possui algumas características, como:

- **Suporte a transações** – Recurso muito importante na utilização de bancos de dados, que nos permite efetuar um conjunto de operações somente se todas forem bem-sucedidas (*commit*), caso contrário, as operações são desfeitas (*rollback*).
- **Local** – O SQLite não necessita de instalação, pois está disponível no sistema operacional, bastando apenas usá-lo.

Entre outros recursos disponíveis no SQLite 3 estão: criação de chaves (primária e estrangeira), índices e gatilhos (*triggers*).

Usaremos o SQLite para sincronizar o conteúdo da aplicação com o servidor remoto. No banco de dados local será armazenado conteúdo vindo servidor, prevenindo que a aplicação seja inutilizada caso o usuário não esteja conectado à internet.

Para utilizar o SQLite devemos acessar o pacote **android.database.sqlite**. Ao criarmos um banco de dados para a aplicação, este será salvo na memória interna do dispositivo no caminho `/data/data/<pacote do aplicativo>/databases/<nome do banco>`. Repare que, mesmo o banco de dados sendo salvo na memória interna do dispositivo, por motivos de segurança, ele não pode ser acessado pelo usuário através do celular, sendo acessado por ferramentas de desenvolvimento da IDE, como *adb shell* e *Android Device Manager*.

3.12.2 MySQL

Através de um servidor MySQL remoto, será possível conectar diversos dispositivos rodando o aplicativo à mesma fonte de dados, sincronizando com banco de dados local. O servidor de banco de dados será utilizado na versão 5.6.21, sendo gerenciado através do **phpMyAdmin**. O servidor remoto de banco de dados será utilizado através de um *web service* e também será integrado o com o serviço **Google Cloud Messaging**, que serão abordados mais adiante.

3.12.3 Shared Preferences

Preferências Compartilhadas (ou Shared Preferences) no Android são uma forma de armazenamento simples e eficiente para gravar dados primitivos essenciais na aplicação evitando, por exemplo, que o usuário tenha que entrar com dados diversas vezes. No Android, podemos fazer isso com a classe `SharedPreferences`, onde salvamos informações no formato “chave/valor” para recuperá-los posteriormente Glauber (2015).

As preferências no Android são geralmente utilizadas para gravar as configurações do usuário na sua aplicação – como, por exemplo, se você deseja salvar os downloads na memória interna do dispositivo ou no `sdcard` – e lembrar do usuário logado na aplicação, evitando que o mesmo seja perguntado sobre login e senha toda vez que o aplicativo é executado.

Assim com o banco de dados SQLite (seção 3.12.1), o arquivo de preferência será gravado em `/data/data/<pacote da sua aplicação>/shared_prefs/<nome do seu arquivo>`.

3.12.4 Preferences Framework

Segundo Glauber (2015, p. 295) “Muitas vezes o usuário acha um aplicativo melhor que o outro pelo simples fato de poder configurá-lo da maneira que ele desejar”. O Preferences Framework é um *framework* em Android que permite criar telas de configurações padronizadas para suas aplicações. Estas telas são desenvolvidas da mesma forma como o *layout* das telas que compõe a sua aplicação em si são desenvolvidas, abstraindo a complexidade da implementação. O preferences framework permite que o usuário configure o seu aplicativo da forma que julgar mais apropriada. Alguns componentes podem ser adicionados nas telas de configuração, os mais importantes são:

- **CheckBoxPreference** – Exibi um elemento do tipo checkbox que permite que o usuário “marque” ou “desmarque” uma opção, habilitando-a ou não;
- **SwitchPreference** – Este componente é semelhante ao `CheckBoxPreference`, porém exibe um botão de liga/desliga o componente;
- **ListPreference** – Exibe uma lista de opções para o usuário escolher uma (por exemplo, escolher qual som tocar ao receber uma notificação);

- **Preference** – É uma preferência simples que adiciona um componente com texto na tela, que pode ser clicado ou não. Geralmente utilizado para informar algo para o usuário ou exibir opções de sair da aplicação.

O preferences framework salva as configurações definidas pelo usuário em um arquivo de preferências – shared preferences – gerenciado pelo próprio Android. Tudo que temos que fazer é ler essas opções e determinar como nossa aplicação irá funcionar a partir destas configurações.

Além das vantagens já citadas anteriormente, este recurso da plataforma ainda proporciona outra facilidade para o usuário: as telas de configurações do próprio sistema operacional Android e dos aplicativos desenvolvidos pelo Google utilizam este *framework*. Desta forma, os usuários já estão acostumados com o seu funcionamento.

3.13 Compatibilidade

Ao desenvolver em Android, algumas especificações precisam ser levadas em conta, como: a versão da plataforma em que o aplicativo será desenvolvido (*targetSdkVersion*), a versão mínima do Android (*minSdkVersion*) que poderá executar a aplicação e a versão mais recente (*compileSdkVersion*). No caso da versão em que o aplicativo está sendo desenvolvido e na versão mais recente, geralmente se utiliza a última versão do Android, assim tendo acesso às atualizações e recursos mais recentes da plataforma. No caso da versão mínima é um pouco mais complicado. Por um lado, se uma versão muito antiga do Android for escolhida, o aplicativo poderá rodar em um número muito grande de dispositivos, porém os recursos mais novos da plataforma, como por exemplo: **Fragments**, **Toolbar**, **DrawerLayout**, não estarão disponíveis. Pensando nesta dificuldade dos desenvolvedores, o Google desenvolveu um padrão para que seja possível escolher a versão mínima mais adequada para o projeto de forma mais simples. A fórmula é: a versão mínima do seu aplicativo deve atingir pelo menos 90% dos dispositivos no mercado. A partir deste conceito, a versão mínima escolhida para este trabalho foi a API Level 17 ou Android 4.2 (Jelly Bean).

3.13.1 Support Library

Ainda falando de compatibilidade de versões do Android, um esclarecimento ainda precisa ser feito. *Como um aplicativo projetado para executar nos smartphones mais modernos do mercado pode se adaptar aos dispositivos mais antigos sendo que muitos recursos foram somente introduzido nas versões mais novas?* É exatamente aqui que surge uma poderosa biblioteca: a Support Library. Com recursos mais modernos e a reformulação do design das aplicações proposto pelo Google no Android 5.0 (Lollipop), vários elementos foram introduzidos à plataforma, protagonizando a maior mudança visual desde o Android 4.0, motivando a utilização de novos componentes em versões mais antigas. Foi pensando nisso que a biblioteca de suporte foi apresentada, trazendo os recursos mais modernos disponíveis também em

dispositivos antigos (a partir do Android 1.6 – API Level 4), dando suporte a todos os tipos de usuários e criando uma aplicação homogênea e que se adapta a diferentes configurações.

A utilização desta biblioteca é muito simples e se resume a importar duas dependências no projeto. A primeira é a **SupportAppCompat**, que adiciona suporte a ActionBar da aplicação. A segunda é a **SupportDesign**, que tem como objetivo implementar os recursos que estão sendo introduzidos pelas atualizações da plataforma, fazendo que os desenvolvedores foquem nas funcionalidades do projeto.

Tabela 3.1: Dependências da biblioteca de Support.

Biblioteca	Dependência
AppCompat	compile 'com.android.support:appcompat-v7:23.1.1'
Design	compile 'com.android.support:design:23.1.1'

Após atualizar o arquivo de dependências, é preciso alterar a super classe das activities de **extends Activity** para **extends AppCompatActivity**. É preciso também sempre verificar a versão do Android em tempo de execução para garantir que a chamada de um determinado método realmente existe para aquela versão, evitando que o aplicativo pare de responder.

3.14 Layouts

Os elementos que compõem a tela de uma aplicação Android são organizados por gerenciadores de layouts, que são componentes que agrupam outros componentes dentro de um espaço determinado, compondo uma tela do aplicativo. Todos os componentes visuais do Android herdam direta ou indiretamente da classe **android.view.View**, e todos os gerenciadores de layout são subclasses de **android.view.ViewGroup**, que também é subclasse de View Glauber (2015). Embora existam diversos tipos de gerenciadores de layout, os mais utilizados são: **LinearLayout**, **RelativeLayout** e **FrameLayout**. Recém introduzido na plataforma, o **CoordinatorLayout** será abordado na seção 3.15 sobre Material Design. Os elementos visuais que podem ser adicionados a uma tela – como, por exemplo, botões, caixas de texto, etc – são chamados de view, pois são elementos visuais. Por esse motivo, os gerenciadores de layout são chamados de viewGroup (ou “agrupador de views”), já que agrupam outras views. Cada gerenciador de layout organiza os elementos na tela de uma forma específica, como veremos a seguir.

3.14.1 LinearLayout

Segundo Glauber (2015), o **LinearLayout** organiza os componentes um ao lado do outro, ou um abaixo do outro, dependendo da propriedade **android:orientation**. Esta propriedade

precisa ser definida na declaração deste componente no arquivo xml de layout. Esta propriedade deve ser adicionada se dois ou mais itens forem adicionados a tela, pois, se somente um elemento for adicionado à tela (por exemplo, um botão), não tem sentido definir a orientação pela qual este item deve ser exibido, já que só existe um.

A propriedade *orientation* do `LinearLayout` pode ser: *horizontal* e *vertical*. Se for definido horizontal como valor da orientação, as views que serão adicionadas a esse layout vão se organizar alinhadas horizontalmente umas as outras, isto é, lado a lado. Por outro lado, se for definida como vertical, os elementos vão ser posicionados um embaixo do outro.

3.14.2 RelativeLayout

Este é o layout mais flexível de todos, pois nos permite adicionar os componentes em relação a outros componentes, ou às bordas. É possível indicar, por exemplo, que um componente ficará acima, abaixo, ou ao lado de outro Glauber (2015). O próprio nome (Layout Relativo, do português) já indica como funciona este gerenciador de layout. Neste gerenciador de layout, os elementos são posicionados relativamente a outros componentes. Por exemplo, é possível definir o botão de submeter o formulário ficará embaixo do campo de texto que o usuário digitou sua mensagem. Informando isso ao Android, o sistema operacional se encarregará de posicionar o botão com relação ao formulário, inclusive se a tela for rotacionada.

3.14.3 FrameLayout

Conforme Glauber (2015), o `FrameLayout` é o gerenciador de layout mais simples, e normalmente usado quando queremos colocar apenas um único elemento dentro dele. Embora este gerenciador tenha esta propriedade, é possível adicionar mais elementos dentro dele, porém, estes elementos irão sobrepor uns aos outros. O componente que for adicionado por último ficará no topo dos outros. Embora em um primeiro momento seja complicado pensar em uma utilização para este tipo de layout, é muito comum que este tipo de comportamento seja utilizado no desenvolvimento de telas para aplicações.

3.15 Material Design

Com o lançamento do Android Lollipop (5.0), o Material Design foi apresentado e adotado pelo Google como padrão de design a ser seguido para aplicações Android. Com o objetivo de criar uma identidade visual, este padrão traz uma série de mudanças e introduz diversos componentes que serão utilizados em nosso aplicativo. Segundo Glauber (2015), o Material Design não é exclusivo do Android, e seus princípios podem ser empregados também em outras plataformas.

3.15.1 CardView

O componente CardView nada mais é do que um `FrameLayout` com bordas arredondadas, `background` e uma sombra Glauber (2015). Introduzido no Android Lollipop 5.0, o CardView pode ser usado como contêiner para outras views (por exemplo, texto e imagem), representando cada item de uma lista, proporcionando um visual moderno e amigável. Para utilizar este componente nas versões mais novas e com suporte às versões antigas, é necessário adicionar a dependência do gradle: `compile 'com.android.support:cardview-v7:23.+'`.

3.15.2 RecyclerView

Assim como o `ListView`⁶, a `RecyclerView` comporta um conjunto de views, formando uma lista de elementos que podem ser exibidos em uma `activity` ou `fragment` para o usuário. Entretanto, `RecyclerView` é uma versão mais avançada e eficiente do `ListView`, permitindo, como sugere o seu nome, “reciclar” os elementos com maior eficiência, proporcionando ganho considerável de desempenho ao carregar uma lista.

Embora seja um pouco diferente da `ListView`, o `RecyclerView` é utilizado de forma semelhante. Para ser acessado via `layout` e código, é preciso adicionar a dependência no projeto: `compile 'com.android.support:recyclerview-v7:23.+'`. Feito isso, é preciso criar um `adapter` que estende de `RecyclerView.Adapter`, que irá popular nossa lista de elementos, e definir um gerenciador de `layout`, que definirá a posição dos elementos da lista.

Finalmente, é preciso definir um `ViewHolder`⁷, que fará reutilização dos objetos já criados na lista para alocar novos objetos, evitando criar um novo objeto a cada chamada, melhorando a performance da aplicação.

3.15.3 TextInputLayout

`TextInputLayout` é um componente do Material Design que permite animar caixas de texto (`EditText`) no Android. O funcionamento deste componente depende da existência da propriedade `hint` (dica), que é um *placeholder* na caixa de texto informando ao usuário o que deve ser digitado. A mensagem de *hint* é exibida normalmente, porém quando a caixa de texto ganha foco, um efeito é aplicado, movendo esta mensagem para fora da caixa de texto, a posicionando como uma *label*.

Utilizar este componente é muito simples, basta envolver o `EditText` com o `TextInputLayout`, disponível na API de design, que a animação ocorrerá automaticamente.

⁶Componente que suporta conjunto de views no Android formando uma lista.

⁷`ViewHolder` é um padrão de desenvolvimento para criação de listas eficientes, evitando criação desnecessárias de objetos.

3.15.4 Snackbar

O Snackbar é um componente que permite enviar uma mensagem rápida para o usuário, sem que o usuário seja interferido na utilização da aplicação. Este componente é muito semelhante ao toast, porém alguns recursos foram adicionados, além de seu visual ter sido modificado. Com o Snackbar é possível executar ações, o que não é permitido usando toast.

Embora seja permitido realizar ações através da Snackbar, não é comum que operações que demandam muito processamento sejam executadas. Geralmente, essas ações se limitam a desfazer outras ações feitas pelo usuário ou do tipo “tente novamente”. Nesta aplicação, a Snackbar será utilizada para informar procedimentos bem-sucedidos e permitir que o usuário tente novamente quando a aplicação perder conexão com a internet.

3.15.5 AppBar

Este componente representa a barra de ferramentas posicionada na parte superior da tela de uma aplicação Android. A Application Bar ou Toolbar foi introduzida para substituir a ActionBar⁸, que se tornou obsoleta nas versões mais recentes por sua capacidade limitada de personalização. A AppBar pode ser posicionada em qualquer local da tela e inclui diversas opções de customização, como: logo da aplicação, nome, ações e ainda é possível definir animações que serão executadas durante movimento de scroll, como será abordado a seguir.

3.15.6 CoordinatorLayout

Este componente em conjunto com outros componentes permite realizar um efeito muito legal em aplicações Android. O CoordinatorLayout introduziu na plataforma, a habilidade de ocultar e reposicionar elementos na interface gráfica quando uma ação do usuário é tomada. Com isso, o usuário tem feedback imediato de que suas ações estão surtindo efeito no aplicativo, além de ganhar mais espaço de visualização na tela.

Como citado anteriormente, este comportamento é alcançado com a ajuda de outros componentes como: RecyclerView e NestedScrollView. A integração destes componentes permite realizar uma animação muito interessante na aplicação. Ao realizar um movimento de *scroll* para cima em uma lista (para carregar os elementos que estão mais abaixo), a AppBar (barra de ferramentas superior da aplicação) é omitida, uma vez que o usuário não precisa dela neste momento, já que a lista de elementos está em foco. No momento em que um movimento de *scroll* para baixo é realizado (para carregar os itens mais acima), a AppBar é carregada novamente, exibindo as opções para o usuário.

⁸Antiga barra de tarefas do Android que gradualmente é substituída pela Toolbar.

3.16 Navegação

A navegação de uma aplicação Android deve seguir algumas recomendações para que a experiência do usuário utilizando a sua aplicação seja a melhor possível. Nesta aplicação, alguns padrões desenvolvidos e indicados pelo Google foram utilizados, promovendo uma estrutura já conhecida pelo usuário, pois são também utilizadas por diversas outras aplicações.

Para tornar atraente a navegabilidade da aplicação, é necessário que padrões sejam estabelecidos e que a estrutura tenha coerência. Pensando nisso, um padrão de navegação muito comum nas aplicações atuais é o Navigation View, que veremos na sequência. Além disso, uma aplicação geralmente possui uma hierarquia de telas, que pode ser implementada utilizando a navegação Up. Por fim, a navegação em abas é muito comum para organizar conteúdo por categorias, além de tornar a aplicação mais atraente.

3.16.1 Navigation View

O Navigation View (ou Navigation Drawer) é um menu lateral que pode ser acionado ao clicar no ícone do canto superior esquerdo ou deslizando o dedo na tela da esquerda para a direita para abrir, e da direita para a esquerda para fechar. Este componente é muito utilizado e indicado pelo Google para criação de menus para oferecer opções para os usuários da sua aplicação. Algumas das vantagens de utilizar este tipo de navegação:

- Menu fica oculto quando o foco do usuário está no conteúdo da aplicação, desta forma maximizando o espaço a ser utilizado pela aplicação;
- Pode ser aberto facilmente através de gestos e toques na tela, o que é familiar para a maioria dos usuários;
- É possível implementar comportamento personalizado, como, por exemplo, fechar o menu automaticamente ao clique do usuário e abrir-lo quando a aplicação for executada pela primeira vez, mostrando ao usuário que existe um menu de navegação lateral;
- Muitas aplicações atualmente usam este tipo de navegação, facilitando na aprendizagem do usuário.

O componente Navigation View está disponível através da biblioteca de design disponibilizada pelo Google e geralmente é utilizado em conjunto com um arquivo xml de menu, onde as opções do menu são definidas. Além do menu de opções, é possível definir um cabeçalho, onde informações como: nome, e-mail, foto, etc., do usuário podem ser exibidas.

3.16.2 Up Navigation

Ao clicarmos em alguma opção do menu lateral, ou em algum evento ou disciplina da tela principal, outra tela será aberta. Clicando novamente, outra tela será aberta e assim suces-

sivamente. Ao abrir outras telas da aplicação, estamos tornando o conteúdo da mesma mais específico, ou seja, estamos aprofundando na hierarquia de telas do aplicativo.

Se aprofundarmos muito nesta hierarquia, seria desconfortável do ponto de vista de navegabilidade, o usuário ter que clicar varias vezes no botão *back* (voltar) do Android. Pensando nisso, a plataforma disponibilizou um recurso muito interessante: o Up Button ou Up Navigation. Este recurso é um botão localizado na ponto superior esquerdo e pode ser configurado para voltar um nível na hierarquia de navegação. Diferente do botão voltar, o up button pode pular algumas telas de uma só vez, já que direciona a aplicação para a tela anterior na árvore de navegação.

3.16.3 Layout com abas usando TabLayout

Definir o modelo de navegação de uma aplicação pode ser uma tarefa desafiadora. Quando se tem bastante conteúdo a ser exibido para o usuário, uma alternativa é exibi-los por categorias, de forma organizada. Pensando nisso, o Android disponibiliza um componente muito interessante: o uso de abas com TabLayout. Segundo Glauber (2015), o uso de abas no Android passou a ser feito através da toolbar a partir da versão Lollipop, por ser mais flexível que a antiga ActionBar.

3.17 Rede

A internet impulsionou de forma exponencial como a informação é obtida por uma aplicação. Através dela é possível ter acesso a inúmero recursos e conectar as pessoas de forma direta e indireta. Nesta aplicação, vamos utilizar os protocolos HTTP para criar um serviço web, através do padrão REST, que será implementado afim de permitir a comunicação dos usuários de nosso aplicativo.

3.17.1 Web Services

Web service é um serviço que não depende de uma linguagem de programação específica e pode ser criado e disponibilizado através de um servidor web e consumido por outras aplicações. Web services são utilizados como forma de integração e comunicação de sistemas, de modo que um sistema possa realizar uma chamada para um serviço de outro sistema a fim de obter informações Lecheta (2015). Através deste serviço, um cliente poderá enviar e receber dados de um servidor, fazendo com que a troca de informações ocorra de forma transparente. Segundo Lecheta (2015), existem várias formas de criar *web services*, mas podemos dizer que duas das mais conhecidas são *web services* em SOAP ou REST. Neste projeto, iremos utilizar o padrão REST.

3.17.2 Padrão REST

O REST (Representational State Transfer) tornou-se o padrão de web services mais adotado mundialmente Glauber (2015). Embora não seja restrito ao protocolo HTTP (*Hyper Text Transfer Protocol*), este é o protocolo mais utilizado nas implementações deste modelo. Através deste padrão, alguns métodos podem ser implementados, são eles: GET, POST, PUT e DELETE. Os métodos disponíveis não são uma regra, mas estão à disposição do desenvolvedor. Geralmente serviços que implementam o padrão REST são representados usando XML ou JSON, por serem linguagens independentes de plataforma, flexíveis e leves. Para este aplicativo, iremos utilizar o JSON, por ser mais comum neste tipo de projeto e mais leve que XML.

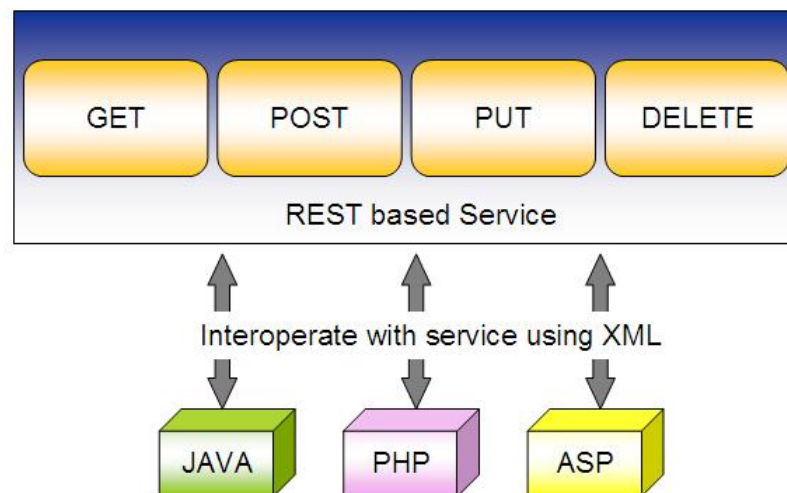


Figura 3.5: Esquema exemplificando o RESTful Web Service.

Fonte: Sur (2014).

Na figura 3.5 observa-se a arquitetura REST utilizada pelo web service RESTful, onde temos os seguintes métodos disponíveis:

- **GET:** É utilizado para obter recursos do servidor, ou seja, para ler determinados dados que estão disponíveis em uma aplicação disponível na internet;
- **POST:** É utilizado para adicionar um recurso ao servidor através de um dispositivo cliente, ou seja, inserir informações em uma aplicação que executa na internet;
- **PUT:** Este método atualiza um recurso disponível no servidor, alterando o seu valor/conteúdo;
- **DELETE:** O método de exclusão permite deletar recursos previamente inseridos no servidor.

3.17.3 RESTful Web Service com PHP

Um RESTful *web service* nada mais é do que um serviço web que utiliza o padrão REST. Pode ser implementado em qualquer linguagem de programação que suporte esta tecnologia, porém neste projeto iremos utilizar o PHP. Por ser uma linguagem que faz a integração com banco de dados de forma simples, o PHP é uma alternativa interessante para o desenvolvimento de *web services*. Por serem assuntos complexos e não serem o foco principal deste trabalho, não temos intuito de nos aprofundar em *web services* e PHP.

Nosso serviço ficará disponível no servidor para acesso da aplicação. Para acessá-lo, será declarado no aplicativo o endereço na web em que o servidor pode ser encontrado. A comunicação entre cliente e servidor precisa ser rápida, pois o Android impõe algumas restrições caso a aplicação esteja demorando muito para responder, resultando no fechamento da mesma. Pensando nisso, iremos utilizar JSON – para consumir e enviar dados ao servidor – em conjunto com a biblioteca volley, que falaremos a seguir.

3.17.4 Acesso à rede eficiente com Volley

Volley é uma biblioteca mantida pelo Google e introduzida no Google I/O do ano de 2013 que facilita o desenvolvimento de aplicações que acessam a rede. O volley vem com tudo que o desenvolvedor precisa – APIs para baixar JSON e imagens, cache em disco ou memória – é totalmente customizável e apresenta facilidade de depuração Glauber (2015). Vamos detalhar alguns benefícios da utilização desta biblioteca:

- “Tentar novamente” automático, caso a conexão com a rede não esteja disponível no momento;
- Suporta requisições do tipo string, imagens e objetos JSON;
- Mecanismo de cache, que melhora consideravelmente o desempenho em requisições web;
- Suporta priorização de requisições: é possível enfileirar diversas requisições e definir quais têm prioridade;
- Permite carregar imagens da web para nosso aplicativo.

Embora seja uma biblioteca externa para acesso à rede – em detrimento da classe nativa `URLConnection` –, volley apresenta melhoras significativas de desempenho em aplicações Android, graças a mecanismos de cache, prioridades e, principalmente, por utilizar o padrão Singleton⁹. na sua implementação, economizando recursos do sistema.

⁹Permitir a criação de uma única instância de uma classe e fornecer um modo para recuperá-la (K19 Treinamentos, 2015, p. 30).

3.17.5 Tarefas assíncronas

Ao executarmos uma aplicação Android, ela será executada por uma *thread*, que é um processo que contém uma linha de execução, que será interpretada pelo sistema e executada. Esta *thread* é chamada *main thread* ou *ui thread*. Glauber (2015) define que nela são tratadas todas as interações do usuário com os componentes de interface gráfica. Por este motivo, deve-se evitar executar processamento demorado ou acesso à rede que pode bloquear esta *thread*, pois a aplicação irá travar.

É possível realizar o acesso à rede por uma *thread* separada, e depois comunicar a *main thread* de que ela deve avisar o usuário de que o processamento terminou, pois somente a *thread* principal pode alterar a interface gráfica de uma aplicação Android.

Pensando nisso, tarefas assíncronas podem ser utilizadas para contornar este problema. Tarefas assíncronas são um recurso que permite executar tarefas em uma *thread* separada e automaticamente avisar a *thread* principal que o processo foi concluído, e assim sendo possível atualizar a interface gráfica.

Para utilizá-las, é preciso estender a classe `AsyncTask`. Ao fazer isso, é possível sobrecrever alguns métodos importantes:

- **`onPreExecute()`**: é executado na *main thread* antes da *thread* separada ser criada para executar o acesso à rede, portanto é possível avisar o usuário da execução do processo através da criação de uma barra de progresso, por exemplo;
- **`doInBackground()`**: este método é o único que é executado em uma *thread* separada, ou seja, será responsável por fazer o acesso à web e retornar o resultado para o terceiro método, pois, lembre-se, este método não pode alterar a interface gráfica;
- **`onPostExecute()`**: este é o método responsável por mostrar uma mensagem de sucesso ou erro para o usuário, dependendo da resposta retornada pelo servidor. Este método é executado exatamente após o `doInBackground()` terminar de executar.

Tarefas assíncronas desempenham um papel muito importante em uma aplicação Android, pois sabem quando devem executar um processamento e quando devem prover feedback ao usuário. O seu uso também permite economizar a escrita de código, pois em uma classe é possível receber parâmetros, acessar à rede e avisar o usuário.

3.18 Segurança

Segundo Six (2012), com a plataforma Android rapidamente se tornando um alvo para hackers mal-intencionados, a segurança de aplicativos é crucial. Os dados armazenados em uma aplicação vão de uma informação simples de uma nota do usuário para determinada disciplina até informação sensível de usuário e senha de acesso ao sistema. Neste tópico vamos analisar alguns aspectos importantes de proteção de aplicações Android, como encriptação de dados, interação com base de dados e permissões de acesso.

3.18.1 Encriptação de dados armazenados

Encriptação pura e simples se refere ao processo de pegar uma mensagem (que iremos chamar de texto simples) e transformá-la em uma versão misturada de si mesma (que iremos chamar de texto cifrado) (Six, 2012, p. 87). A encriptação de dados é um passo importante para garantir a integridade dos dados armazenados. É considerado uma boa prática de segurança armazenar dados pessoais de forma codificada ou encriptada. Six (2012) cita que a teoria é que é seguro ter o texto cifrado comprometido, já que um atacante não será capaz de deduzir o texto simples somente a partir da forma misturada.

Um texto encriptado geralmente é produzido através de um algoritmo de encriptação e uma chave de encriptação, que pode ser **simétrica** ou **assimétrica**, como veremos a seguir.

Chave Simétrica

Chave simétrica é uma chave utilizada em criptografia para gerar texto criptografado e obter o texto original utilizando a mesma chave. Pode-se voltar do texto cifrado ao texto simples (descriptografar) se o algoritmo e a chave forem conhecidos (Six, 2012, p. 87). Diferentemente da chave simétrica, a chave assimétrica usa chaves diferentes para encriptar e descriptografar dados: uma chave é utilizada para criar texto cifrado e outra chave para voltar ao texto original.

Algoritmo AES

Advanced Encryption Standard (AES) é um algoritmo de chave simétrica criado por Vincent Rijmen e Joan Daemen em uma competição que visava criar uma ferramenta de criptografia que substituísse o DES (*Data Encryption Standard*). Conforme Six (2012, p. 88) “hoje, o AES é o padrão para encriptação de chave simétrica”.

Dentre algumas características deste algoritmo, podemos citar:

- Disponibilizado livremente;
- Usa chave simétrica;
- Chave de encriptação pode ser aumentada.

Além de sua eficiência, o AES também foi projetado para permitir a expansão da chave quando necessário Trevisan et al. (2013). O tamanho da chave de encriptação está diretamente relacionada com a força do conteúdo encriptado que será produzido. Segundo Six (2012), quanto mais longa a chave, mais forte a proteção oferecida pela encriptação.

Neste projeto iremos encriptar os dados de acesso do usuário da aplicação. A autenticação, por sua vez, será abstraída da aplicação e será feita no servidor, visando maior proteção.

3.18.2 Protegendo consultas no banco de dados

O banco de dados de uma aplicação pode conter informações pessoais dos seus usuários e sua exposição aos usuários mal-intencionados é uma falha de segurança gravíssima. Nesta seção vamos abordar algumas técnicas de proteção à base de dados.

SQL Injection

Os dados já salvos no banco de dados também merecem atenção, pois existem técnicas em que os atacantes exploram vulnerabilidades através de comandos SQL, visando ter acesso a dados sensíveis. Um dos ataques SQL mais comuns é o SQL Injection, onde código SQL é “injetado” como entrada de dado, tirando proveito de consultas SQL vulneráveis. O principal fator motivador deste tipo de ataques em aplicativos é a falta de validação da entrada do usuário. Segundo Six (2012) “os aplicativos confiam que a entrada fornecida a eles é o que eles esperam”.

Prepared Statements

Outro fator motivador de ataques por injeção de código é a montagem das consultas SQL. É considerada boa prática de programação desvincular os parâmetros das consultas em si. Pensando nisso, os *prepared statements* estão presentes na maioria das linguagens de programação atuais para auxiliar na implementação de *queries* seguras. Esta classe permite realizar consultas parametrizadas, compilando a consulta e seus argumentos separadamente, facilitando a identificação de código malicioso.

Quaisquer interações com bases de dados baseadas em SQL devem sempre usar pesquisas parametrizadas. Você nunca deve formar uma declaração concatenando comandos e dados, porque a base de dados não seria capaz de diferenciá-los (Six, 2012, p. 135).

Pensando nisso, os scripts PHP que realizam a interação com o servidor utilizam a classe *prepared statement* para realizar suas consultas no banco de dados. Além disso, a aplicação Android faz a validação dos dados, evitando que o usuário entre com dados incompatíveis com os requisitados.

3.18.3 Permissões do Android

Permissões no Android informam o usuário no momento da instalação da aplicação, quais recursos do aparelho a aplicação poderá usar. Estes recursos são definidos no arquivo de manifesto do Android e devem ser precisos, isto é, só deve listar os recursos que realmente irão ser acessados pelo aplicativo.

Permissões também ajudam os usuários a identificarem possíveis ameaças em aplicações que estão sendo instaladas. Por exemplo, se uma aplicação de imagem e multimídia está sendo instalada, é normal que a aplicação requisiite autorização para acessar a câmera do

dispositivo. Por outro lado, um aplicativo tocador de música seria suspeito se pedisse para ter acesso a dados da conta do usuário.

Uma pergunta legítima que surge naturalmente, quando se entende esse modelo de permissões, é: “Os usuários realmente leem as requisições de permissão de um app e tomam uma decisão informada sobre instalar ou não um app?” Basicamente, os usuários aprovarão quaisquer permissões estabelecidas para um app que eles querem usar (Six, 2012, p. 48).

Este é uma problemática comum quando se trata de permissões no Android. Ao decidir instalar uma aplicação, as permissões que aquela aplicação utiliza são mostradas de uma só vez, em uma lista onde, raramente os usuários se importam em ler.

3.18.4 Permissões em tempo de execução

Pensando nisso, o Android introduziu as *Runtime Permissions* ou permissões em tempo de execução. Segundo Google, Inc. (2016) “os usuários concedem permissões para os aplicativos enquanto eles estão sendo executados, não quando são instalados”.

Este novo recurso introduzido no Android 6.0 (API Level 23) possibilita que os usuários eventualmente revoguem alguma permissão e, mesmo assim, continuem usando o aplicativo.

3.19 Google Cloud Messaging

O Google Cloud Messaging, ou simplesmente GCM, é um serviço que permite que aplicações servidoras enviem mensagens para aplicações Android Glauber (2015). Para utilizar este recurso é necessário um aparelho com Android 2.2 (API Level 8) ou superior e ter o Google Play Services – aplicativo que permite a utilização das APIs do Google – instalado.

Segundo Glauber (2015, p. 462) “o intuito não é trafegar uma grande massa de dados, mas sim enviar mensagens simples de até 4KB. Embora muito flexível, o GCM é geralmente utilizado para avisar aplicações clientes sobre atualizações no servidor. Essa implementação é muito comum para notificar os usuários que algo novo está disponível e que ele deve dar atenção a isso.

Para utilizar o GCM é preciso de um dispositivo real executando o Android. Isso porque o emulador da plataforma não é capaz de instalar o Google Play Services (necessário para rodar). É possível utilizar um emulador alternativo chamado Genymotion¹⁰ (<https://goo.gl/yE4kRI>) com alguns ajustes. Embora seja possível executar no Genymotion, iremos utilizar um dispositivo real neste projeto.

Para receber uma mensagem do GCM, a aplicação não precisa estar executando naquele momento, uma vez que o sistema Android envia uma mensagem de *broadcast*, que é recebida

¹⁰O Genymotion não vem com o Google Play Services instalado por padrão, porém é possível realizar os ajustes necessários para o seu funcionamento. Esta alternativa pode ser encontrada neste endereço: <http://goo.gl/5pkyIR>.

pelo sistema e identificada pela aplicação. A partir daí, podemos realizar ações em nosso aplicativo.

3.19.1 Arquitetura

A arquitetura do GCM tem como principais elementos os componentes físicos e credenciais (Glauber, 2015, p. 463). Esta arquitetura envolve o dispositivo móvel, que é independente de plataforma, a aplicação servidora que registrará os dispositivos no servidor e a aplicação servidora que enviará mensagens para estes dispositivos cadastrados.

Além da aplicação servidora e dos dispositivos clientes, é preciso realizar etapas de validação das credenciais, tanto no aplicativo, quanto no servidor. Segundo Glauber (2015), as credenciais são identificadores e chaves utilizadas durante as etapas do processo de registro e comunicação da aplicação servidora com o dispositivo móvel.

Nesta etapa de credenciais, algumas chaves são geradas pelo serviço, são elas:

- ***Sender ID*** - Identificador do projeto obtido ao registrar a aplicação no GCM;
- ***Application ID*** - Identificador único do projeto registrado no GCM, por padrão é definido o nome do pacote da aplicação, já que não é permitido ter mais de uma aplicação com o mesmo nome de pacote no Android;
- ***Registration ID*** - Esta chave é gerada pelo servidor quando uma aplicação cliente se conecta a ele. É necessário enviar esta chave ao se comunicar com o servidor GCM, para que seja feita a autenticação do dispositivo;
- ***Server API key*** - É a chave única gerada pelo servidor para sua aplicação. Esta chave é necessária para que o servidor possa enviar a mensagem para todos os dispositivos que têm o aplicativo instalado e deve ser concatenada ao cabeçalho da requisição no servidor.

Vejamos agora na figura 3.6 como é o fluxo de execução do GCM:

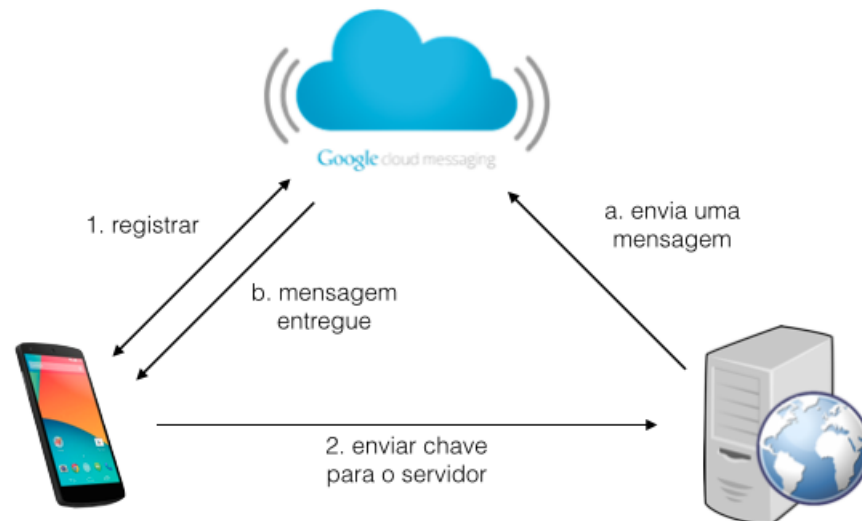


Figura 3.6: Arquitetura do Google Cloud Messaging.

Fonte: Glauber (2014)

Primeiramente o dispositivo deve ser registrado no servidor, conforme mostrado no fluxo 1. Ao instalar a aplicação, o dispositivo se conecta com o servidor, solicitando o *Registration ID*. O aplicativo deve guardar esta chave, pois ela será usada posteriormente para enviá-la à aplicação servidora. Neste projeto, iremos armazenar esta chave localmente e na base de dados remota, juntamente com o identificador único (ID) do aluno.

Após registrar a aplicação no servidor GCM, é possível enviar mensagens para a aplicação. Para isso, a aplicação servidora utiliza o *Sender ID* e *Registration ID* – que são enviadas para o servidor (fluxo 2) – para identificar quais usuários irão receber a mensagem (fluxo a e b), por isso precisamos gerar estas chaves nas etapas anteriores.

3.20 Gradle

O Gradle (<http://www.gradle.org>) é uma avançada ferramenta de automação do processo de *build* baseada em *plug-ins* utilizada pelo Android Studio (Glauber, 2015, p. 740). Esta ferramenta permite a personalização do processo de *build*, permitindo criar variações da mesma aplicação, como também facilitar o processo gerenciamento de dependências. Segundo Glauber (2015, p. 740), “ela combina o poder e a flexibilidade do Ant (<http://ant.apache.org>) com o gerenciamento de dependências do Maven (<http://www.maven.apache.org>)”.

Com o gradle, é possível criar variações do mesmo projeto (*build variants*), aumentando a produtividade. Por exemplo, é possível criar uma versão gratuita e comercial do mesmo aplicativo sem precisar duplicar código. Além disso, o processo de injeção de dependências

é muito utilizado, como veremos a seguir.

3.20.1 Dependências

O gerenciamento de dependências do Maven permite adicionarmos bibliotecas em nosso projeto. Este recurso é muito importante, pois flexibiliza o desenvolvimento da aplicação e nos permite utilizar recursos da plataforma essenciais para o funcionamento do nosso aplicativo, assim como bibliotecas de terceiros, que podem adicionar funcionalidades interessantes em nosso projeto.

Segundo Glauber (2015), o gerenciamento de dependências suporta tanto dependências locais (no sistema de arquivos do computador) quanto remotas (em servidores da web). Ao utilizar dependências remotas, o próprio gradle se encarrega de baixar a biblioteca e sincronizar com o código da aplicação. Para compilar a dependência, é preciso adicionar o nome do pacote da biblioteca no arquivo **app.gradle**, disponível na raiz do projeto.

A utilização de dependências do Maven disponível no Android Studio é equivalente à operação de adicionar uma biblioteca externa na IDE Eclipse. Entretanto, o AS com auxílio do *framework* gradle executa o processo de download e sincronização com o projeto de forma automática.

3.21 Alocação dinâmica de recursos

Recursos de uma aplicação é tudo que pode ser utilizado no desenvolvimento da mesma, incluindo: strings (texto simples), array de strings (lista de itens), layouts (telas), estilos, etc. Com isso, é possível reutilizar os recursos de uma aplicação com o intuito de adaptar sua aplicação a uma mudança de estado. Um exemplo da utilização disto seria a adaptação da sua aplicação quando o usuário gira a tela do smartphone. Quando isso acontece, o Android automaticamente “destrói” a activity que estava executando e a recria, pois entende que houve uma mudança de configuração do dispositivo.

O Android tem um conceito de alocação dinâmica de recursos, em que o sistema operacional seleciona o recurso mais apropriado de acordo com a configuração do aparelho (Glauber, 2015, p. 46). Com este mecanismo, podemos definir diferentes recursos para diversos tipos de configurações, tornando nossa aplicação mais amigável rodando em qualquer aparelho. Neste projeto, basicamente vamos utilizar a seleção dinâmica dos recursos para: tamanhos de tela, rotação da tela, versões do Android e para suporte a tablets.

3.21.1 Recursos de Imagem

O tamanho de uma tela de um smartphone é medido em dpi ou *Dots per inch* (pontos por polegada). Onde um aparelho com uma tela com maior dpi, possui qualidade de imagem superior e, ao mesmo tempo, precisam de recursos com maior resolução para serem exibidos sem distorções. Os recursos de imagem da aplicação ficam dentro das pastas *mipmap* dentro

do pacote da aplicação. Depois do “-” no nome da pasta, são definidos os qualificadores do diretório, que é como o Android diferencia qual recurso mais apropriado a ser utilizado para o aparelho que está executando a aplicação naquele momento. Cada pasta *mipmap* com seu respectivo qualificador contém imagens em diferentes resoluções, que basicamente são 5, como podemos ver na tabela abaixo:

Tabela 3.2: Densidades de tela.

Qualificador	DPI	Proporção
mdpi	160 dpi	1
hdpi	240 dpi	1.5
xhdpi	320 dpi	2
xxhdpi	480 dpi	3
xxxhdpi	640 dpi	4

Podemos concluir através desta tabela que imagens para aparelhos com densidade **xhdpi** devem ter o dobro do tamanho de imagens para dispositivos **mdpi**. Gerar arquivos de imagens (por exemplo logo da aplicação) em diversos tamanhos e colocando-os nas suas devidas pastas com os qualificadores garante que o Android escolha o recurso mais indicado para ser exibido, deixando a aplicação homogênea, independente do aparelho que está executando. Por outro lado, se não forem definidos os qualificadores e todas as imagens forem disponibilizadas na mesma pasta e com o mesmo tamanho, o Android terá que redimensioná-las para as exibir em aparelhos com o tamanho de tela diferente. O resultado dessa transformação pode ser desagradável para o usuário, já que uma imagem distorcida pode diminuir drasticamente a qualidade da aplicação.

3.21.2 Orientação

Além dos recursos de imagens, é possível usar os qualificadores do Android para definir diferentes arquivos de layout (tela) para quando o aparelho mudar de orientação, por exemplo: retrato para paisagem e vice-versa. Quando o aparelho é rotacionado para o modo paisagem, a aplicação ganha mais espaço na largura da tela e perde espaço na altura, ou seja, é interessante que seja feita uma adaptação para aproveitar melhor os espaços disponíveis na tela.

Tabela 3.3: Diretórios de layouts e descrição.

Diretório	Descrição
res/layout	Nesta pasta ficam os arquivos padrão de layout da sua aplicação
res/layout-land	Esta pasta contém os arquivos de layout em modo paisagem

Para fazer isso é muito simples, basta criar uma pasta com o qualificador “land” (*landscape* ou paisagem) desta forma “layout-land” e copiar os arquivos da pasta layout que deseje

alterar e colar na pasta `land` (com o mesmo nome). Feito isso, basta fazer as alterações necessárias nos arquivos modo paisagem que o Android se responsabiliza de todo o resto, desde a identificação de uma mudança de orientação até a chamada do arquivo de layout na pasta personalizada.

3.21.3 Versões

Da mesma forma, é possível definir valores de dimensões (pasta `dimens`), estilos customizados (pasta `styles`), Strings (pasta `strings`) e cores (pasta `colors`), para diferentes versões do Android. Partindo do conceito de que versões do Android mais recentes não suportam alguns recursos mais antigos da plataforma e vice-versa, é possível criar recursos diferenciados para cada versão, evitando que a aplicação trave ao executar, já que exceções de incompatibilidade são lançadas em tempo de execução. Embora a API de compatibilidade permita padronizar o design das aplicações em diferentes versões, ajustes específicos para cada versão da plataforma precisam ser feitos. A tabela 3.4 mostra os diretórios `values` criados neste projeto para alocar os recursos para as diferentes versões suportadas.

Tabela 3.4: Pastas `values` para diferentes versões da aplicação.

Diretório	Descrição
<code>values</code>	Pasta onde ficam os recursos em comum para todas as versões
<code>values-v19</code>	Recursos para versões acima da API level 19 (Android 4.4 Kitkat)
<code>values-v21</code>	Recursos para versões acima da API level 21 (Android 5 Lollipop)

Através destas pastas é possível informar o Android que alguns recursos estarão disponíveis somente em algumas versões. Um exemplo disso é o efeito *ripple*, que proporciona feedback visual quando o usuário clica em um item. Este efeito foi adicionado à plataforma juntamente com o material design, na API 21, ou seja, só estará disponível para aparelhos que possuem o Android 5.0 ou superior. O efeito é mostrado na figura 3.7.

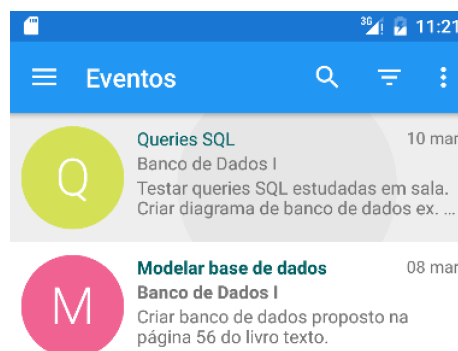


Figura 3.7: *Ripple effect* ao clicar em um item, disponível a partir da versão 5.0.

3.21.4 Suporte a Tablets

Outra possibilidade interessante é adaptar a aplicação para aparelhos com telas muito maiores do que os smartphones convencionais: os tablets. Como os tablets têm tamanho de tela muito superior às dos smartphones, é possível adaptar o aplicativo para tirar vantagem desta característica. Por padrão, se nenhum recurso for definido, o Android simplesmente vai executar a mesma aplicação no celular e no tablet. Podemos adaptar nossa aplicação para uma melhor experiência do usuário, aproveitando o tamanho da tela para diminuir cliques.

Tabela 3.5: Pasta contendo recursos para tablets.

Diretório	Descrição
values-sw600dp	Pasta onde ficam os recursos para tablets com tela acima de 600 dpi.
values-w820dp	Pasta onde ficam os recursos para tablets com tela acima de 820 dpi.

Os recursos que estão dentro destas pastas serão carregados se a aplicação estiver sendo executada a partir de um tablet. Além deste qualificador mostrado na tabela 3.5, podemos criar um arquivo do tipo bool (*boolean*) que irá identificar se estamos executando em um smartphone ou em um tablet. Isso é interessante pois, se estivermos em um tablet, podemos executar o aplicativo em modo *dual pane* (dois painéis), tornando o menu lateral fixo, evitando um clique do usuário.

3.22 Ferramentas para web

A aplicação web irá gerenciar o conteúdo disponibilizado para a aplicação Android, além de informar os usuários quando uma nova informação estiver disponível no servidor. Para total integração com a aplicação *mobile*, alguns *frameworks* foram utilizados, como mostraremos nas próximas seções.

3.22.1 Material Design Lite (MDL)

Segundo definição do próprio site “Material Design Lite permite que você aplique o Material Design em seus websites”. Este *framework* permite representar a experiência do design material do Google em sistemas web. É desenvolvido e mantido pela equipe de desenvolvedores do Google e pode ser encontrado através deste endereço: <http://www.getmdl.io/index.html>.

3.22.2 Materialize

Esta ferramenta é semelhante à citada anteriormente, pois permite recriar a experiência do Material Design para web. Porém, como o MDL está em estágios iniciais, alguns componentes ainda estão ausentes. Por este motivo, iremos utilizar o Materialize, que é mantido por uma

equipe de estudantes da Carnegie Mellon University em Pittsburgh, Pennsylvania. Este framework pode ser encontrado neste endereço: <http://materializecss.com/>.

3.22.3 Data Table

Segundo próprio site “DataTables é um *plug-in* para biblioteca jQuery em Javascript. É uma ferramenta altamente flexível, baseada nos fundamentos do avanço progressivo, e irá adicionar controles avançados em qualquer tabela HTML”. Esta biblioteca permite adicionar funcionalidades interessantes às tabelas padrões HTML, tornando-as dinâmicas. Entre os recursos que podem ser utilizados, temos: paginação automática, limitar números de registros exibidos, exportar células da tabela em diversos formatos, como: PDF e planilha do Microsoft Excel, barra de busca, etc. É altamente customizável e apresenta suporte para diversos outros *frameworks*, como o próprio MDL. Data Table está disponível no endereço: <https://datatables.net/>.

No próximo capítulo iremos detalhar o processo de desenvolvimento deste trabalho, desde a definição dos requisitos da aplicação, sua modelagem, preparação do ambiente, implementação, validação e licença de uso.

Capítulo 4

UFMS Mobile

O desenvolvimento deste projeto se divide em diversas etapas que, em linhas gerais, permitiram elaborar uma maneira eficiente para que um aplicativo Android se comunique com uma aplicação web através de ferramentas da plataforma. Neste capítulo iremos detalhar as etapas executadas na implementação deste trabalho.

4.1 Roteiro de Desenvolvimento

Nesta seção iremos abordar o plano de desenvolvimento seguido e executado neste projeto. As etapas que serão demonstradas a seguir nos levaram à criação de protótipos de nossa aplicação.

4.1.1 Visão Geral

Este aplicativo foi implementado para executar em smartphones e tablets com o sistema operacional Android que têm acesso à internet. A aplicação deve ser capaz de mostrar os eventos adicionados pelos professores das disciplinas em que o aluno está matriculado, as notas através de uma lista e de gráficos de análise de desempenho, materiais disponíveis para download, alunos matriculados na mesma turma e informações do curso.

4.1.2 Requisitos

Um projeto planejado corretamente deve incluir uma boa documentação que dê suporte a todo o processo de construção do sistema. Neste trabalho, um documento de requisitos foi elaborado para auxiliar na fase de desenvolvimento da aplicação e, posteriormente, realizar a validação e testes do projeto. Este documento está disponível no apêndice A deste trabalho.

4.1.3 Modelos Criados

O consumo de conteúdo por parte dos alunos será feito através da aplicação para Android. A figura 4.1 apresenta o diagrama de casos de uso, explicitando as operações básicas que os alunos podem executar no aplicativo.

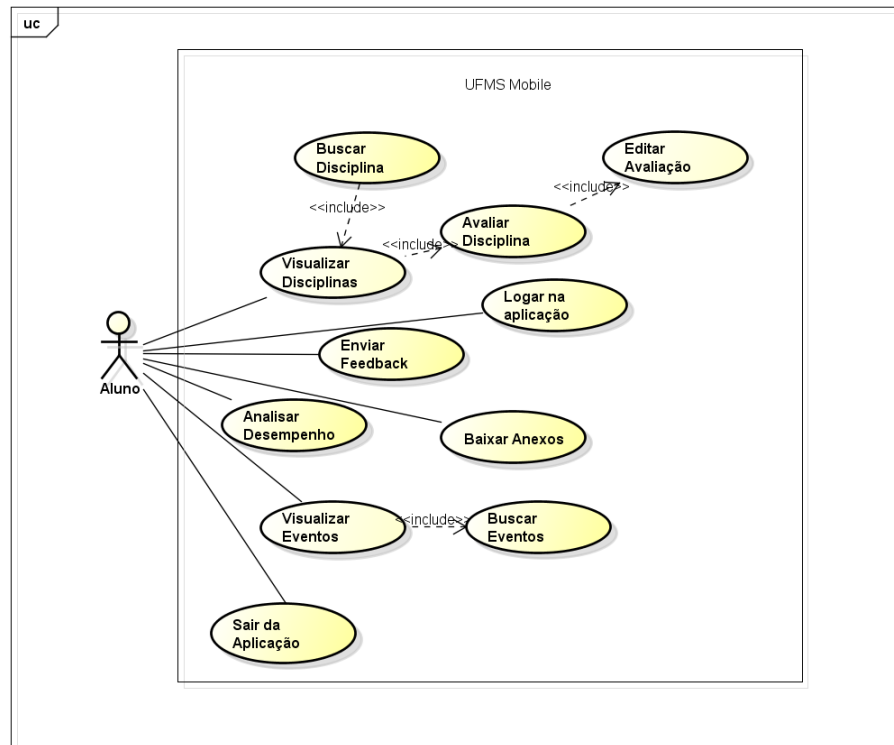


Figura 4.1: Casos de uso dos alunos.

A figura acima lista as funcionalidades que os alunos terão acesso na aplicação Android. Em linhas gerais, os alunos poderão:

- **Disciplinas** – Listar disciplinas, buscar disciplina, avaliar e editar avaliação referente à uma disciplina específica;
- **Feedback** – Enviar feedback da experiência de uso da aplicação para o desenvolvedor;
- **Desempenho** – Analisar o desempenho de notas nas disciplinas através de gráficos dinâmicos;
- **Acesso** – Realizar *login* e *logout* na aplicação;
- **Anexos** – Fazer download de anexos disponibilizados pelos professores;
- **Eventos** – Visualizar e buscar por eventos.

A interação com os professores das disciplinas será feita através da interface web desenvolvida. Os professores terão à disposição ferramentas para divulgar informação, arquivos, avisos, etc. As funcionalidades básicas deste usuário são representadas no modelo de casos de uso abaixo:

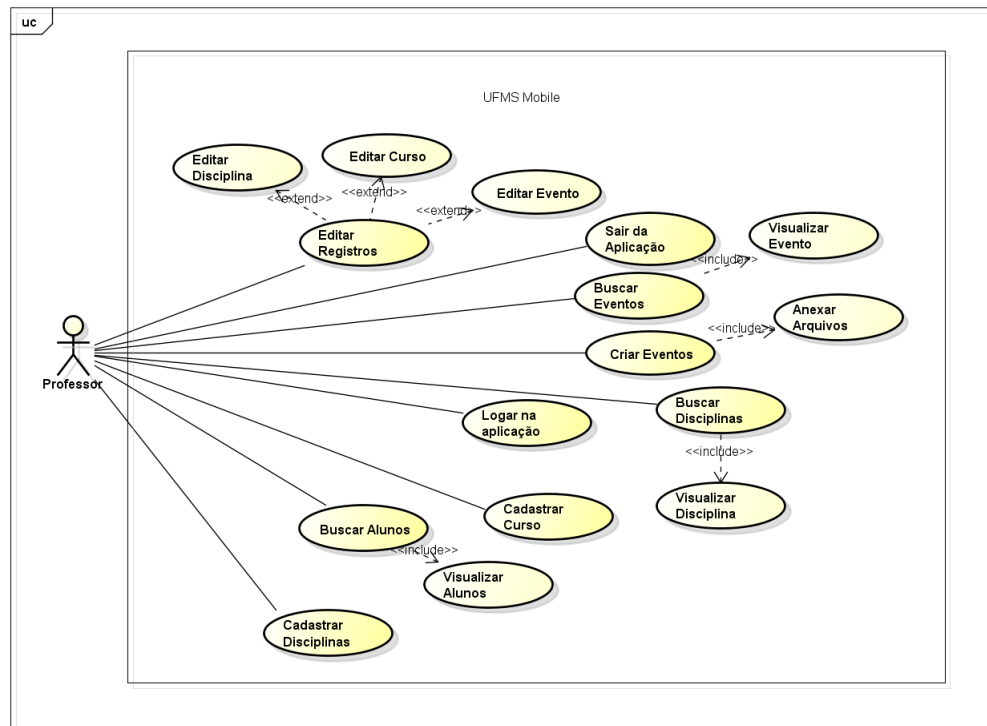


Figura 4.2: Casos de uso dos professores.

Na figura 4.2 são apresentadas as possíveis interações entre os professores e a aplicação. Dentre suas características, o sistema web apresenta:

- **Eventos** – Criar, buscar, visualizar e adicionar anexos a eventos;
- **Disciplinas** – Cadastrar, visualizar, editar e buscar disciplinas;
- **Curso** – Criar e adicionar informações acerca do curso;
- **Alunos** – Registrar os alunos às suas respectivas disciplinas. Visualizar e buscar alunos;
- **Acesso** – *Login* e *logout* na aplicação web.

A tabela de evento armazena informações a respeito do evento a ser criado, como nome, descrição, data de criação, data limite e ícone. Os eventos são ligados por chave estrangeira com disciplina e tipo de evento. A figura 4.3 ilustra este diagrama.

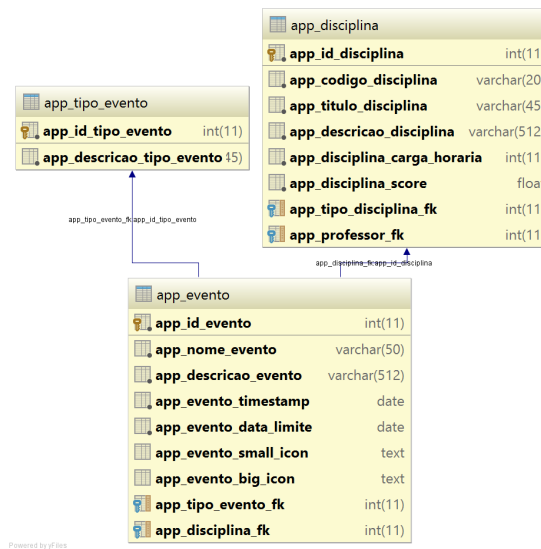


Figura 4.3: Trecho do diagrama relacional – tabela evento e suas ligações.

A tabela de disciplina grava informações pertinentes a respeito da disciplina, seu tipo (por exemplo, obrigatória, optativa, etc.) e o professor responsável. O diagrama abaixo representa uma disciplina na aplicação:

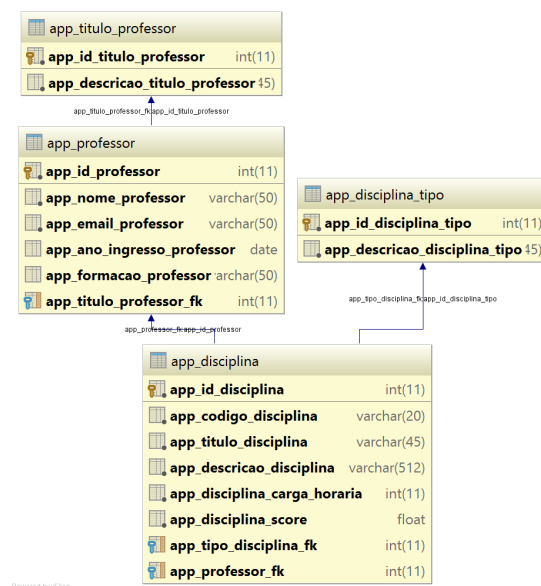


Figura 4.4: Trecho do diagrama relacional – tabela disciplina e suas ligações.

Os demais diagramas gerados no desenvolvimento deste projeto estão disponíveis no

apêndice B deste trabalho.

4.1.4 Protótipos da aplicação

Os protótipos da aplicação foram criados utilizando a ferramenta Justinmind (seção D.7) para criação de protótipos de telas. Estes protótipos possibilitaram visualizar como o sistema será desenvolvido do ponto de vista da interface gráfica, além de permitirem que as diretrizes de design propostas pelo Material Design sejam demarcadas, auxiliando a criação das telas da aplicação. No decorrer desta seção será apresentado o conjunto de protótipos de telas criado.

Telas de acesso à aplicação

As telas de acesso à aplicação – login e cadastrar – representam a forma como os usuários acessam o aplicativo Android, seja o acesso pela primeira vez, ou um usuário que está retornando à aplicação. Os protótipos ilustrados na figura 4.5 mostram como estas telas devem ser visualmente e demarcam suas propriedades: cores, tamanhos e tipos de entrada de dados.

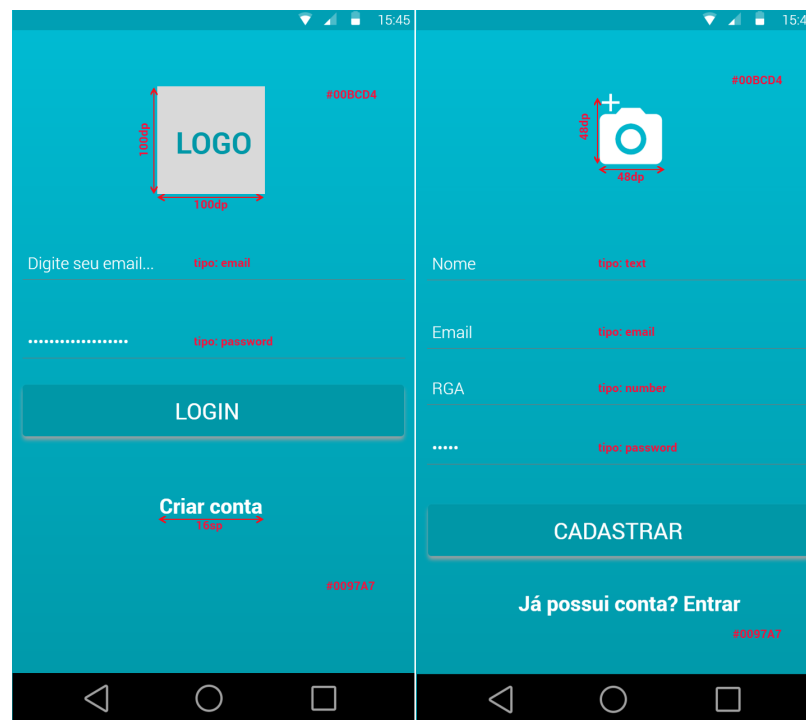


Figura 4.5: Protótipos das telas de acesso à aplicação Android: Login e Cadastrar.

Detalhes da disciplina

A tela de detalhes de uma disciplina é subdividida em abas, de forma a organizar o conteúdo a ser mostrado. Estas abas são: Informações, Alunos matriculados e Materiais disponíveis

(anexos). Na primeira aba, a aba de informações, são mostradas ementa, tipo de disciplina, professor responsável, nota média – atribuída pelos alunos –, e quantidade de avaliações que a disciplina teve. Na aba de alunos são apresentados os estudantes que participam da disciplina em questão. Por fim, a aba de materiais exhibe todos os anexos disponibilizados por eventos que pertencem à esta disciplina.

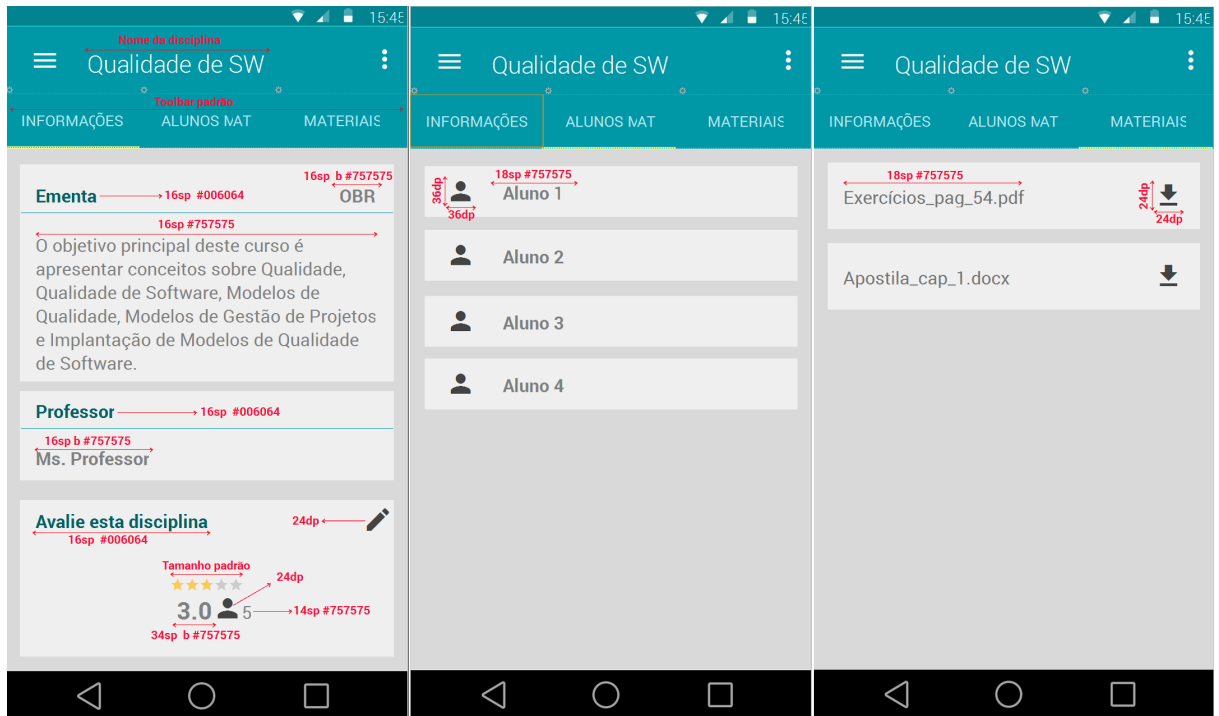


Figura 4.6: Protótipos das telas de detalhes da disciplina.

A figura 4.6 ilustra o protótipo destas telas de detalhes de disciplinas, demarcando o tamanho em sp – unidade de medida exclusiva para fontes no Android, equivalente ao pixel –, cor e estilo das fontes utilizadas, tamanho das figuras em dp – equivalente ao pixel – e a utilização padrão da Toolbar e do componente Ratingbar, que representa a avaliação do aluno através de estrelas de 1 a 5.

Detalhes do evento

A tela de detalhes dos eventos mostra as informações do evento criado, como: o nome do evento, sua descrição completa, a data limite (se houver), a disciplina associada, o professor responsável, permite habilitar ou desabilitar notificações e exibir anexos disponíveis.

O protótipo da tela de eventos foi desenvolvido em duas iterações. A primeira exibia todas as informações do evento em uma única tela, o que poderia ficar extensivo, dependendo do número de documentos anexados. Para organizar melhor o conteúdo exibido, a tela de detalhes foi dividida em duas abas: uma para as informações do evento e uma para anexos, como mostrado na figura 4.7.



Figura 4.7: Primeira e segunda iteração dos protótipos da tela de detalhes de eventos.

Telas de listagem de disciplinas e eventos

As telas de listagem de disciplinas e eventos exibem disciplinas e eventos cadastrados, respectivamente. Na figura 4.8 percebe-se que estas telas tem layout semelhante, buscando facilitar o aprendizado do usuário. Porém, algumas diferenças podem ser notadas, como: os eventos podem ser marcados como lidos ou não lidos, diferenciando a cor de fundo (branco ou cinza) das linhas destes eventos, destacando-os para acesso rápido do usuário.

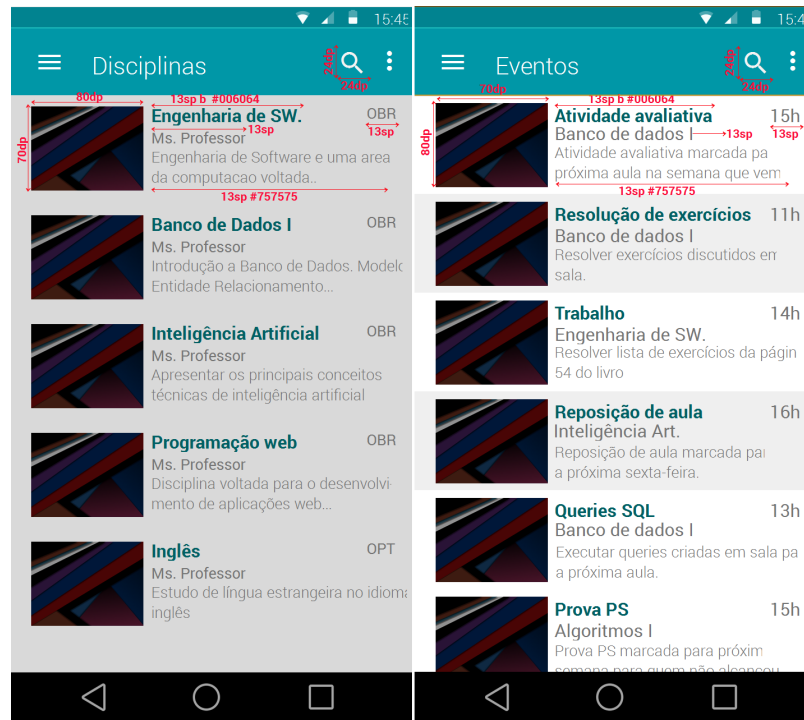


Figura 4.8: Protótipos das telas de lista de disciplinas e eventos.

Lista de disciplinas vazia ou sem conexão

A tela de lista de disciplinas exibe as disciplinas que o estudante está atualmente relacionado. Esta tela pode estar vazia em algumas situações, como: o aluno acabou de se cadastrar no aplicativo e está aguardando ser matriculado; o usuário perdeu a conexão com a internet antes de fazer a sincronia com o servidor ou o usuário está totalmente desconectado ao acessar a tela de lista de disciplinas. Quando estas situações ocorrem, a aplicação deve prover feedback para o usuário, informando a situação atual e também permitindo que o usuário possa “tentar novamente” caso tenha resolvido o problema da falta de conexão, por exemplo. O protótipo exibido na figura 4.9 exemplifica estes cenários.

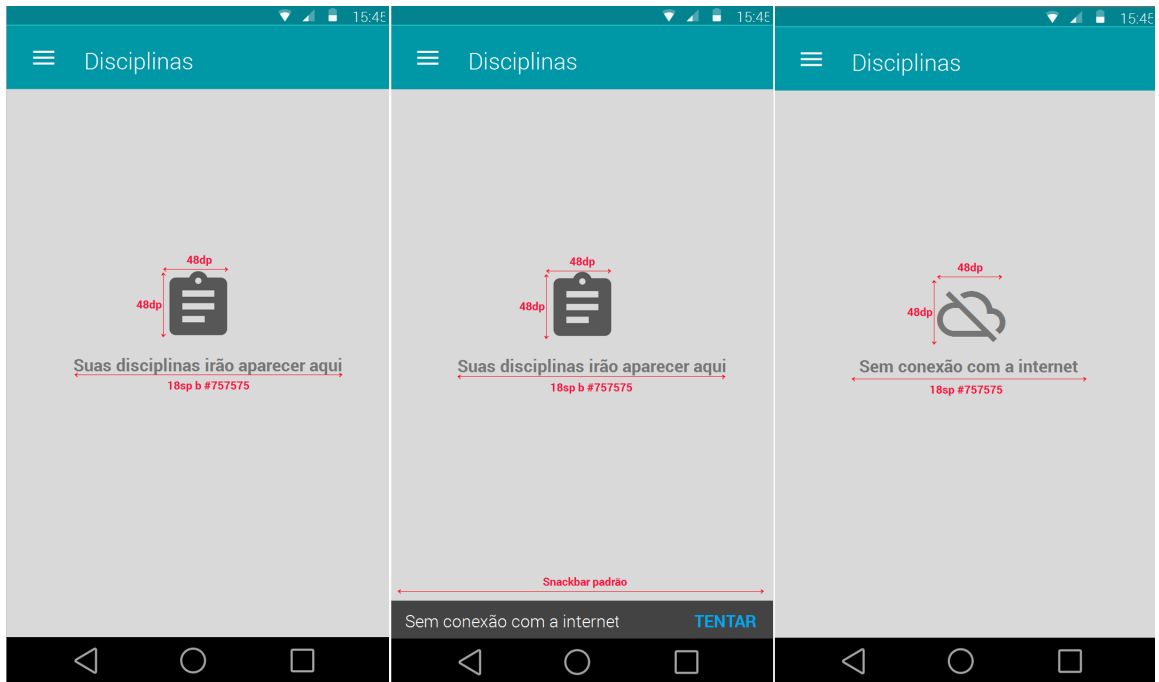


Figura 4.9: Protótipos das telas de lista de disciplinas quando estiverem vazias.

Lista de eventos vazia ou sem conexão

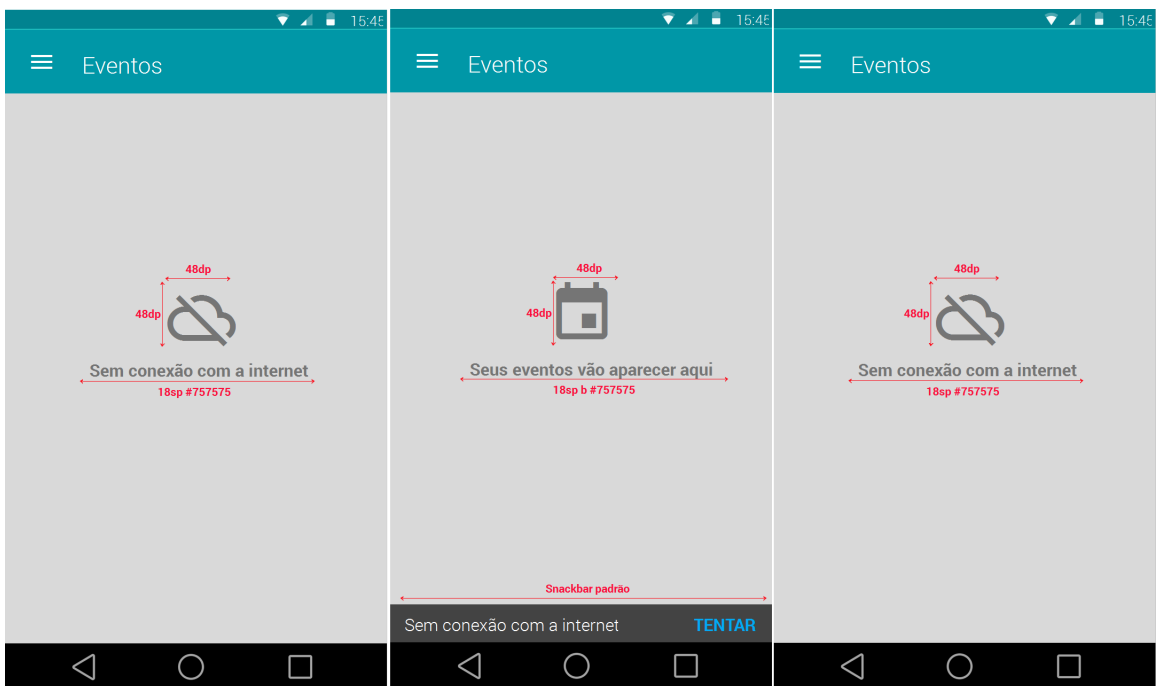


Figura 4.10: Protótipos das telas de lista de eventos quando estiverem vazias.

De forma semelhante aos protótipos apresentados no item anterior, as telas de lista de eventos também devem prover feedback ao usuário sobre a situação atual do sistema. A figura 4.10 ilustra as telas de lista de eventos vazias.

Acesso às notas do usuário

O acesso às notas do usuário é feito através de basicamente duas telas: a primeira exibe a disciplina cursada com a média atual, o período e o tipo da disciplina; a segunda exibe as tarefas avaliadas para determinada disciplina. Esta segunda tela – das atividades avaliadas – pode ser representada de duas formas: lista e desempenho. Em lista, as atividades são simplesmente listadas por ordem cronológica. Em desempenho, as notas são exibidas através de gráficos, permitindo que o aluno analise o seu desempenho, comparando suas notas em diversos aspectos (ordem crescente, desempenho histórico das avaliações e comparação entre nota individual com a média da turma), como mostrado nas telas 2 e 3, respectivamente, da figura 4.11.

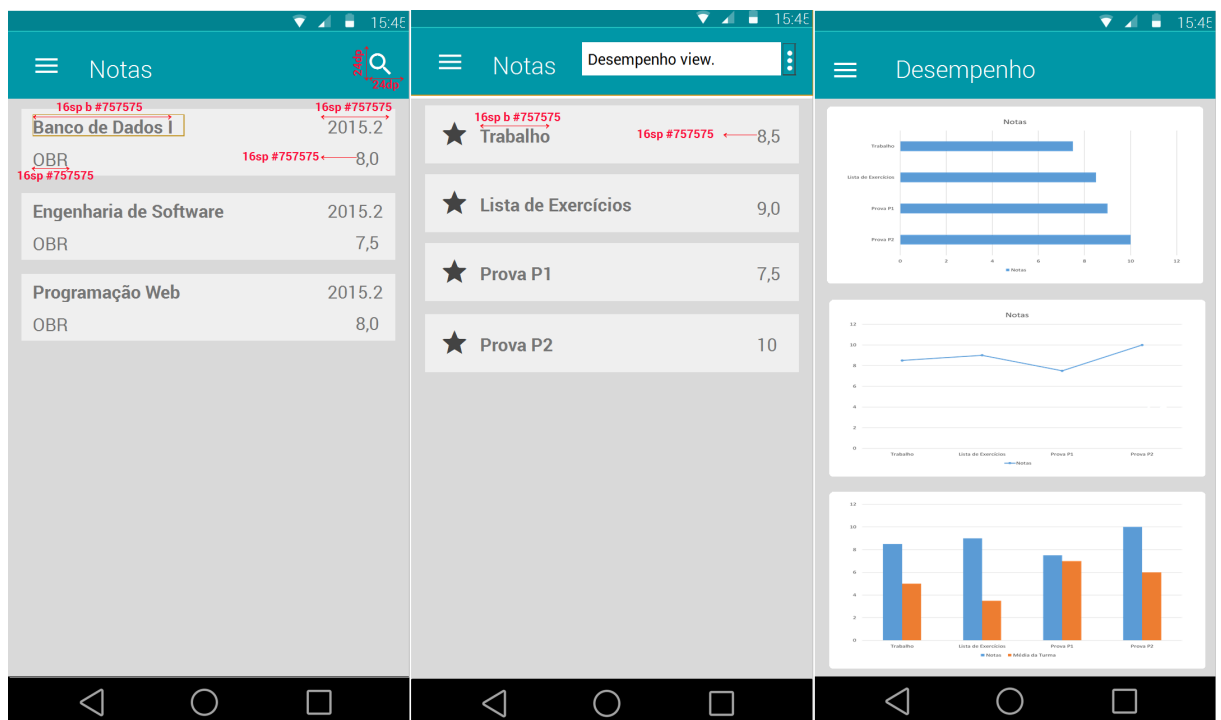


Figura 4.11: Protótipos das telas de acesso às notas do aluno.

Tela inicial e menu da aplicação

A tela inicial – ou tela “Explore” – apresenta um resumo dos últimos eventos adicionados e das disciplinas com avaliação mais positiva entre os alunos. Já o menu de opções da aplicação agrupa os módulos em que a aplicação Android é dividida. O menu de opções deve seguir as

regras e diretrizes do Material Design¹, que estão demarcadas no protótipo da tela. A figura 4.12 exibe a tela com o menu de opções e a tela inicial, respectivamente.

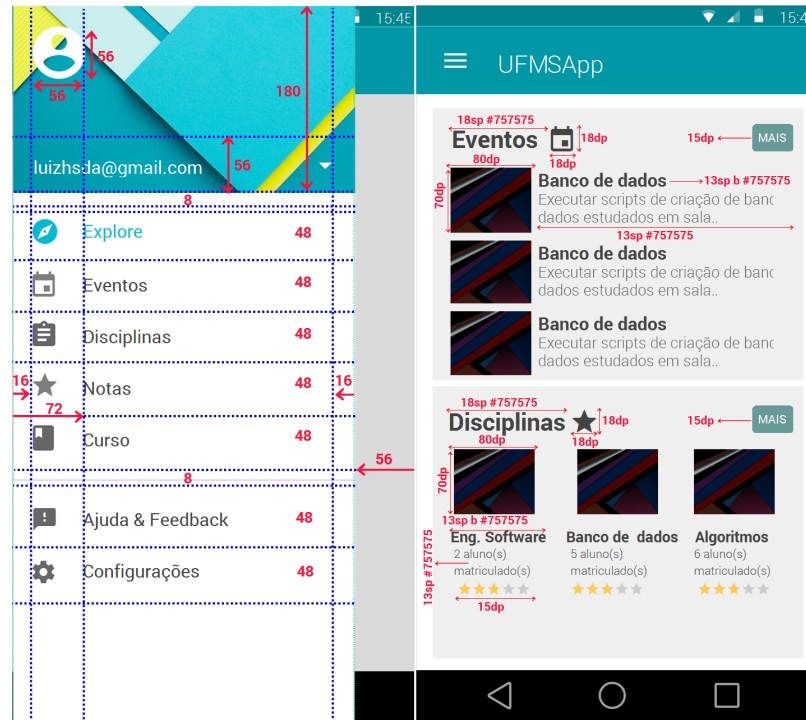


Figura 4.12: Protótipos das telas inicial e menu de opções.

Tela de configuração

A tela de configurações da aplicação Android representa as opções de preferência que os usuários têm acesso e podem configurar na aplicação. O layout desta tela é desenvolvido com auxílio do *framework* Preferences Framework (abordado na seção 3.12.4). Embora a construção da tela de configurações seja realizada com auxílio da ferramenta, a organização dos componentes é totalmente personalizada, definida pelo desenvolvedor.

¹Disponível em: <https://goo.gl/XYha4G>.

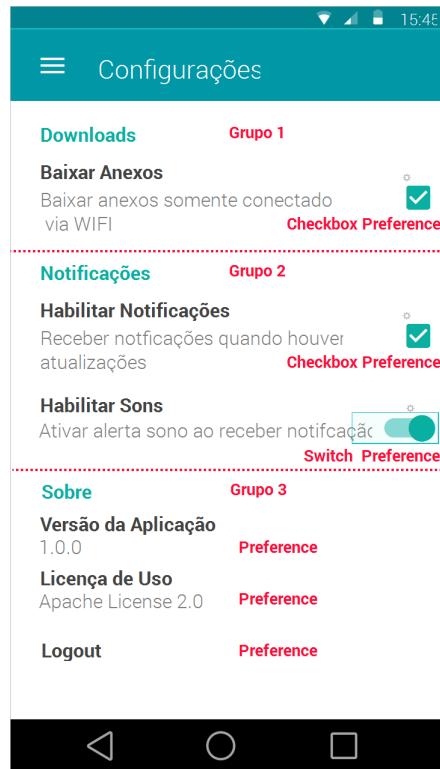


Figura 4.13: Protótipos da tela de configuração da aplicação.

A figura 4.13 ilustra o protótipo da tela de configurações desejada para a aplicação Android, assim como os componentes necessários para sua implementação: **checkbox**, **switch** e **preference**.

Páginas da aplicação web

Protótipos da aplicação web também foram criados, com base nos conceitos do Material Design e se baseando no design da aplicação Android.

A página inicial da aplicação web exibe uma breve descrição sobre este projeto, com link para descrição detalhada e os eventos mais recentes cadastrados. A figura 4.14 ilustra este layout. As cores da aplicação web seguem os padrões definidos para o aplicativo Android.

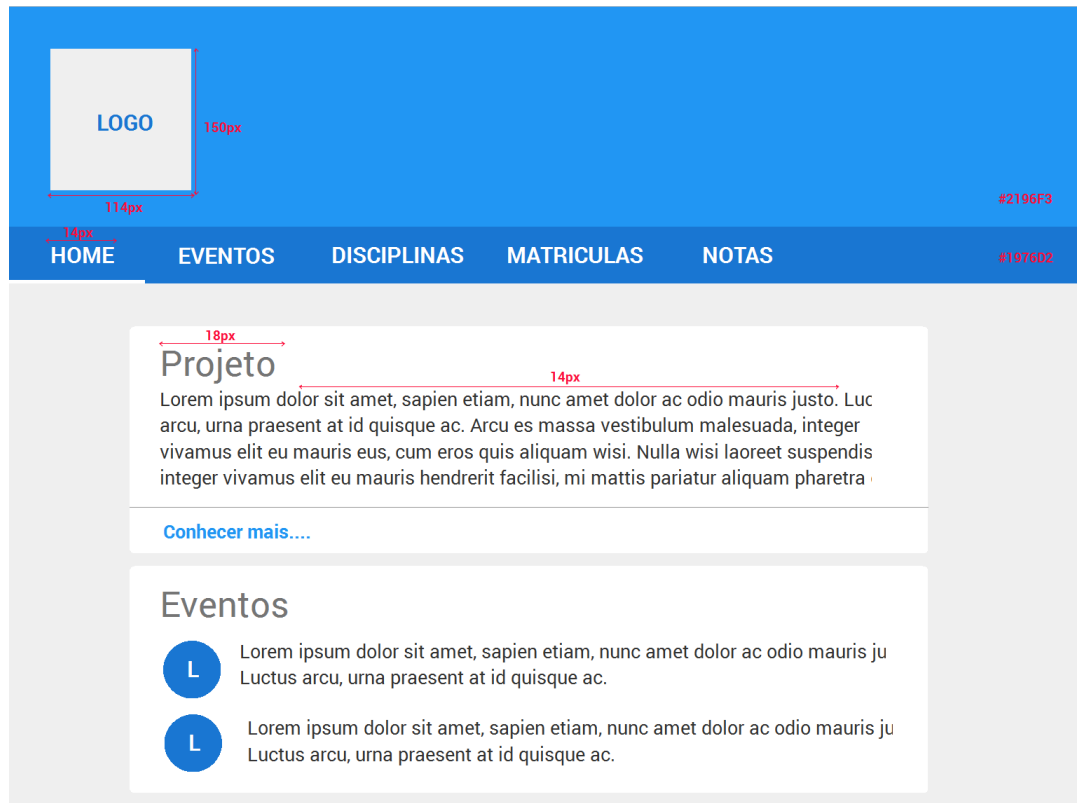


Figura 4.14: Protótipos da página inicial da aplicação web.

A página de eventos pode ser acessada ao clicar na opção “Eventos” no menu principal da aplicação. Esta tela exibe todos os eventos criados por ordem cronológica, assim como uma descrição detalhada, data de criação, tipo de evento (atividade, avaliação, aviso e geral) e disciplina associada. Na página de eventos existe um link (botão) para o formulário de cadastro de um novo evento, que é exibido na mesma página. O protótipo da página de eventos é exibido na figura 4.15.

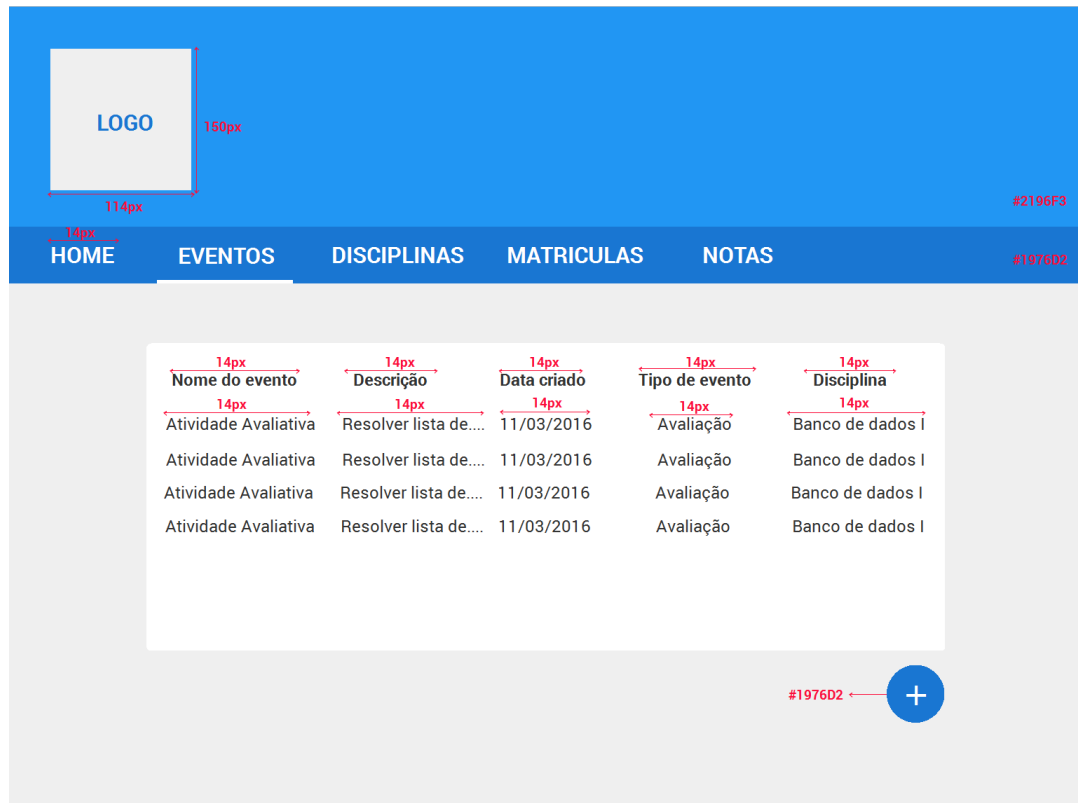


Figura 4.15: Protótipos da página de eventos da aplicação web.

O formulário para adicionar um novo evento é exibido na mesma página dos eventos. Alguns dados sobre o evento devem ser preenchidos pelo usuário para inserir um novo evento, como: nome do evento, descrição, data limite, tipo do evento, disciplina e anexos (se houver). Estes campos também possuem tipos de entradas de dados definidos, como demarcado no protótipo da figura 4.16.

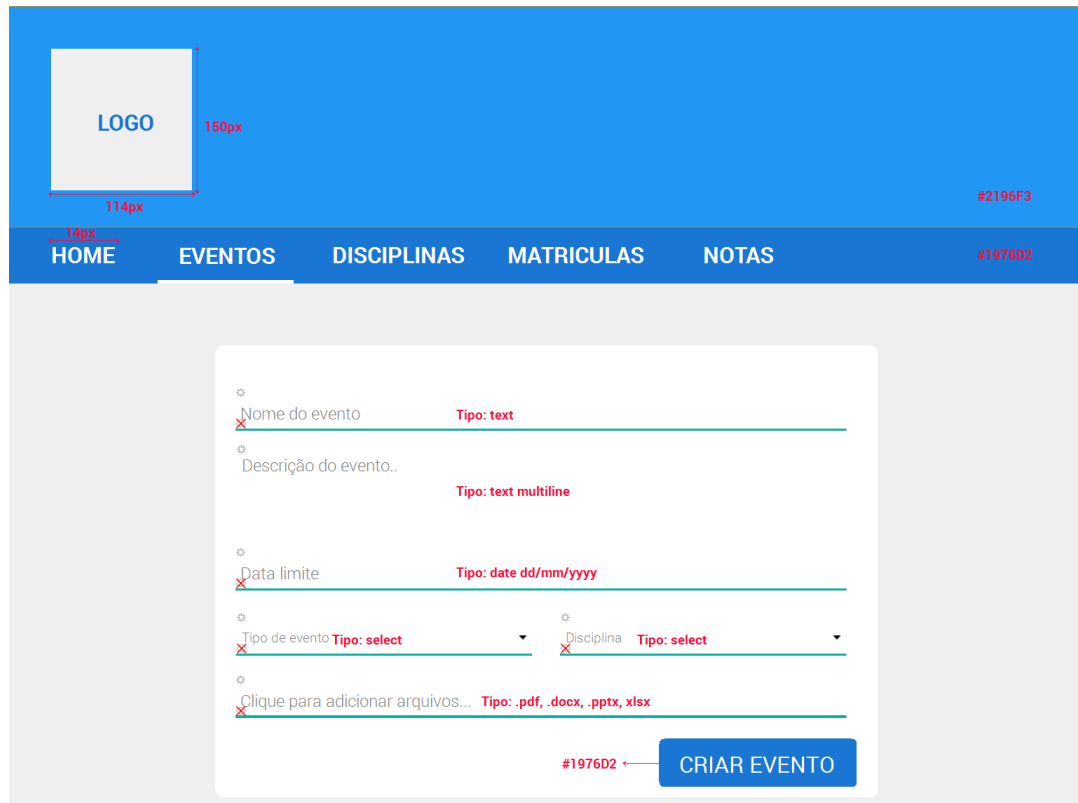


Figura 4.16: Protótipos da tela de adicionar evento.

4.2 Aplicativo UFMS Mobile

Esta seção aborda como foi realizada a etapa de desenvolvimento deste projeto, desde a etapa de instalação do ambiente até a implementação e validação.

4.2.1 Ambiente de Instalação

Para o desenvolvimento deste trabalho foi utilizado o pacote (*bundle*) do Android Studio contendo a IDE mais *Software Development Kit* (SDK) do Android, disponibilizado pelo Google². O SDK presente neste pacote inclui todas as ferramentas necessárias para desenvolvimento e depuração de aplicações. É possível baixar as ferramentas separadamente, mas é recomendado obter o pacote, uma vez que são imprescindíveis para o correto funcionamento. Além destas ferramentas, é preciso ter o JDK (*Java Development Kit*) instalado³.

O SDK presente no pacote baixado possui o SDK Manager, que é a ferramenta de gerenciamento do SDK do Android, possibilitando baixar diversos componentes para a fase de desenvolvimento, como: imagens do sistema operacional Android para executar em um

²Disponível para download neste endereço: <http://goo.gl/b930eB>

³O JDK pode ser baixado através do endereço: <http://oracle.com/java>

emulador, ferramentas de teste e driver para testar os aplicativos em dispositivos reais com Android.

Feita a instalação da IDE, é preciso baixar os componentes necessários para a criação da AVD (*Android Virtual Device*), que é um emulador disponível para testar aplicações em desenvolvimento. Embora seja possível rodar o aplicativo diretamente em um dispositivo real, é interessante fazer o download de algumas imagens do sistema Android para executar no emulador, já que temos praticamente todas as versões do sistema disponíveis para emulação. É desejável criar dispositivos virtuais em diferentes versões, pois podemos analisar o comportamento do nosso aplicativo em diversos ambientes.

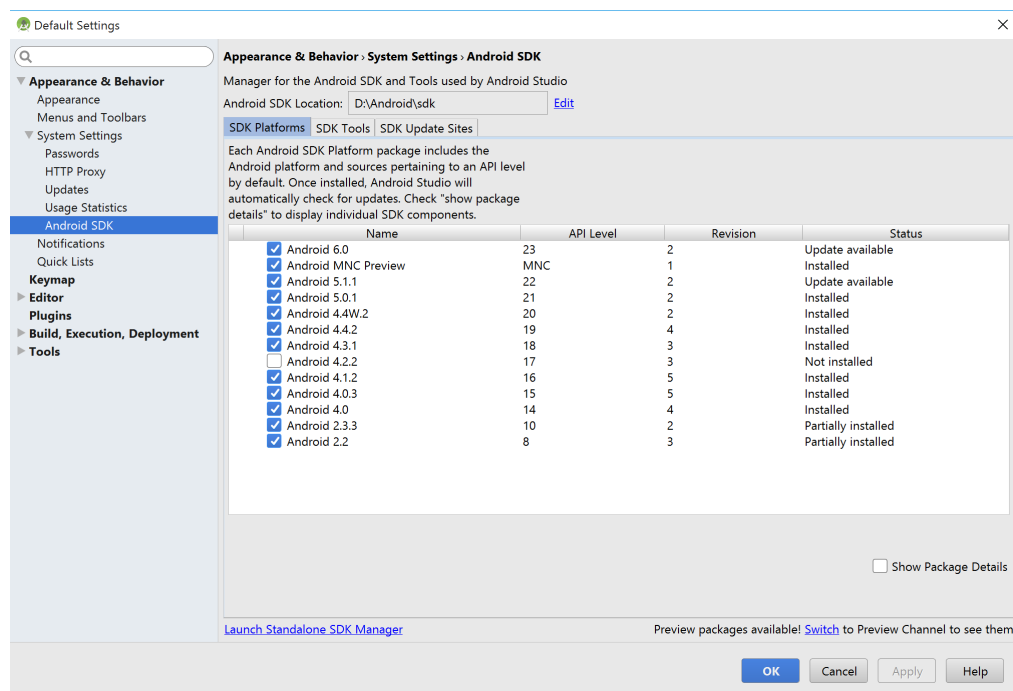


Figura 4.17: Tela de configuração do SDK Manager.

O SDK Manager pode ser acessado em **Configure > SDK Manager** na tela inicial do AS e deve ser semelhante à figura 4.17.

Para finalizar a configuração devemos configurar um emulador para testar nossas aplicações. Para criar um emulador, basta acessar o menu **Tools > Android > AVD Manager** no Android Studio. Caso exista algum emulador já criado, é exibido nesta lista. Para criar um novo, devemos clicar em **Create Virtual Device**. Ao clicar nesta opção, uma lista de possíveis configurações de aparelhos é mostrada, conforme a figura 4.18. Podemos escolher um destes perfis ou, se preferir, é possível criar uma nova configuração. Podemos escolher um perfil desta lista, uma vez que são configurações reais de aparelhos. Clicamos em **Next** para confirmar.

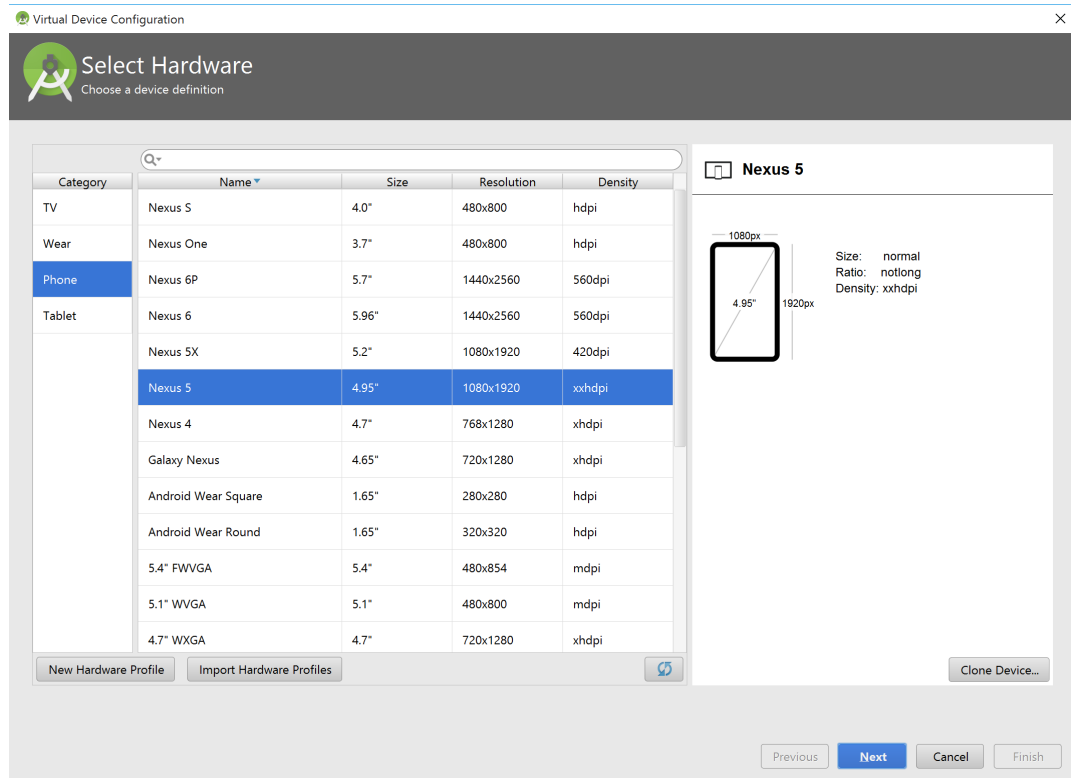


Figura 4.18: Configurações disponíveis de aparelhos Android.

Após escolher o tipo de dispositivo, devemos indicar qual versão do Android iremos usar na criação do emulador. É importante ressaltar que as imagens do sistema que irão aparecer neste estágio são as mesmas que foram baixadas no SDK Manager. A figura 4.19 exhibe as versões disponíveis para escolha. Uma mesma versão pode aparecer mais de uma vez, porém configurada para executar em um processador diferente. Basta escolhermos uma opção e clicar em **Next**. Clique em **Finish** na próxima tela para confirmar. Nosso ambiente está completo.

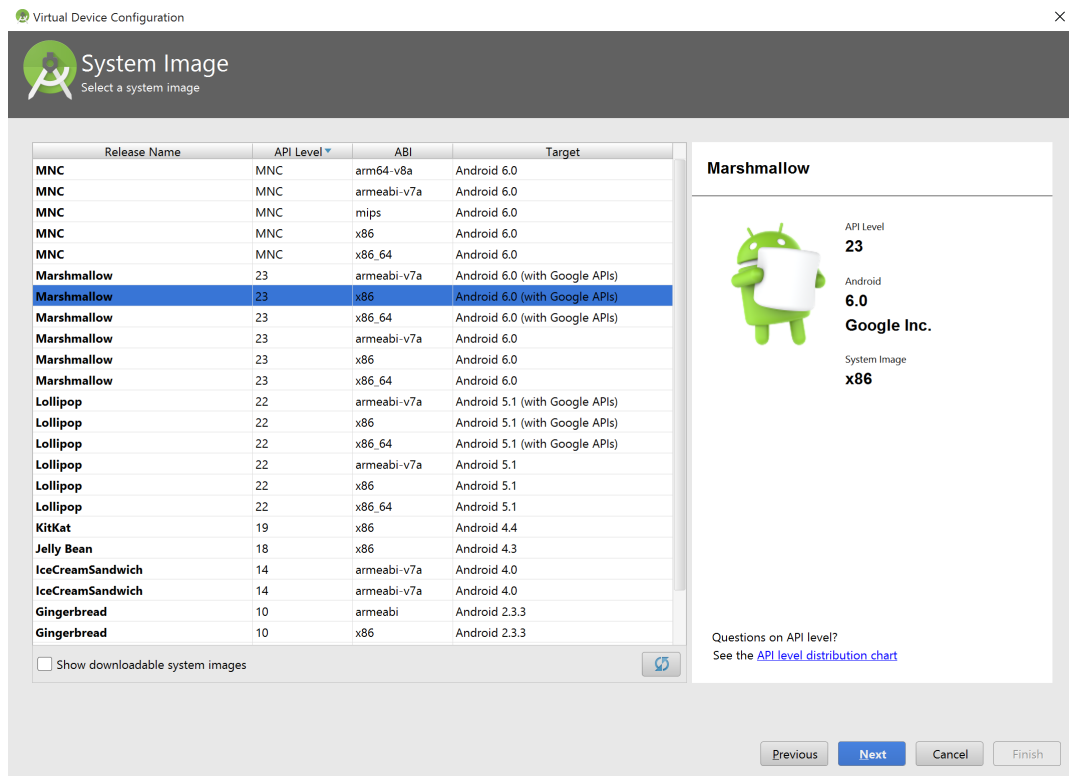


Figura 4.19: Imagens do sistema Android disponíveis para uso.

4.2.2 Processo de Desenvolvimento

O processo unificado de desenvolvimento de software (seção 3.1) foi utilizado neste projeto, dividindo-o em pequenas iterações que, a cada implementação gerou um subproduto deste trabalho. Ao todo, 13 iterações foram executadas no desenvolvimento deste projeto. O apêndice C lista todas as atividades executadas em cada etapa.

4.2.3 Arquitetura do Projeto

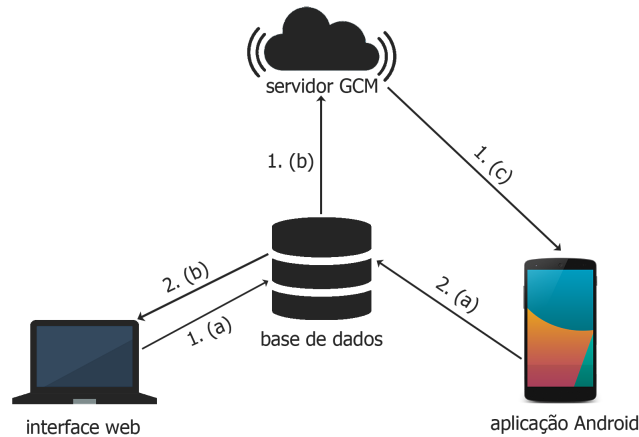


Figura 4.20: Arquitetura do projeto.

A figura 4.20 ilustra a arquitetura de funcionamento da UFMS Mobile. Como pode ser observado, o fluxo 1 (a) armazena os dados da aplicação no banco de dados remoto através da aplicação web. Em 1 (b) a própria aplicação web avisa o servidor GCM da existência de atualizações no servidor após inserir informação no banco de dados. Feito isso, em 1 (c) o servidor GCM notifica o aplicativo Android sobre atualizações disponíveis no servidor. O fluxo contrário é menos comum, porém possível. Por exemplo, ao avaliar uma disciplina através da aplicação Android, esta avaliação é registrada na base de dados remota 2 (a) e refletida no sistema web 2 (b).

As requisições são enviadas para o servidor com o auxílio da biblioteca volley (seção 3.17.4) e é retornado uma lista de objetos correspondentes à requisição. Esta lista retornada é inserida no banco de dados local, evitando fazer conexões desnecessárias e disponibilizando os dados da aplicação mesmo em modo offline.

4.2.4 Interface Gráfica

Embora seja possível criar a interface gráfica diretamente arrastando e soltando os componentes na tela, neste projeto optamos por desenvolver o código manualmente para facilitar o controle de erros gerados o corrigi-los facilmente. A prévia do layout sendo implementado também foi utilizada para melhor experiência de desenvolvimento.

A interface gráfica no Android pode ser desenvolvida tanto através de editor xml, quanto via programação Java. Embora este segundo método seja possível, é altamente recomendado que seja feito através de arquivos xml de layout, uma vez que são mais facilmente editáveis e personalizados, além de permitir que a lógica da aplicação seja desacoplada da interface gráfica.

Neste projeto, o `LinearLayout` é o gerenciador de layout mais comum, pois permite muita flexibilidade ao desenvolvedor, que pode definir manualmente onde cada componente gráfico vai se posicionar. É importante mencionar também que é possível adicionar layouts aninhados. Com isso, podemos adicionar elementos na vertical e na horizontal na mesma tela.

Outro gerenciador de layout utilizado neste projeto é o `FrameLayout`, pois permite adicionar views sobre as outras. Esta característica é muito útil quando queremos mostrar algo na tela que só aparecerá mediante uma condição. Por exemplo, no caso da lista de eventos estar vazia – sem nenhum evento disponível no momento – a `recyclerview` é ocultada e um `textview` é exibido com uma mensagem informando a ausência de itens.

As telas da aplicação foram desenvolvidas utilizando o editor de layout do Android Studio que possui diversas ferramentas que auxiliam na codificação, como:

- **Visualização** – É possível editar o código xml do layout através de uma interface gráfica ou diretamente no código;
- **Prévia** – Se a implementação for feita pelo código diretamente, podemos ver a prévia das alterações em tempo de execução;
- **Dispositivos** – Podemos alterar o aparelho que está sendo mostrado a prévia da nossa tela, possibilitando prever como ficará o layout em um dispositivo diferente;
- **Checagem de erros** – O editor possui um poderoso framework de análise e detecção de erros.

4.2.5 Dispositivos Utilizados

A aplicação para Android foi executada em diversos dispositivos para garantir o seu correto funcionamento. Emuladores DVM foram criados para facilitar o processo de desenvolvimento, uma vez que são integrados com a IDE. Outro benefício notado com a utilização do Android Studio foi a possibilidade de executar diversas instâncias da mesma aplicação em diversos dispositivos ao mesmo tempo. Por exemplo, executar a aplicação em emuladores e dispositivo real ao mesmo tempo.

A lista de dispositivos e suas respectivas configurações é exibida na tabela 4.1. Note que para executar a aplicação em um dispositivo real, é preciso instalar o driver `usb`⁴, ativar **USB debugging** em opções de desenvolvedor no aparelho e conectá-lo ao computador através da porta `usb`.

⁴Disponível via **Android SDK > Extras > Google USB Driver**.

Tabela 4.1: Dispositivos usados no desenvolvimento.

Nome	Resolução	API	Versão	Espaço em Disco	Tipo
Nexus 4	768x1280	19	Android 4.4.2	566MB	Emulador
Nexus 5	1080x1920	23	Android 6.0	1GB	Emulador
Nexus 6	1440x2560	23	Android 6.0	1GB	Emulador
Nexus 5	1080x1920	23	Android 6.0.1	N/A	Dispositivo Real
Nexus 9	1536x2048	23	Android 6.0	1GB	Emulador (tablet)

4.2.6 Permissões

Para o correto funcionamento desta aplicação, algumas permissões precisam ser adicionadas, como mostrado na tabela 4.2:

Tabela 4.2: Lista das permissões utilizadas.

Nome da permissão	Descrição da permissão
INTERNET	Permissão necessária para o aplicativo acessar à Internet.
ACCESS_NETWORK_STATE	Permissão necessária para ler o estado da rede, como: tipo de conexão (WIFI ou 3G) ou se está conectado.
WAKE_LOCK	Permissão para “despertar” aparelho se estiver bloqueado.
RECEIVE_BOOT_COMPLETED	Utilizada para informar que o aparelho foi iniciado com sucesso. Utilizada para receber notificações pendentes ao ligar o dispositivo.
READ_EXTERNAL_STORAGE	Acesso de leitura ao armazenamento externo (SD Card).
WRITE_EXTERNAL_STORAGE	Acesso de escrita ao armazenamento externo (SD Card).

Estas permissões listadas acima são requisitadas em tempo de execução da aplicação. Por exemplo, a permissão de ler e escrever no armazenamento externo só será requisitada quando o usuário fizer o download de arquivos e salvar no sistema de arquivos.

4.2.7 Preparação da Aplicação Cliente

Para configurar o aplicativo para receber mensagens do GCM, é preciso adicionar a dependência no arquivo gradle: `compile 'com.google.android.gms:play-services-gcm:8.1.0'`.

Esta dependência será utilizada para poder utilizar a API do Google Cloud Messaging do Android para integrar as aplicações através de um servidor na web. Como mostrado na tabela acima, basta adicionar a dependência que o gradle é encarregado de fazer a sincronia das bibliotecas necessárias com o código da aplicação.

Além da dependência mencionada, é preciso indicar as permissões necessárias para acessar os recursos do aparelho. Neste caso, iremos adicionar a permissão `C2D_MESSAGE`⁵ no arquivo de manifesto do Android, conforme mostrado abaixo:

Algoritmo 1: Trecho de código do arquivo de configuração `AndroidManifest.xml`

```
<uses-permission
  android:name="ufms.br.com.ufmsapp.permission.C2D_MESSAGE" />
<permission android:name="ufms.br.com.ufmsapp.permission.C2D_MESSAGE"
  android:protectionLevel="signature" />
<uses-permission
  android:name="ufms.br.com.ufmsapp.permission.C2D_MESSAGE" />
```

Feito isso, nossa aplicação está pronta para se comunicar com o GCM. Ao receber uma mensagem do servidor, podemos escolher a forma como desejamos informar o usuário sobre o recebimento da mesma. A maneira mais comum – e a que iremos utilizar – é através de o disparo de uma mensagem de notificação.

4.2.8 Dependências do projeto

As bibliotecas utilizadas no desenvolvimento da aplicação Android precisam ser adicionadas ao projeto através de suas dependências, para que possam ser referenciadas e utilizadas via código. A tabela 4.3 ilustra as dependências das bibliotecas que foram utilizadas neste trabalho.

⁵Anteriormente a API Google Cloud Messaging era chamada de **Cloud To Device Messaging**. Por este motivo a permissão é indicada como `C2D_MESSAGE`.

Tabela 4.3: Lista de dependências das bibliotecas utilizadas.

Nome da biblioteca	Dependência
RecyclerView	<code>compile 'com.android.support:recyclerview-v7:23.1.1'</code>
DSpec	<code>compile 'org.lucasr.dspec:dspec:0.1.1'</code>
CardView	<code>compile 'com.android.support:cardview-v7:23.1.1'</code>
Volley	<code>compile 'com.mcxiaoke.volley:library:1.0.19'</code>
Picasso	<code>compile 'com.squareup.picasso:picasso:2.5.2'</code>
Material Calendar View	<code>compile 'com.prolificinteractive:material-calendarview:1.1.0'</code>
Google Cloud Messaging	<code>compile 'com.google.android.gms:play-services-gcm:8.3.0'</code>
WilliamChart	<code>compile 'com.diogobernardino:williamchart:2.2'</code>
TextDrawable	<code>compile 'com.amulyakhare:com.amulyakhare.-textdrawable:1.0.1'</code>

Logomarca da Aplicação

A ícone do aplicativo para Android deve ser parametrizado utilizando as métricas de alocação dinâmica de recursos para imagem abordadas na seção 3.21.1. Como foi mencionado, a mesma logomarca deve ser redimensionada para diferentes resoluções e alocadas em seus respectivos diretórios no projeto do Android Studio. Feito isso, o Android se encarrega de fazer a escolha do melhor recurso possível para o dispositivo que está rodando a aplicação no momento. As diferentes versões da logomarca seguindo as métricas de design do Google foram criadas utilizando a ferramenta **Android Asset Studio**⁶ e Adobe Photoshop (seção D.6).

O ícone da aplicação com a logomarca em múltiplas resoluções é mostrado na figura 4.21:



Figura 4.21: Logomarca da aplicação Android em diferentes resoluções.

⁶Ferramenta disponível no endereço: <https://goo.gl/Y3pXQ0>.

4.2.9 Aplicação Web

A aplicação web segue os padrões do material design (seção 3.15) para desenvolvimento web. Esta representação de design para websites foi alcançada com auxílio das bibliotecas *materialize* (seção 3.22.2) e *material design lite* (seção 3.22.1). O esquema de cores também foi associado com o aplicativo Android, tornando as aplicações com visual semelhante, embora sejam diferentes tecnologias, aplicando assim os conceitos do material design.

A aplicação web se conecta diretamente com a base de dados do servidor que, por sua vez, se comunica com o servidor GCM, que envia mensagem para os aparelhos Android, como foi mencionado anteriormente. A figura 4.22 mostra a página inicial do projeto web⁷. Para aprimorar a experiência de desenvolvimento, um serviço de hospedagem e domínio foram adquiridos.

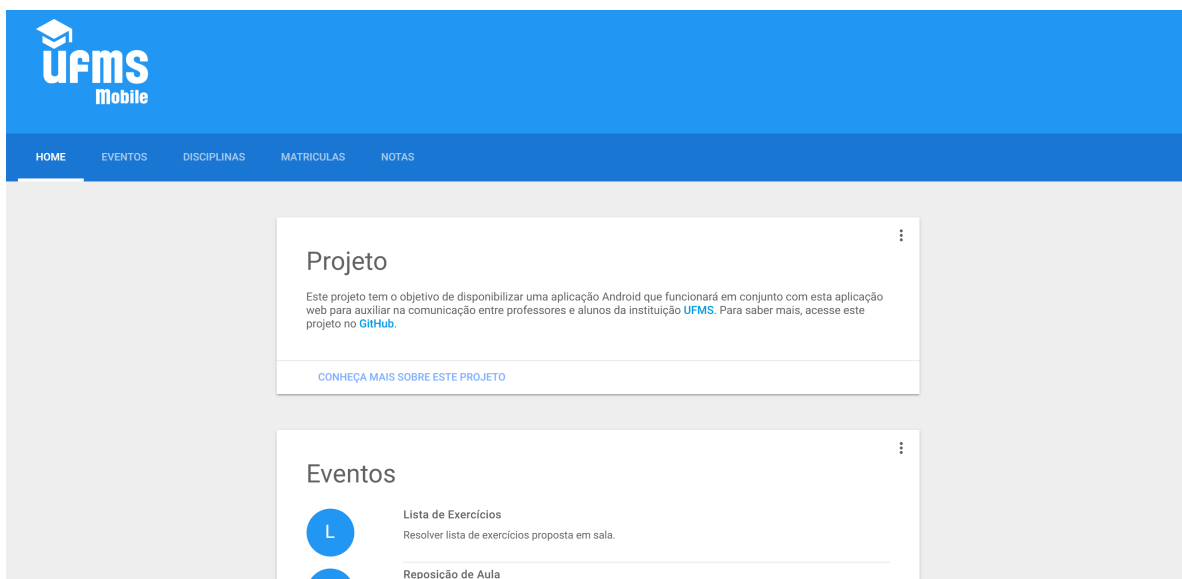


Figura 4.22: Página inicial da aplicação web.

Através da aplicação web temos as mesmas categorias disponíveis no aplicativo: eventos, disciplinas, matriculas e notas. A figura 4.23 mostra os eventos cadastrados no servidor.

⁷Disponível no endereço: <http://henriqueweb.com.br/ufms>.

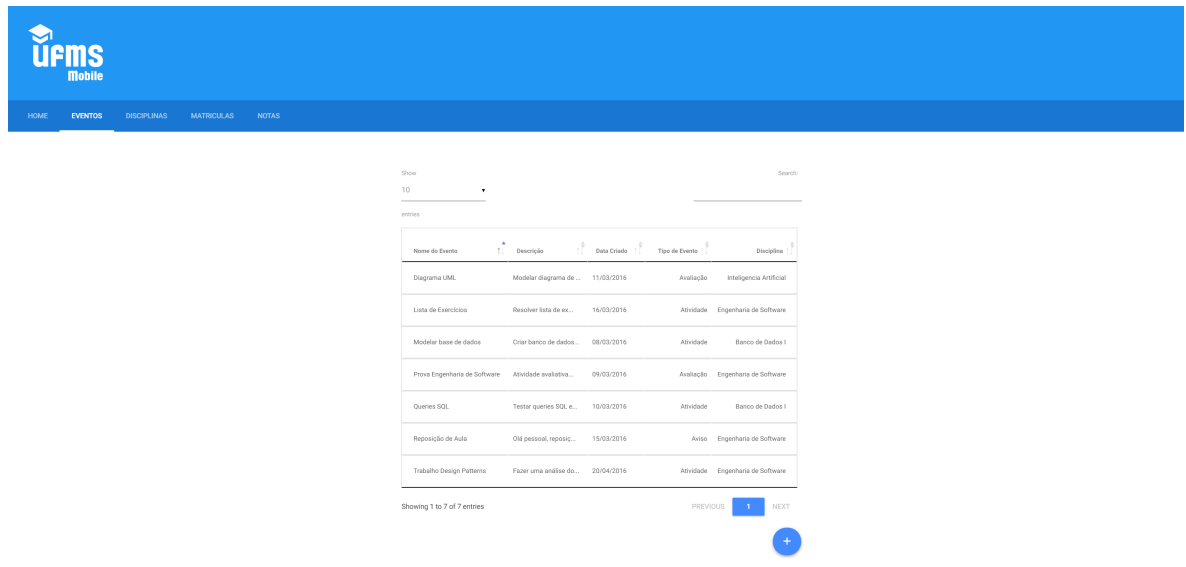


Figura 4.23: Lista de eventos disponíveis na aplicação web.

Embora seja parte essencial deste projeto, a aplicação web precisa de ajustes para funcionar em completa harmonia com o aplicativo Android. Por este motivo, futuras melhorias e expansões estão previstas.

4.2.10 Iterações de design

As iterações de design demonstram a evolução da interface gráfica da aplicação até chegar ao design mais recente. Os protótipos foram o ponto de partida destas iterações. A princípio, as telas foram desenhadas em alto nível de forma manual em papel. Partindo destas imagens desenhadas em alto nível, as telas foram desenhadas com mais detalhes, com suas *labels*, botões, campos de texto, etc. Este arranjo mais detalhado das interface gráfica da aplicação posteriormente foi desenhada em software específico para criação de protótipos de telas e, em seguida, desenvolvidas no projeto.

Este processo é iterativo, ou seja, a interface é melhorada nas iterações futuras, conforme necessário. No apêndice E são apresentadas algumas iterações de design executadas neste trabalho.

4.3 Funcionalidades da aplicação

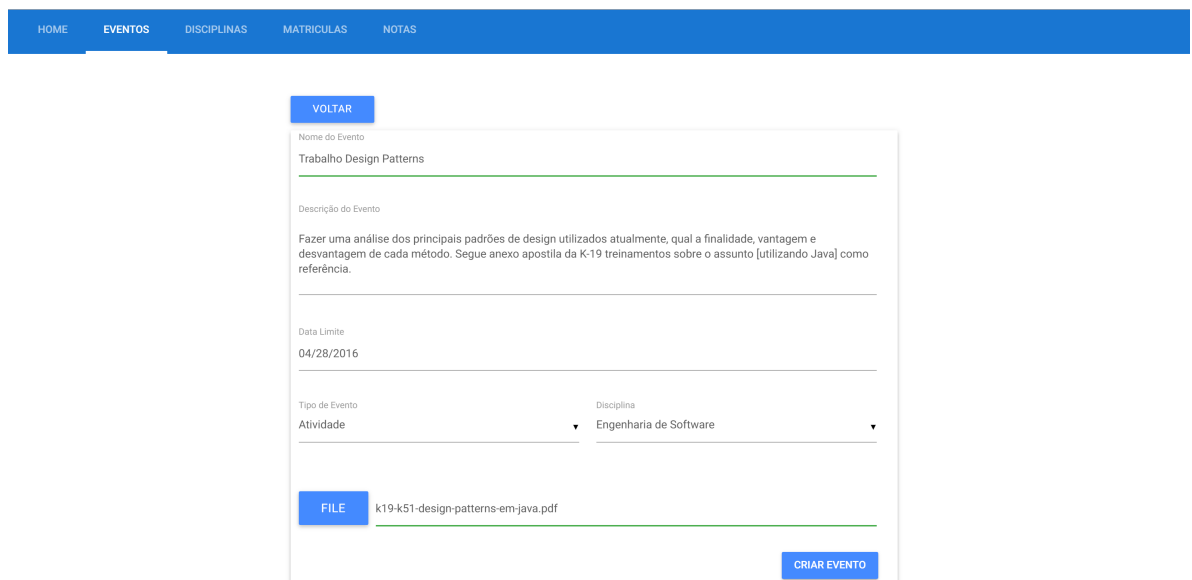
Nesta seção serão abordadas as principais funcionalidades deste projeto, a comunicação entre aplicativo Android e aplicação web, download de arquivos, recebendo e enviando dados para o servidor, visualizando listas, configurando aplicativo, concedendo permissões, entre outras.

4.3.1 Inserindo um evento na aplicação

Para disponibilizar um evento para os alunos, devemos adicioná-lo através da aplicação web – que é a gerenciadora de conteúdo. Ao adicionar eventos no website, uma mensagem de aviso via internet é disparada para todos os dispositivos Android dos usuários que têm acesso aos eventos de uma disciplina específica. Estas mensagens de rede alcançam os aparelhos Android utilizando serviço GCM (seção 3.19) e são convertidas em notificações (seção 3.9) no celular, alertando sobre a disponibilização. Este processo acontece de forma transparente para o usuário e garante que o mesmo tenha acesso às atualizações mais recentes de conteúdo.

Cadastrando evento na aplicação web

Para adicionar um evento no aplicação web, deve-se clicar no botão “+” (adicionar evento) no menu **eventos**. O formulário para adicionar um evento será carregado na mesma página, evitando carregamentos desnecessários. Para completar a inserção, é preciso preencher os campos com as informações do evento, conforme mostrado na figura 4.24.



A imagem mostra a interface de usuário para criar um novo evento. No topo, há uma barra azul com o menu: HOME, EVENTOS, DISCIPLINAS, MATRICULAS, NOTAS. Abaixo, um formulário branco com o seguinte conteúdo:

- Botão "VOLTAR" no canto superior esquerdo.
- Campo "Nome do Evento" com o texto "Trabalho Design Patterns".
- Campo "Descrição do Evento" com o texto: "Fazer uma análise dos principais padrões de design utilizados atualmente, qual a finalidade, vantagem e desvantagem de cada método. Segue anexo apostila da K-19 treinamentos sobre o assunto [utilizando Java] como referência.".
- Campo "Data Limite" com o texto "04/28/2016".
- Dois menus suspenso: "Tipo de Evento" com o valor "Atividade" e "Disciplina" com o valor "Engenharia de Software".
- Campo de upload de arquivos com o botão "FILE" e o nome do arquivo "k19-k51-design-patterns-em-java.pdf".
- Botão "CRIAR EVENTO" no canto inferior direito.

Figura 4.24: Página para adicionar eventos na aplicação web.

Neste formulário, é necessário informar:

- **Nome do evento** – Título a ser adicionado na criação do evento;
- **Descrição** – Texto detalhando o evento em questão;
- **Data limite** – Último prazo para finalizar atividade (se houver este prazo);
- **Tipo de evento** – Categoria em que se encaixa o evento (Atividade, Avaliação, Aviso, Geral);

- **Disciplina** – Disciplina a qual o evento está associado;
- **Anexos** – Documentos que podem ser disponibilizados (se houver).

Disparando mensagens com o servidor GCM

Ao completar o preenchimento dos campos necessários para adicionar um evento, é preciso clicar no botão “Criar Evento”. Se o cadastro for validado corretamente, os dados são inseridos no banco de dados e uma mensagem é enviada para o servidor GCM. Ao receber esta mensagem, o servidor é responsável por avisar os dispositivos Android a respeito da atualização. Todos os aparelhos registrados no servidor GCM são candidatos a receberem esta mensagem, porém, somente os alunos que estão cadastrados na disciplina em questão serão notificados.

Recebendo notificação no aplicativo Android

Quando a aplicação recebe mensagens do servidor GCM, algumas ações podem ser tomadas. Uma delas é notificar o usuário de que uma atualização está disponível no servidor através do *framework* nativo de notificações do Android. A figura 4.25 representa a aplicação Android disparando a notificação para o usuário de um novo evento disponível. O aplicativo Android não precisa estar executando no momento em que a mensagem é recebida do servidor, já que a mesma é avisada pelo Android sobre este recebimento.

A notificação padrão para um evento adicionado tem o formato: título do evento, descrição, hora (que a notificação foi disparada), disciplina associada e ícone da aplicação adaptado para ser exibido em notificações⁸.

⁸A partir da versão 5.0 do Android, os ícones exibidos nas notificações devem ser adaptados para mostrarem apenas a cor branca.

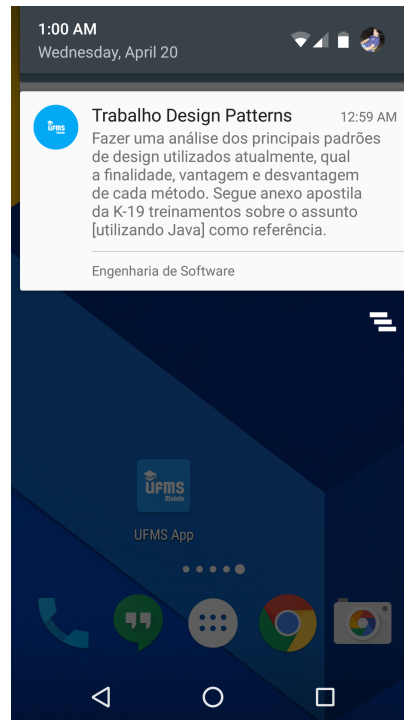


Figura 4.25: Notificação recebida pelo aplicativo Android.

Verificando atualização

Clicando na notificação recebida da aplicação, a tela de detalhes do evento recentemente adicionado é diretamente aberta para o usuário e seus detalhes são mostrados. A figura 4.26 representa a aba “Evento” com as informações sobre a atualização recebida.



Figura 4.26: Tela com detalhes do evento recém criado.

Fazendo download de anexo

A aba “Anexos” exibe os arquivos disponíveis (se houver) como anexo. Neste evento criado, foi adicionado um arquivo PDF para ser usado como material de apoio. Arquivos com extensões **.pdf**, **.doc**, **.xls**, **.ppt** são suportadas. A figura 4.27 ilustra o documento de anexo disponível com o evento adicionado.

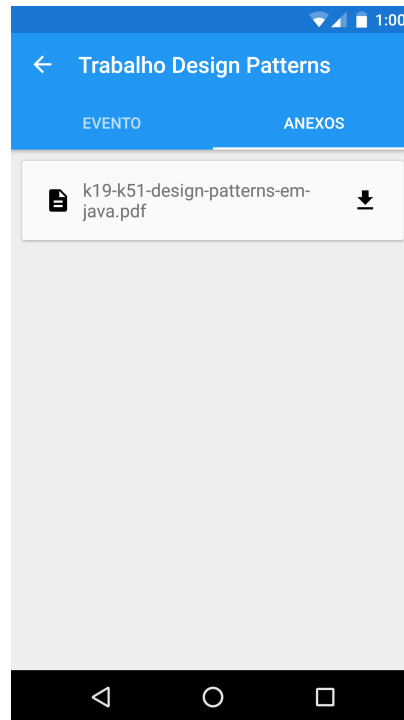


Figura 4.27: Tela com documento de anexo do evento recém criado.

Concedendo permissão

A partir do Android 6.0 (Marshmallow, 3.18.4) o recurso de conceder permissões em tempo real foi adicionado, evitando que o usuário deixe passar permissões suspeitas despercebidas no processo de instalação. Ao baixar um arquivo, o mesmo é salvo no sistema de arquivos do Android e, para ter acesso ao mesmo, é preciso que o usuário conceda esta permissão. Por este motivo, antes de baixar e salvar o anexo, deve-se perguntar ao usuário se ele aceita que a aplicação tenha acesso a tal recurso. A figura 4.28 representa esta requisição. É interessante ressaltar que, se o usuário recusar determinada permissão, o aplicativo de forma alguma terá acesso a um recurso, porém o aplicativo continua executando.

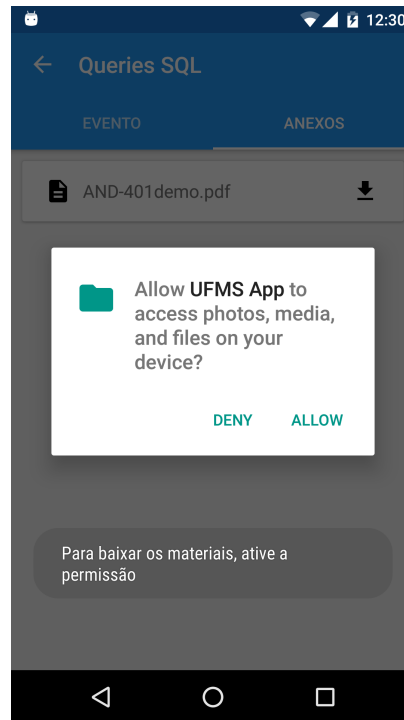


Figura 4.28: Permissão para gravar no armazenamento externo do telefone.

Baixando arquivo

Uma vez concedida a permissão, o arquivo pode finalmente ser baixado. Uma notificação é lançada avisando sobre o progresso do download, promovendo feedback para o usuário a respeito de uma operação que está em execução. A figura 4.29 ilustra este comportamento.

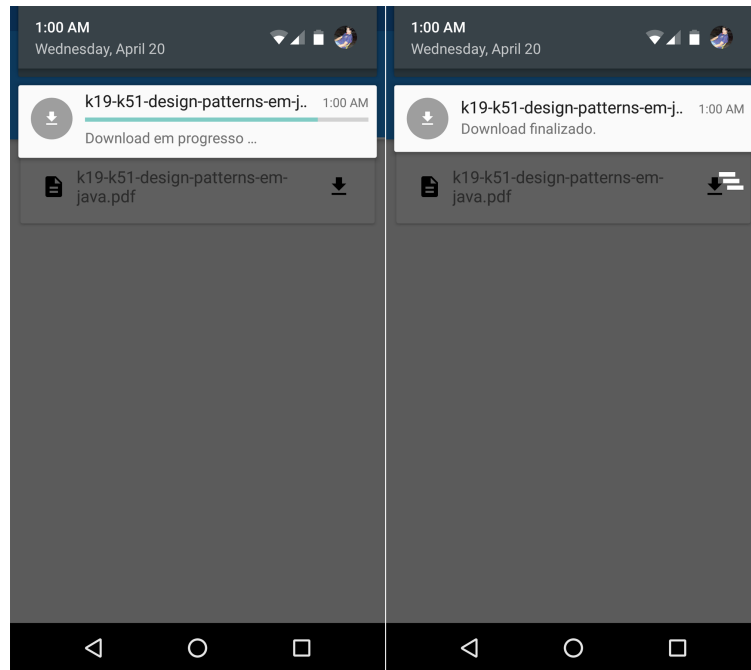


Figura 4.29: Fazendo download do anexo.

Após a conclusão de download do anexo, a barra de progresso na notificação é omitida, dando lugar a uma mensagem de êxito na operação. Ao clicar na notificação de “Download finalizado”, o documento baixado é aberto.

Abrindo arquivo baixado

Como mencionado no tópico anterior, o anexo baixado pode ser aberto ao clicar na notificação. Quando a mensagem de notificação é clicada para que o documento seja aberto, uma **intent** (seção 3.10.3) é disparada para que uma aplicação que execute a extensão do determinado arquivo seja invocada para que o documento seja visualizado. No exemplo da figura 4.30, o aplicativo de leitura de arquivos PDF é executado para abrir a apostila baixada.



Figura 4.30: Abrindo documento baixado.

Permitir download via dados móveis

Por padrão a opção “Baixar arquivos somente via WI-FI” é ativada nas configurações da aplicação Android, evitando que grande quantidade de dados seja consumida da internet móvel do dispositivo, podendo implicar custos adicionais. Porém, se o usuário desejar baixar anexos via dados móveis, é possível desativar esta opção, assim habilitando o download via 3G, 4G, etc. A figura 4.31 ilustra esta opção. Se o aplicativo estiver configurado para baixar somente via WI-FI e o usuário tentar fazer download via dados móveis, uma mensagem é enviada alertando o usuário sobre a situação e o anexo não é baixado até que a determinada opção esteja habilitada.

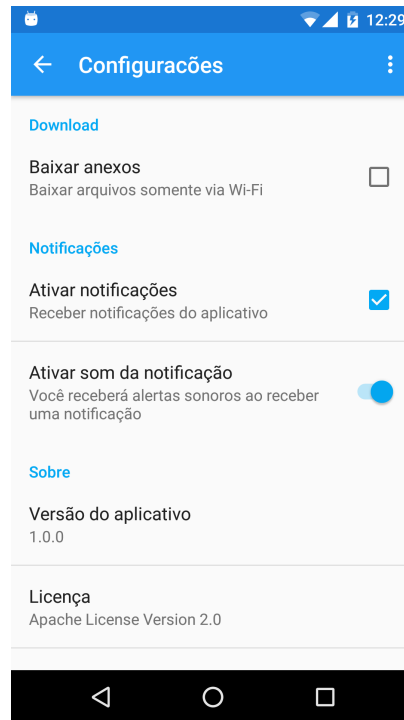


Figura 4.31: Fazendo download de anexo via dados móveis.

4.3.2 Acessando menu principal do aplicativo

O menu principal da aplicação é o ponto de entrada para acessar as categorias do aplicativo Android. Este menu é criado com o componente Navigation Drawer (seção 3.16.1) disponível na plataforma, e permite criar um menu lateral (funciona na esquerda e direita) personalizado para a aplicação, podendo adicionar ícones com a foto do usuário, texto, imagens de fundo e lista de itens com as opções. Nesta seção o menu lateral será estudado em mais detalhes, bem como suas funcionalidades.

Abrindo o menu

Existem duas maneiras de abrir o menu lateral: a primeira é deslizando o dedo no canto da tela da esquerda para a direita (gesto de deslizar) e a segunda é clicando no botão “sanduíche”⁹ na Toolbar (seção 3.15.5). Este botão tem a responsabilidade de abrir o menu caso o usuário não deseje utilizar gestos. A figura 4.32 representa este botão.



Figura 4.32: Botão “sanduíche” da barra de tarefas para abrir o menu principal.

⁹Chamado assim pois sua aparência, com três linhas horizontais, remonta ao visual de um sanduíche.

Selecionando uma opção

Com o menu de navegação aberto, é possível acessar as categorias em que o aplicativo é dividido (Explore, Eventos, Disciplina, Notas, Curso e Configuração). Este menu fica visível a partir de cada categoria destas, permitindo que o usuário navegue livremente entre os módulos durante a interação com a aplicação. A figura 4.33 ilustra o menu principal do aplicativo.

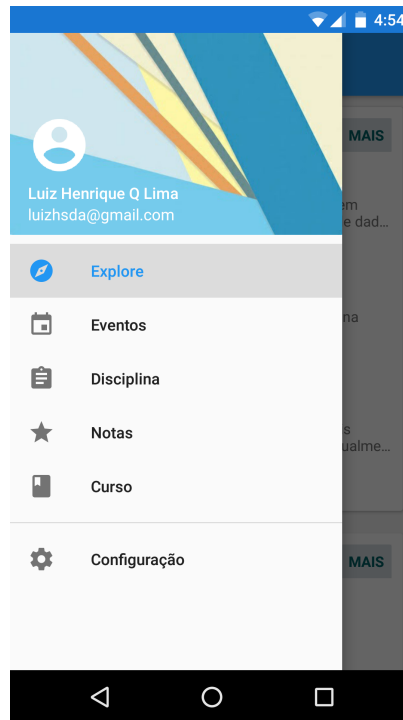


Figura 4.33: Menu principal da aplicação posicionado no lado esquerdo da tela.

4.3.3 Avaliando disciplina cursada

Os usuários têm a possibilidade de avaliar as disciplinas que estão cursando ou que já foram cursadas, proporcionando feedback para os professores responsáveis e também para os futuros alunos. Uma avaliação atribuída a determinada disciplina pode ser posteriormente editada, visto que os alunos podem mudar sua opinião sobre uma disciplina ou pode cursar uma mesma disciplina mais de uma vez. Nesta seção será abordado o mecanismo de avaliações do aplicativo e suas funcionalidades.

Cadastrando avaliação

A opção de cadastrar uma nova avaliação está disponível na tela de detalhes da disciplina, como mostrado na figura 4.34. Para avaliar a disciplina é preciso clicar na nota (estrela de 1 a 5) desejada e depois clicar em “Avaliar” ou simplesmente pressionar o botão diretamente.

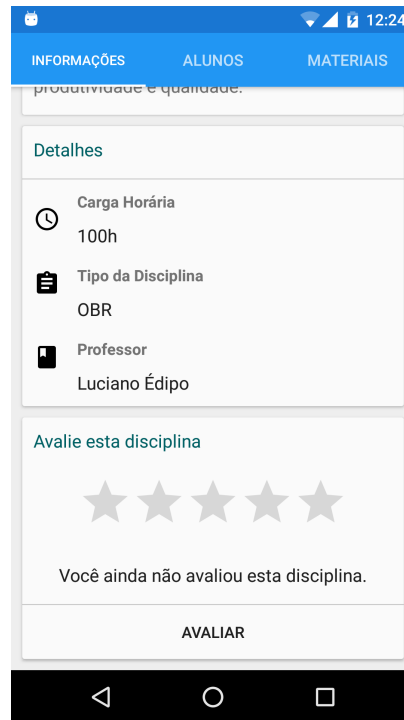


Figura 4.34: Avaliando disciplina.

Ao clicar no botão de avaliar, uma janela de diálogo é exibida com a nota de escolhida e uma categoria (Ruim, Mediano, Bom e Excelente) associada de acordo com um intervalo de notas. Para confirmar, o botão “Avaliar” no diálogo deve ser pressionado, e a avaliação é enviada para o servidor, sendo somada às demais. Se a avaliação for cancelada, a nota da disciplina é zerada. A figura 4.35 representa este comportamento.

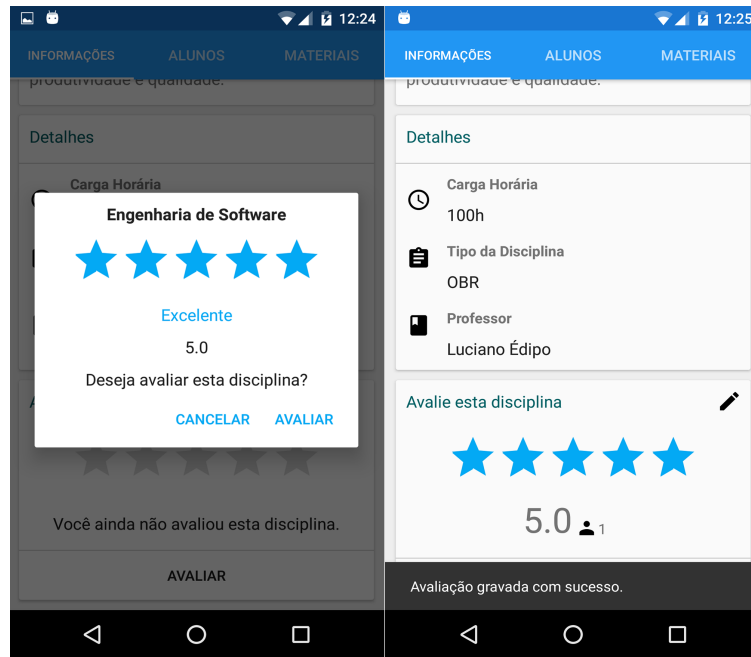


Figura 4.35: Confirmando e enviando nota – 1 a 5 – para servidor.

Editando avaliação

Se o aluno desejar alterar sua avaliação, uma nova nota pode ser atribuída à disciplina. Para isso, o usuário deve clicar no ícone de lápis (editar). Ao pressionar esse botão, a mesma janela de diálogo é aberta, mostrando a nota atual. Ao clicar em “Editar”, a nova nota é enviada ao servidor; já se o usuário cancelar a operação, a antiga nota é mantida. A figura 4.36 ilustra esta funcionalidade.

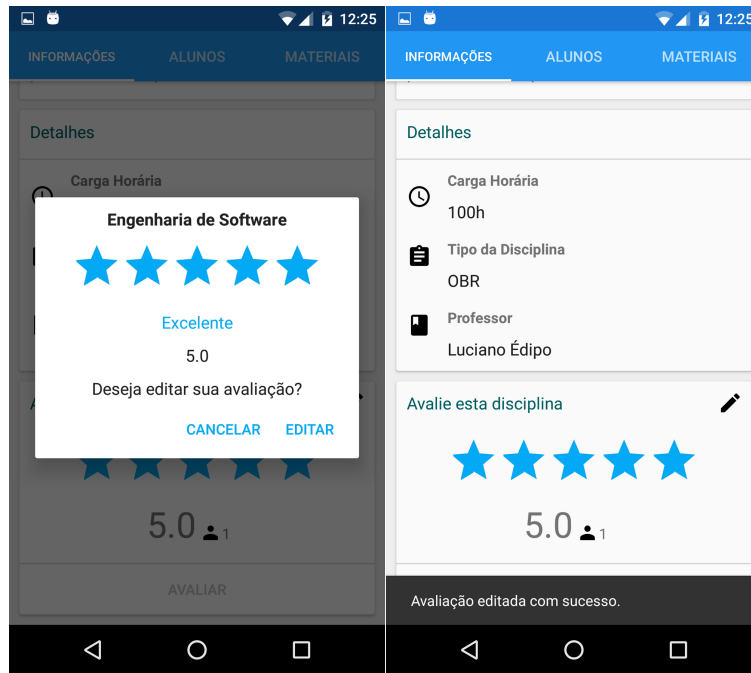


Figura 4.36: Editar avaliação de disciplina realizada anteriormente.

Mostrando média de avaliações da disciplina

Na lista de disciplinas do aluno, a média geral das avaliações feitas por todos os usuários é mostrada antes mesmo de abrir a tela de detalhes, como pode ser observado na figura 4.37. Esta nota é modificada sempre que uma nova nota é atribuída à disciplina. A média das notas é utilizada para filtrar as disciplinas com melhores notas.



Figura 4.37: Média de nota das avaliações realizadas pelos usuários.

4.3.4 Visualizando notas disponíveis

Os professores podem adicionar notas em suas disciplinas, para que os alunos tenham acesso às atualizações. A tela de notas já calcula a média atual do aluno na disciplina, permite visualizar detalhadamente as atividades que foram realizadas e analisar o desempenho através de gráficos comparativos. Esta seção aborda os recursos do aplicativo relacionados às notas do aluno.

Visualização em modo lista

As notas são agrupadas por disciplinas e, se houver ao menos uma nota disponível, a média é mostrada já na tela das disciplinas disponíveis, conforme figura 4.38. Além da média, o tipo de disciplina e o período atual são mostrados.

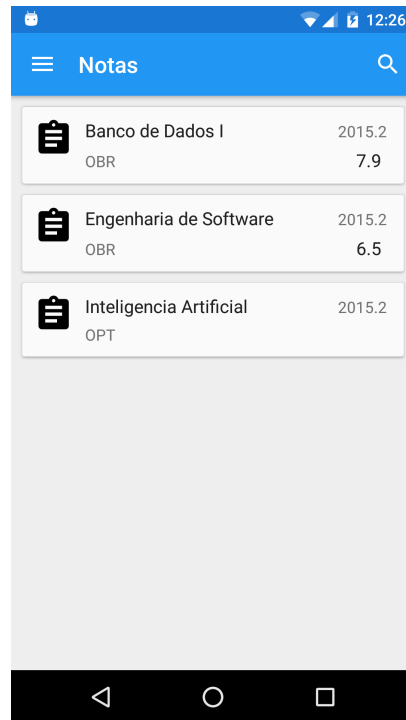


Figura 4.38: Disciplinas em que o aluno está matriculado.

Ao clicar em um disciplina que possui notas postadas, a tela com a lista de atividades disponível é mostrada. Todas as avaliações com notas disponíveis serão mostradas nesta seção. É possível alternar para o modo de desempenho, clicando no menu de opções da tela de listagem, e pressionando “Desempenho View”. Esta opção só está disponível se a disciplina em questão tiver alguma nota; caso contrário é omitida.

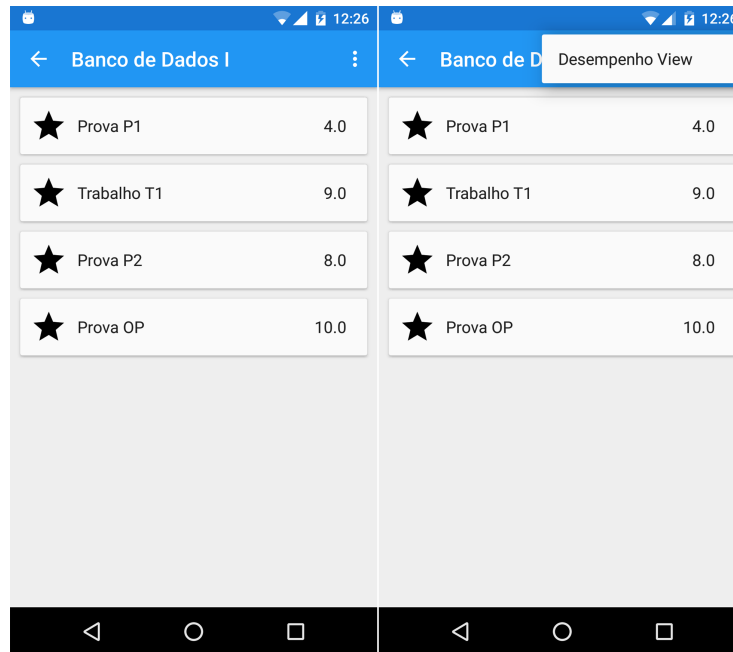


Figura 4.39: Atividades com notas disponíveis em determinada disciplina.

Visualização em modo de análise de desempenho

A visão de desempenho permite analisar as notas através de gráficos e compará-las sob algumas perspectivas. O primeiro gráfico mostra as notas em ordem decrescente, exibindo sempre a maior nota primeiro. O segundo gráfico analisa o desempenho das notas no decorrer do tempo, evidenciando progresso, regresso, ou as notas permaneceram imutáveis. O terceiro gráfico compara as notas individuais dos alunos com a média das notas da turma para a mesma atividade. A figura 4.40 ilustra esta tela de desempenho.

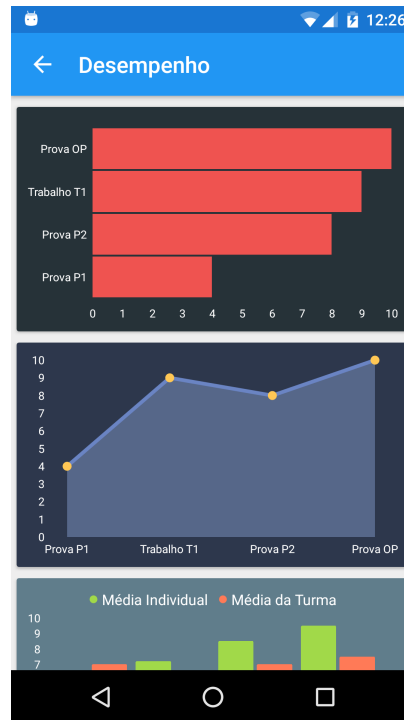


Figura 4.40: Modo de visualização de notas para análise de desempenho com gráficos.

4.3.5 Acessando configurações do aplicativo

As configurações da aplicação Android permitem que as preferências do usuário sejam salvas e o aplicativo se comporte adequadamente, de acordo com o que foi configurado. Nesta seção serão abordadas as preferências do usuário, desde o monitoramento de mudança nas mesmas e aplicando as alterações no aplicativo.

Monitorando mudanças de configuração

Para saber quando uma preferência é alterada, deve-se registrar um *listener*¹⁰ no componente desejado e, quando a preferência em questão é acionada (checada ou ligada/desligada), a aplicação é avisada. Na tela de preferências do aplicativo Android, é possível desativar o recebimento das notificações pela aplicação, bem como seus alertas sonoros. Para funcionar corretamente, é preciso monitorar quando o **checkbox** de “Ativar notificações” é desmarcado para que a preferência de som da notificação seja desligada também, uma vez que não pode haver alerta sonoro para notificações se as mesmas estão desabilitadas. A figura 4.41 ilustra este comportamento.

¹⁰Método que monitora algum acontecimento e avisa às classes interessadas para que seja realizada alguma ação.

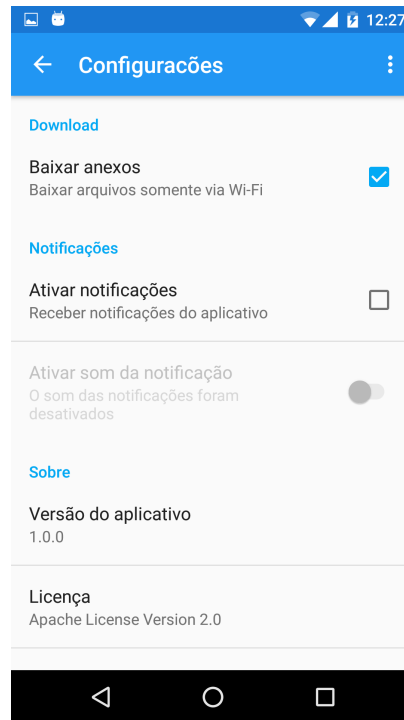


Figura 4.41: Monitorando alterações nas preferências do usuário na tela de configurações.

Aplicando preferências do usuário

As preferências salvas na tela de configurações são armazenadas em arquivos com ajuda do *framework* Shared Preferences (seção 3.12.3) e podem ser recuperadas em qualquer classe da aplicação. Por este motivo, é possível acessar as configurações salvas antes de executar alguma tarefa, aplicando o que foi determinado anteriormente. Na figura 4.42 é definido que a aplicação continue recebendo notificações do servidor, porém sem nenhum tipo de alerta sonoro, silenciosamente. A partir da próxima notificação recebida, este comportamento já será aplicado.

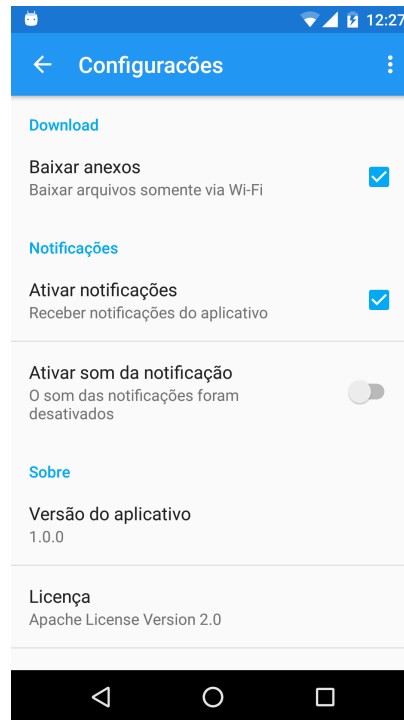


Figura 4.42: Aplicando preferências do usuário no comportamento da aplicação.

4.3.6 Exibindo lista de eventos

Ao fazer uma requisição no servidor pela lista de eventos, um **array** JSON é retornado com os elementos disponíveis no servidor, conforme mostrado no trecho de código 2. Este arquivo JSON retornado é acessado e consumido pela aplicação Android, onde este array é convertido em uma lista de objetos Java, possibilitando que estes eventos sejam inseridos em lote no banco de dados local e exibidos na tela para o usuário.

Após o consumo do JSON e sincronia com o banco de dados local, os eventos disponíveis para o aluno são mostrados na tela de listagem conforme exemplo da figura 4.43, que pode ser acessada a partir da opção “Eventos” no menu lateral. É possível marcar um evento “como não lido”, mudando a cor da linha do item para branco, destacando-se dos demais, permitindo que o usuário os acesse posteriormente com facilidade.

Algoritmo 2: Trecho do array de JSON com os eventos disponíveis retornado pelo servidor, disponível em: <http://www.henriqueweb.com.br/webservice/list/listEventos.php>.

```

{
    "eventos": [
        {
            "app_id_evento": "3",
            "app_nome_evento": "Queries SQL",
            "app_descricao_evento": "Testar queries SQL estudadas em sala. Criar diagrama de banco de dados ex. 02.",
            "app_evento_timestamp": "2016-03-10",
            "app_evento_data_limite": "2015-12-14",
            "app_evento_small_icon": "ws_images/edit.png",
            "app_evento_big_icon": "ws_images/edit2.png",
            "app_tipo_evento_fk": "1",
            "app_disciplina_fk": "1"
        },
        {
            "app_id_evento": "2",
            "app_nome_evento": "Prova Engenharia de Software",
            "app_descricao_evento": "Atividade avaliativa confirmada. Assuntos: Métodos Ágeis SCRUM e Extreme Programming.",
            "app_evento_timestamp": "2016-03-09",
            "app_evento_data_limite": "2015-12-11",
            "app_evento_small_icon": "ws_images/edit.png",
            "app_evento_big_icon": "ws_images/edit2.png",
            "app_tipo_evento_fk": "2",
            "app_disciplina_fk": "2"
        },
        {
            "app_id_evento": "1",
            "app_nome_evento": "Modelar base de dados",
            "app_descricao_evento": "Criar banco de dados proposto na página 56 do livro texto.",
            "app_evento_timestamp": "2016-03-08",
            "app_evento_data_limite": "2015-12-10",
            "app_evento_small_icon": "ws_images/edit.png",
            "app_evento_big_icon": "ws_images/edit2.png",
            "app_tipo_evento_fk": "1",
            "app_disciplina_fk": "1"
        }
    ]
}

```

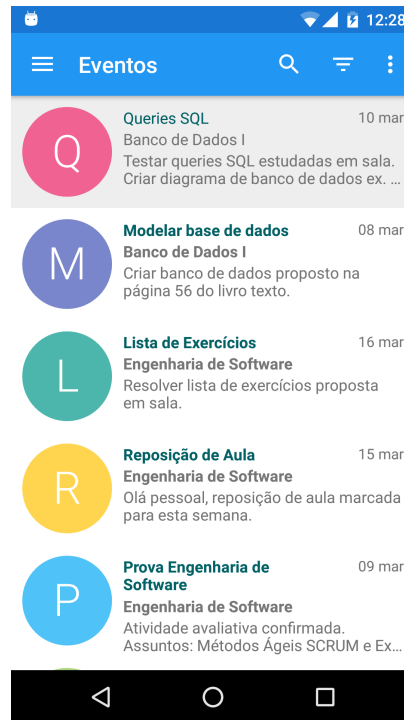


Figura 4.43: Lista de eventos disponíveis na aplicação Android.

Buscando um evento

Na tela de lista de eventos é possível buscar por um evento em específico digitando seu nome no campo de busca e os resultados serão carregados com os registros correspondentes. Esta busca por itens pode ser de grande valia em uma lista muito extensa onde um evento em específico interessa. Os resultados são carregados em tempo real, ou seja, em quanto o usuário está digitando a busca. A figura 4.44 representa esta funcionalidade.

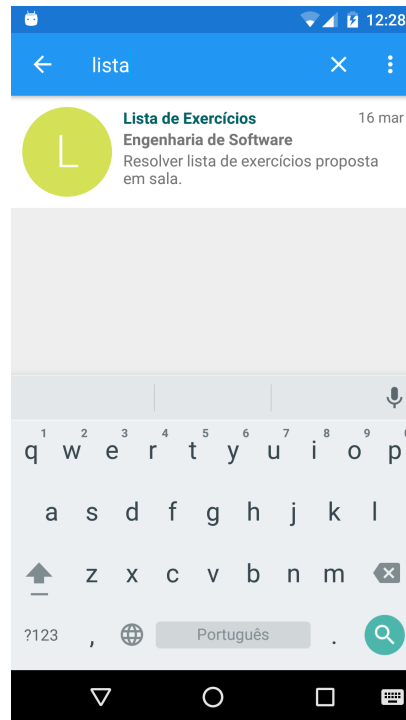


Figura 4.44: Buscando um evento específico na lista de eventos.

Eventos em modo calendário

Por fim, uma característica da tela de eventos é poder alternar entre o modo lista e calendário. O modelo calendário (implementado com ajuda da biblioteca Material Calendar View, seção D.4) permite visualizar os eventos em uma perspectiva mais ampla, promovendo a possibilidade do usuário se planejar a respeito das datas dos mesmos. Para acessar este modo de exibição, basta clicar no menu de opções da tela de listagem e depois pressionar a opção “Ver calendário” no mesmo. A figura 4.45 representa este mecanismo. Para voltar para o modo lista, o mesmo menu deve ser acessado, agora clicando em “Ver lista de eventos”.

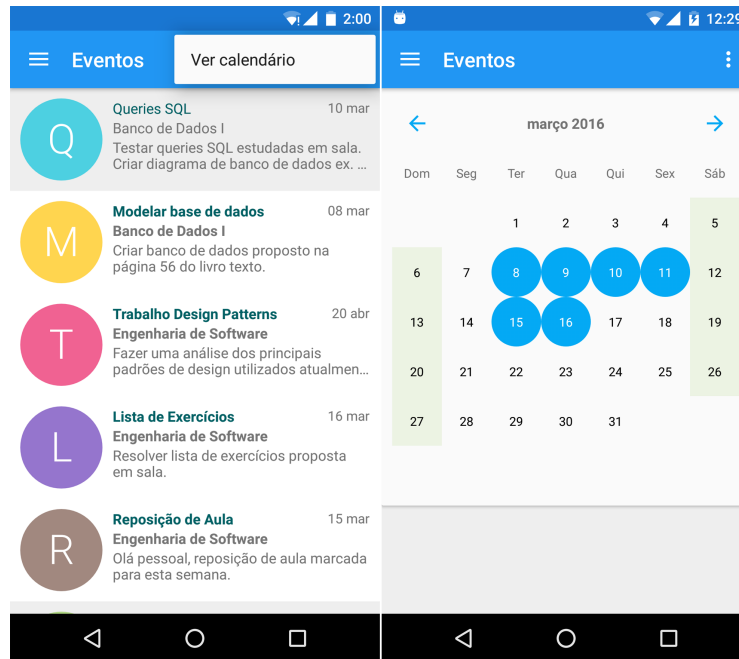


Figura 4.45: Exibindo eventos em modo calendário.

4.3.7 Licença de Utilização

Esta aplicação é código aberto e esta registrada sob licença Apache License 2.0¹¹. A licença da Apache é uma permissão para uso de software livre escrita pela Apache Software Foundation (ASF).

A licença Apache 2.0 foi criada em 2004 e garante liberdade para usar, distribuir, modificar e distribuir versões modificadas do software desde que esteja sob os termos de aceitação da licença.

4.3.8 Repositório de Código

A IDE Android Studio possui integração com VCS (*Version Control System* ou Sistema de Controle de Versionamento)¹², que possibilita registrar mudanças feitas no projeto em arquivos de forma que seja possível acessar as alterações ao longo do tempo. Como esses sistemas são centralizados na web, é possível que mais de um desenvolvedor trabalhe em um projeto específico ao mesmo tempo.

São os Sistemas de Controle de Versão Distribuídos (Distributed Version Control System – DVCS) responsáveis por gerenciar os repositórios onde ficam os códigos que são copiados e disponibilizados para os clientes. Existem vários sistemas de controle, como Git, Mercurial

¹¹Os termos de aceitação de uso desta licença estão disponíveis no endereço: <http://www.apache.org/licenses/>.

¹²O VCS pode ser ativado no menu **VCS > Activate VCS for this project**.

e Bazaar. Neste projeto iremos utilizar o Git em conjunto com o GitHub, que é um serviço web de hospedagem para projetos Git.

Dentre as principais funcionalidades do GitHub, podemos citar:

- **Permitir colaboradores em um projeto** – Adicionar desenvolvedores ou colaboradores em geral a um projeto;
- **Tornar um projeto privado (comercial)** – Em geral os trabalhos publicados no GitHub são públicos para os usuários. Entretanto, a versão comercial do serviço possibilita criar projetos privados, onde somente usuários autorizados têm acesso;
- **Clonar projeto** – Faz uma cópia do projeto para seu ambiente de desenvolvimento;
- **Visualizar estatísticas de desenvolvimento** – Mostra diversos gráficos de desempenho geral ou em um projeto específico.

Este trabalho pode ser encontrado através do endereço: <https://github.com/luizhsda10/UFMSApp>. Através deste repositório é possível verificar informações gerais sobre a aplicação, analisar o histórico de desenvolvimento, ter acesso ao código-fonte completo da aplicação, clonar o repositório, reportar um erro, entre outras funcionalidades.

4.4 Considerações e Validação

Uma aplicação Android deve atender um bom número de usuários e aparelhos para que obtenha o desejado retorno. Alguns aparelhos ainda rodam as versões mais antigas do sistema operacional e, por este motivo, não detêm das funcionalidades mais recentes. O material design foi introduzido no Android 5.0 API Level 21. Porém nossa aplicação suporta aparelhos com o Android a partir da API 19. *Como oferecer suporte a dispositivos antigos sem abrir mão dos recursos mais modernos?* É aí que entra a API de compatibilidade.

4.4.1 Adaptando para tablets

A aplicação Android deve se adaptar às diferentes dimensões de telas que os dispositivos possuem. No caso de um tablet, o aplicativo dispõe de uma tela consideravelmente maior que a da maioria de smartphones e a aplicação deve tirar proveito disso. A figura 4.46 mostra a aplicação sendo executada em um tablet inicialmente sem nenhuma adaptação. O próprio sistema Android organiza os elementos gráficos na tela da forma que considera mais conveniente.

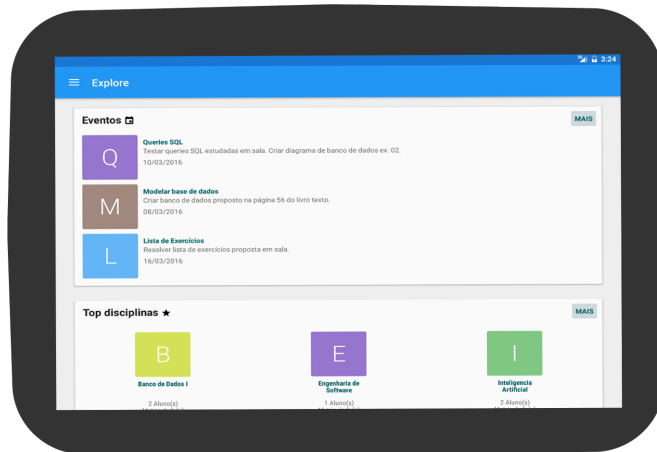


Figura 4.46: Aplicação não adaptada executando em tablet de 9 polegadas.

Embora o próprio Android faça esta adaptação automática da interface gráfica, o ideal é que a aplicação seja totalmente adaptável e interprete os diferentes contextos em que está sendo executada. Com isso, pode-se tirar proveito dos espaços que sobram para melhorar a interação com o usuários e evitar cliques desnecessários.

Neste projeto, ao ser executado em um tablet, o menu lateral é expandido, ocupando melhor o espaço “em branco” na tela e economizando um clique do usuário – que seria necessário para abrir o menu de opções. A figura 4.47 ilustra a aplicação adaptada para este contexto de execução.

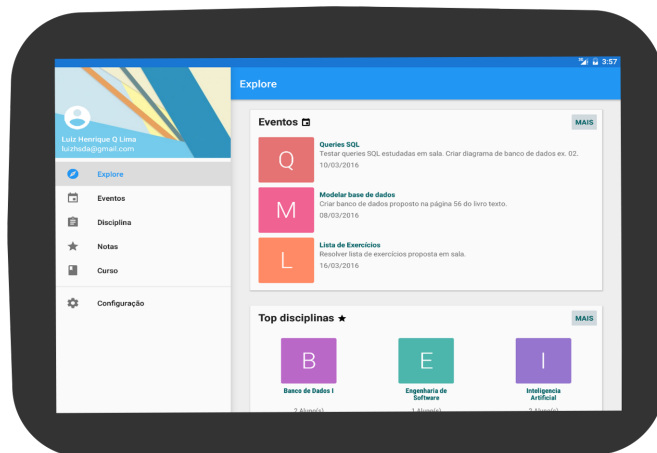


Figura 4.47: Layout da aplicação adaptado para executar em tablet.

4.4.2 Suportando diversas versões do Android

A aplicação Android foi adaptada para executar sem problemas em versões mais antigas. Obviamente alguns recursos não são perfeitamente ajustados, porém o aplicativo se comporta totalmente funcional em qualquer versão suportada. A figura 4.48 ilustra a comparação entre

a tela de eventos disponíveis executando na API 19 (Kitkat) e na 23 (Lollipop), respectivamente. Note que algumas funcionalidades podem não estar disponíveis como, por exemplo, a cor da barra de status¹³ da aplicação rodando no aparelho com o Android Kitkat não é alterada para azul, pois este recurso ainda não foi introduzido na biblioteca de compatibilidade.

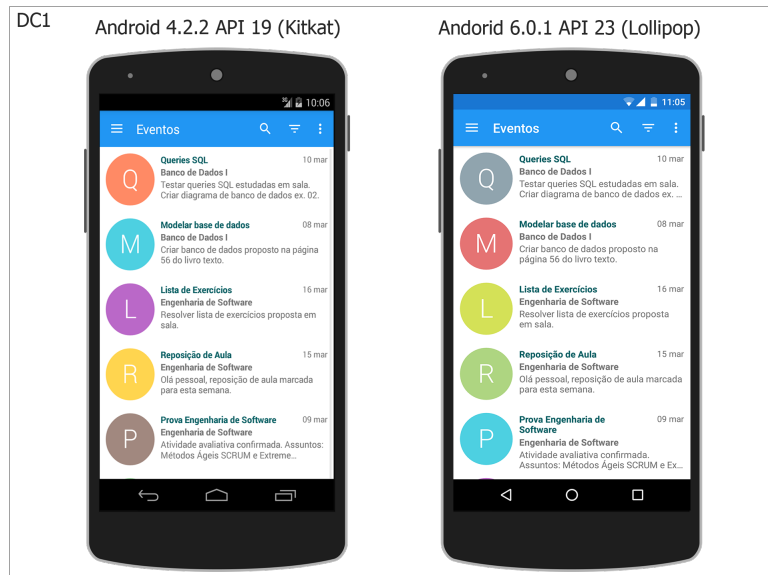


Figura 4.48: Diagrama Comparativo 1 – Lista de eventos API 19 e API 23.

A figura 4.49 compara a utilização da biblioteca para gráficos WilliamChart (seção D.3), utilizada para gerar gráficos de desempenho do estudante. Observa-se que os gráficos funcionam corretamente em ambas as versões. Somente a barra de status apresenta diferença no quesito visual, como mencionado anteriormente.

¹³Barra localizada na parte superior do sistema, onde são informados a hora do sistema, o nível da bateria, a conexão com a internet e notificações.



Figura 4.49: Diagrama Comparativo 2 – Modo desempenho do aluno API 19 e API 23.

A utilização de abas no Android é muito recomendado quando se deseja organizar informações por categoria, facilitando a navegabilidade do usuário. O diagrama comparativo mostrado na figura 4.50 ilustra a tela de detalhes do evento com abas separando dados gerais do evento dos anexos disponíveis.

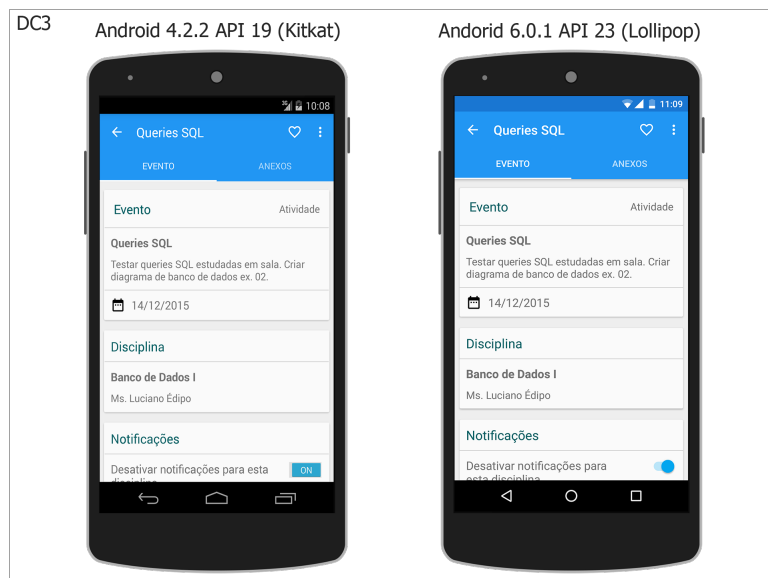


Figura 4.50: Diagrama Comparativo 3 – Detalhes do evento API 19 e API 23.

Desenvolver uma aplicação que se adapte a diferentes dispositivos é uma tarefa desafiadora e uma jornada muito interessante. Por outro lado, usuários deixarão de usar sua

aplicação se ela não funcionar como eles esperam. Dar suporte às versões mais antigas do Android sem deixar o *glamour* de lado é indispensável no desenvolvimento móvel moderno.

4.4.3 Desempenho no acesso à rede

Quando a aplicação Android é executada pela primeira vez, é necessário fazer requisições no servidor para sincronizar o conteúdo presente no banco de dados remoto (centralizado) com o banco de dados local. Ao fazer isso, o aplicativo evita ficar se conectando diversas vezes ao servidor, poupando recursos de rede do dispositivo.

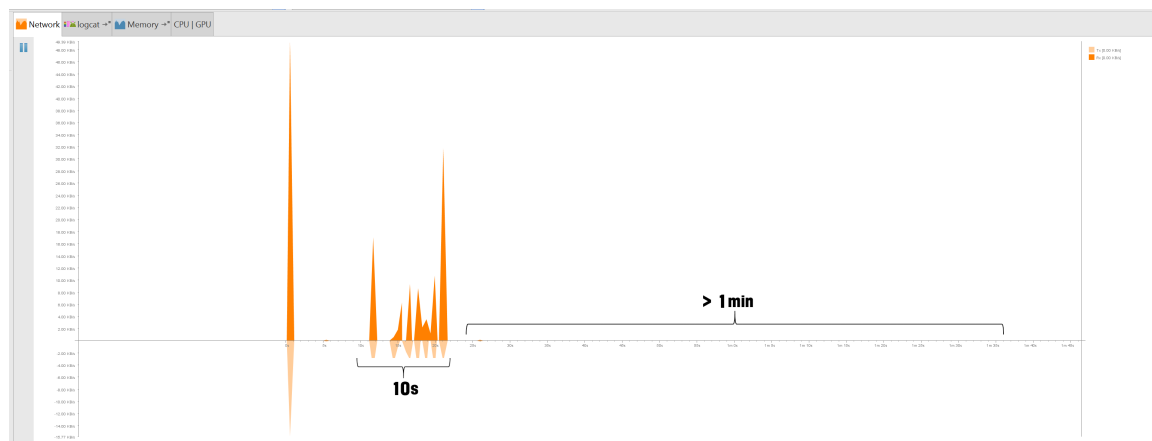


Figura 4.51: Desempenho da aplicação ao fazer sincronia com o servidor através da internet.

A figura 4.51 ilustra o teste de consumo de rede realizado, onde a aplicação permanece executando por cerca de 1 minuto e 30 segundos. Embora a aplicação execute todo esse tempo, nota-se que a sincronia com o servidor é feita nos primeiros 10 segundos do teste, não se conectando mais com o servidor no decorrer do processo. Isto é possível pois os dados retornados do servidor são inseridos no banco de dados local da aplicação (sincronia), evitando que o aplicativo faça futuras requisições para obter os mesmos dados.

4.4.4 Validando design da aplicação

Uma característica marcante do Material Design (seção 3.15) é o design único, ou multi-plataforma, onde a mesma linguagem visual é aplicada em diferentes dispositivos. Neste projeto, representamos as métricas de cores, sombras, navegação e posicionamento do Material Design. Para completar a experiência, devemos padronizar nosso projeto de modo a replicar estes conceitos em outros dispositivos.

Com auxílio das bibliotecas de suporte do Android (seção 3.13.1) podemos criar uma aplicação visualmente padronizada em versões mais recentes até as mais antigas e também adaptamos este projeto para se comportar de forma eficiente em tablets. A estilização Material da aplicação web pode ser atingida com auxílio dos frameworks MDL (seção 3.22.1) e

Materialize (seção 3.22.2). A figura 4.52 representa linguagem visual da aplicação padronizada em diferentes plataformas.



Figura 4.52: Design da aplicação em diferentes plataformas.

No capítulo seguinte serão abordadas as considerações finais acerca deste projeto e os ensinamentos adquiridos. As contribuições que deixamos com este trabalho, suas limitações e futuras melhorias e expansões.

Capítulo 5

Considerações Finais

As instituições de ensino possuem características semelhantes (são compostas por professores, disciplinas, avaliações, notas, etc.), que podem ser alinhados na construção de uma aplicação padronizada, tirando proveito das tecnologias disponíveis. Este trabalho propôs o desenvolvimento e uma aplicação para dispositivos Android integrada com um sistema web gerenciador de conteúdo através de um *web service*. O objetivo destas aplicações é propor a comunicação entre aluno e professor usufruindo da capacidade das tecnologias web e móvel.

Os objetivos inicialmente propostos neste trabalho foram alcançados no decorrer do processo de desenvolvimento do projeto. Porém, para que estes objetivos pudessem ser alcançados, um conjunto de ferramentas, bibliotecas e APIs foi utilizado, conforme descrito no capítulo 3, assim como estudo bibliográfico e levantamento de requisitos, passando pela fase de modelagem e implementação. É possível ainda expandir as funcionalidades do sistema, adicionando recursos e corrigindo possíveis falhas, que podem ser identificadas através da execução da fase de testes. Os prováveis trabalhos futuros serão abordados na seção 5.3.

5.1 Contribuições

Este trabalho é código aberto e pode contribuir com instituições de ensino que não utilizam software para gerenciamento de eventos, disciplinas e notas e comunicação entre aluno e professor. Através do material disponibilizado neste documento e também na web (seção 4.3.8), é possível analisar os artefatos produzidos no desenvolvimento deste trabalho, e ter com base para futuras implementações.

Em primeiro momento a aplicação desenvolvida neste trabalho será executada no ambiente acadêmico da Universidade Federal de Mato Grosso do Sul, porém pode ser utilizada em qualquer instituição de ensino.

5.2 Limitações

Desenvolver uma aplicação Android com melhor experiência do usuário implica tratar os diversos ambientes em que esta aplicação pode ser executada. O Android permite que um aplicativo tenha diferentes comportamentos dependendo do modo em que é executado. Esta aplicação se adapta aos dispositivos cuja versão do sistema é igual ou superior a API 19 (Kitkat).

A plataforma Android é atualizada com muita frequência, o que implica sempre estar buscando novos conhecimentos para atualizar nossa aplicação. Por outro lado, muitas vezes essas mudanças são tão rápidas que até mesmo livros especializados se tornam rapidamente obsoletos. É preciso então estar sempre atento à documentação oficial e no site oficial de desenvolvedores (<https://developers.google.com/>).

A integração entre diferentes aplicações é desafiadora, já que funcionam de formas distintas. Criar uma interface entre uma aplicação Android e uma aplicação web pode se tornar uma tarefa complexa. Entretanto, APIs e bibliotecas disponibilizadas pelo Google e por terceiros facilitam essas tarefas de integração e desenvolvimento destas plataformas.

5.3 Trabalhos Futuros

Algumas melhorias podem ser feitas neste projeto no futuro para atingir um número maior de usuários e garantir melhor experiência de uso. Dentre as modificações, podemos citar:

1. **Transformar aplicações em serviço** – como citado anteriormente, esta aplicação em primeiro momento tem como objetivo o correto funcionamento para a instituição UFMS, porém pode ser adaptada para funcionar como um serviço, disponível para qualquer instituição de ensino;
2. **Expandir funcionalidades** – Algumas funcionalidades podem ser implementadas como, por exemplo: alarmes quando um evento estiver perto de sua data limite, compartilhar eventos com contatos, editar perfil do usuário, fazer download de múltiplos arquivos, etc;
3. **Publicar na Google Play Store** – Publicar aplicativo Android na loja oficial de aplicativos da plataforma;
4. **Aperfeiçoar interface web** – A aplicação web necessita de alguns ajustes no desenvolvimento para melhor integração com a aplicação móvel. Como trabalho futuro, melhorias precisam ser feitas no website, como adicionar disciplinas, notas e alunos;
5. **Testes unitários** Testes unitários irão ser executados utilizando entrada de dados, irão verificar se o sistema está validando corretamente os campos de dados, tamanho das entradas (dos caracteres) e banco de dados. Os testes de banco de dados serão executados a fim de encontrar vulnerabilidade nas consultas;

6. **Suíte de testes** – Os suítes de testes serão responsáveis por executar os testes unitários em lote, minimizando o tempo de duração desta fase e garantindo sua acurácia, já que é possível executar testes similares em paralelo;
7. **Testes funcionais** – Os testes funcionais irão alinhar as funcionalidades da aplicação com os requisitos do sistema. Estes testes terão como objetivo: verificar se o usuário (aluno) consegue acessar as disciplinas em que está matriculado, visualizar suas notas e fazer download de anexos; ainda terá como meta: garantir que o usuário (professor) consegue adicionar registros (evento, disciplina e nota) na aplicação web e enviar mensagens para o aplicativo Android através do servidor GCM;
8. **Testes de aceitação** – O teste de aceitação irá garantir que o sistema possui as funcionalidades de acordo com os requisitos do usuário e critérios de aceitação; O teste será realizado com um grupo usuários e é imprescindível para a validação das funcionalidades da aplicação e, embora tenha sido planejado a realização destes testes, não houve tempo hábil para sua execução;
9. **Ajuda online** – A aplicação **UFMS Mobile** terá disponibilizada uma página de ajuda online para orientar os usuários a melhor utilizar o sistema; a ajuda irá incluir passos para acessar as principais funcionalidades das aplicações através da documentação e um seção de perguntas frequentes.

Referências Bibliográficas

- Amazon Web Services, Inc. (2016). O que é a computação em nuvem? Disponível em: <https://aws.amazon.com/pt/what-is-cloud-computing/>. Acessado em 04/03/2016.
- Bastos, D. F. (2014). O que é model-view-controller (mvc)? Disponível em: https://www.oficinadanet.com.br/artigo/desenvolvimento/o_que_e_model-view-controller_mvc. Acessado em 23/07/2014.
- Dall’Oglio, M. (2013). Aplicativo android para o ambiente univates virtual. *Centro Univer-sitário Univates*, page 65. Acessado em 26/02/2016.
- Dall’Oglio, P. (2009). *PHP: programando com orientação a objetos - Segunda edição*. Editora Novatec.
- eLinux, Org. (2011). Android architecture. Disponível em: http://elinux.org/Android_Architecture. Acessado em: 02/03/2016.
- Glauber, N. (2014). Google cloud messaging. Disponível em: <http://www.nglauber.com.br/2014/02/google-cloud-messaging.html>. Acessado em 01/03/2016.
- Glauber, N. (2015). *Dominando o Android - Segunda edição*. Editora Novatec.
- Google, Inc. (2013). Starting an activity – android developers. Disponível em: <http://developer.android.com/training/basics/activity-lifecycle/starting.html>. Vi-sitado em: 04/03/2016.
- Google, Inc. (2014). Fragments – android developers. Disponível em: <http://developer.android.com/guide/components/fragments.html>. Acessado em 22/06/2014.
- Google, Inc. (2016). Requesting permissions at run time – android developers. Disponível em: <http://developer.android.com/training/permissions/requesting.html>. Acessado em 03/03/2016.
- International Data Corporation (2015). Smartphone os market share, 2015 q2. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acessado em 01/03/2016.

- JetBrains s.r.o. (2016). PhpStorm - lightning-smart php ide. Disponível em: <https://www.jetbrains.com/phpstorm/>. Acessado em: 20/03/2016.
- K19 Treinamentos (2015). *Design Patterns em Java*. K19, <http://www.k19.com.br/>. Acessado em 09/03/2015.
- Lecheta, R. R. (2010). *Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK*. Editora Novatec.
- Lecheta, R. R. (2015). *Web Services RESTful*. Editora Novatec.
- Lorenzoni, L. (2016). Em 2015, android teve dobro de downloads de apps em relação a ios. Disponível em: <http://www.psafe.com/blog/em-2015-android-teve-dobro-de-downloads-de-apps-em-relacao-a-ios/>. Acessado em 01/03/2016.
- Open Handset Alliance (2007). Open handset alliance. Disponível em: <http://www.openhandsetalliance.com/>. Acessado em: 10/03/2016.
- Pereira, A. P. (2009). O que é xml? Disponível em: <http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>. Acessado em 25/06/2014.
- Pereira, Lúcio Camilo O.; da Silva, Michel L. (2009). *Android para Desenvolvedores*. Brasport.
- Portal Brasil (2014). Cerca de 48% dos brasileiros usam internet regularmente. Disponível em: <http://www.brasil.gov.br/governo/2014/12/cerca-de-48-dos-brasileiros-usam-internet-regularmente>. Acessado em 01/03/2014.
- PUC Minas (2015). Puc minas mobile – google play store. Disponível em: <https://goo.gl/uvukLs>. Acessado em 05/03/2016.
- Rahman, F. P. (2011). Connection between php (server) and android (client) using http and json. Disponível em: <https://fahmirahman.wordpress.com/2011/04/21/connection-between-php-server-and-android-client-using-http-and-json/>. Acessado em 25/06/2014.
- Sampaio, C. (2007). *Guia do Java Enterprise Edition 5 – Desenvolvendo aplicações corporativas*. Brasport.
- Simões, R. F. A. (2007). Alternativas ao xml: Yaml e json. *Conferência Nacional, Lisboa*.
- Six, J. (2012). *Segurança de Aplicativos Android*. Editora Novatec.

- Sur, A. (2014). Restful crud operation on a wcf service. Disponível em: <http://www.codeproject.com/Articles/115054/Restful-Crud-Operation-on-a-WCF-Service>. Acessado em 23/07/2014.
- Taurion, C. (2009). *Cloud computing: computação em nuvem: transformando o mundo da tecnologia da computação*. Brasport.
- TIOBE software BV (2016). Tiobe index for march 2016. Disponível em: http://www.tiobe.com/tiobe_index. Acessado em 02/03/2016.
- Trevisan, D. F., da Silva Sacchi, R. P., and Sanabria, L. (2013). Estudo do padrão avançado de criptografia aes – advanced encryption standard. *RITA - Revista de Informática Teórica e Aplicada*, 20(1):12. Acessado em: 02/03/2015.
- UFPE (2015). Ufpe mobile – rede virtus. Disponível em: <https://virtus.ufpe.br/>. Acessado em 04/03/2016.
- Vincent, J. (2015). Android is now used by 1.4 billion people. Disponível em: <http://www.theverge.com/2015/9/29/9409071/google-android-stats-users-downloads-sales>. Acessado em 01/03/2016.

Apêndice A

Engenharia de requisitos

A engenharia de requisitos visa identificar requisitos do sistema, elaborar protótipos, refinar e validar requisitos que, como resultado final, servirão como base para construção de software. Nesta seção, iremos abordar as regras que ditam o caminho a ser seguido no desenvolvimento deste trabalho, bem como identificar os usuários e tecnologias envolvidas, restrições, limitações e critérios de qualidade de software. Ao fim deste capítulo, será apresentada uma visão aprimorada da lista dos requisitos deste trabalho.

A.1 Regras de negócios

As regras de negócios representam as diretrizes ou restrições de uma instituição – ou projeto – sobre uma situação específica e visam indicar qual caminho seguir, independente da aplicação. As regras de negócios indicam “como o negócio funciona”, refletindo a política da organização e são aplicadas às mais diversas situações, não se restringindo apenas ao desenvolvimento de software aplicado. Entretanto, em Engenharia de Software, regras de negócios são convertidas em requisitos e devem ser implementadas como características de um sistema. Neste trabalho, foram levantadas as seguintes regras de negócios:

- O aluno deve alcançar a nota mínima definida pela instituição para ser aprovado;
- Uma disciplina não pode ter mais de 50 alunos matriculados por período;
- O professor terá acesso somente às disciplinas que está habilitado a lecionar;
- O aluno somente receberá atualizações das disciplinas em que está matriculado;
- Um professor pode lecionar mais de uma disciplina ao mesmo tempo;
- Uma disciplina só poderá ser lecionada por um professor ao mesmo tempo;
- O aluno deve estar devidamente matriculado em uma instituição de ensino e possuir código de registro do aluno;

- A instituição deve definir a duração do período letivo (bimestre, trimestre e semestre).

Para acessar a aplicação, o usuário deve:

- Estar cadastrado no sistema;
- Ter acesso à internet;
- Preencher os campos de e-mail e senha com dados válidos.

Para acessar suas notas:

- Estar devidamente autenticado no sistema;
- Estar matriculado e cursando alguma disciplina;
- As disciplinas devem ter notas disponíveis para acesso dos alunos.

A.2 Lista de requisitos

A lista de requisitos representa as características deste projeto, tanto do aplicativo Android, quando da aplicação web, as possibilidades de interação de alunos e professores, desempenho da aplicação, restrições do projeto e usabilidade.

O aplicativo Android deverá:

- Executar em dispositivos Android com versão acima da 4.2;
- Exibir eventos disponíveis através de lista simples e calendário;
- Mostrar média de nota das avaliações dos alunos;
- Agrupar os materiais disponíveis em todos os eventos por disciplina;
- Mostrar grupo de alunos participantes de uma disciplina;
- Exibir notas disponibilizadas pelos professores das disciplinas;
- Mostrar informações geral do curso;
- Receber notificações quando houver atualização disponível no servidor de eventos, disciplinas e notas;
- Atualizar dados da aplicação utilizando gestos;
- Sincronizar com o servidor automaticamente quando conectado à internet;

- Carregar dados de forma automática – ao invés de exibir notificação – se o usuário estiver com a aplicação aberta e tiver alguma atualização no servidor;
- Utilizar os conceitos do material design na composição da sua interface gráfica;
- Destacar os campos obrigatórios aos usuários através de mensagens indicativas;
- Ser capaz de exibir/ocultar a senha sendo digitada;
- Abrir o menu lateral de opções (Navigation Drawer) utilizando cliques e gestos de deslizar;
- Ser capaz de atualizar as listas de dados utilizando gestos de deslizar na tela;
- Informar o usuário quando o mesmo não estiver conectado à internet;
- Informar o usuário se a aplicação estiver configurada para baixar anexos via WI-FI somente;
- Disponibilizar o conteúdo offline;
- Retornar requisições do servidor em até 3 segundos;
- Validar os dados entrados pelo usuário;
- Abrir o menu de opções lateral automaticamente pela primeira vez que o usuário executa o aplicativo, introduzindo o recurso para os usuários que não conhecem o mesmo;
- Alternar o tipo de teclado do sistema dependendo do tipo de entrada de dados.

A aplicação web deverá:

- Enviar mensagem para aplicativo Android sobre atualização;
- Utilizar *prepared statements* nas interações com o banco de dados;
- Autenticar o usuário através de um algoritmo AES de criptografia.

O aluno poderá:

- Avaliar uma disciplina através de uma nota de 1 a 5;
- Editar nota atribuída à determinada disciplina;
- Marcar evento como favorito para utilizar como filtro posteriormente;
- Avaliar desempenho acadêmico através de gráficos intuitivos baseados nas notas individuais e média coletiva;

- Buscar por disciplinas, eventos e notas;
- Configurar a aplicação conforme sua preferência;
- Enviar feedback para desenvolvedor.

O professor poderá:

- Cadastrar notas para as disciplinas;
- Matricular aluno em disciplina;
- Cadastrar eventos em suas respectivas disciplinas, com informações relevantes: nome, disciplina relacionada, data de criação, anexos e data limite;
- Editar registros;
- Fazer upload de material para suas disciplinas;
- Buscar registros;
- Excluir itens do servidor;
- Cadastrar disciplinas com suas respectivas informações: ementa, carga horária, tipo, professor responsável.

A.3 Documento de requisitos

Nesta seção será abordado o documento de requisitos deste projeto, elencando todas as etapas desde o planejamento inicial do sistema **UFMS Mobile** até o gerenciamento do requisitos, dependências e restrições.

A.3.1 Escopo

O escopo da aplicação delimita as fronteiras do projeto, ou seja, o que o sistema deverá contemplar e as características que não serão abordadas devido a restrições de tempo.

Tabela A.1: Escopo funcional da aplicação.

Escopo
Desenvolver aplicação Android para acesso dos alunos de instituições de ensino
Implementar aplicação web para gerenciamento de conteúdo na web
Gerenciar alunos matriculados na instituição, cadastrando-os no software
Estreitar relação aluno-professor

Tabela A.2: Fora do escopo deste trabalho.

Fora do escopo do projeto
Realizar backup de dados na aplicação Android
Desenvolver aplicação para outras plataformas de dispositivos móveis
Implementar funcionalidade do tipo “chat” (troca de mensagens diretas entre alunos) na aplicação Android
Prover suporte para outros dispositivos além de smartphones e tablets

A.3.2 Restrições

As restrições representam as condições de desenvolvimento e construção do projeto, tais como requisitos e tomada de decisão. O prazo para a entrega do projeto é uma restrição que precisa ser seguida à risca, uma vez que o atraso acarreta consequências no resultado esperado do trabalho.

Tabela A.3: Restrições do projeto.

Restrição	Impacto
Aplicação web deverá ser acessada via internet	Arquitetura
Utilizar o Paradigma Orientado a Objetos no desenvolvimento	Paradigma de desenvolvimento
O aplicativo Android deverá utilizar Java	Design do sistema
A aplicação web deve utilizar PHP, HTML e frameworks para personalização visual	Design do sistema
O software deve apresentar uma versão testável em até 04 meses do início do projeto	Cronograma do projeto

A.3.3 Dependências

Visa identificar as dependências e fatores externos à aplicação. Declarar recursos, componentes de software e serviços que são necessários para o correto funcionamento do projeto.

Tabela A.4: Dependências do projeto.

Dependência	Descrição
Serviço de hospedagem	Um servidor de hospedagem para aplicação web deve ser selecionado e contratado para disponibilizar o portal na web
Serviços do Google	Serviços do Google para utilização de recursos na plataforma Android, como: Google Cloud Messaging
Conta na loja oficial do Android (Play Store)	Contratar uma conta de desenvolvedor para publicar produtos na loja oficial de aplicativos para Android

A.3.4 Descrição geral do sistema

Nesta seção será abordada uma descrição geral das funcionalidades da aplicação a ser desenvolvida, facilitando a interpretação dos requisitos do projeto que serão definidos posteriormente neste documento.

Visão geral das características do produto

A aplicação UFMS Mobile é um projeto que contempla o desenvolvimento de um sistema web para gerenciamento de conteúdo de disciplinas de uma instituição de ensino e uma aplicação Android que fará o consumo do conteúdo disponível a partir da web. Este projeto será implementado utilizando o paradigma de orientação a objetos e alguns padrões de desenvolvimento, como o MVC e o padrão Singleton.

Ambiente de operação

O sistema web será desenvolvido utilizando a linguagem PHP e hospedado em servidor na internet, sem restrição de hardware ou software. O aplicativo Android será desenvolvido em Java e deverá executar em smartphones e tablets com acesso à internet e sistema Android na versão 4.2 ou superior.

Características dos usuários

A aplicação deverá ter como usuários professores e alunos de instituições de ensino com conhecimentos básicos de acesso à internet, portais web e sistema Android.

A.3.5 Requisitos de software

Os requisitos representam as características do sistema, suas funcionalidades, restrições de desempenho, segurança, usabilidade, entre outros.

Requisitos funcionais

São associados com funcionalidades que o sistema deve executar para que os objetivos dos usuários sejam atingidos. Os requisitos funcionais devem ser identificados por um código sequencial, sua prioridade, descrição e entrada e saídas esperadas pelo usuário.

Tabela A.5: Requisito funcional RFN01.

RFN01	Avaliar uma disciplina através de uma nota de 1 a 5
Prioridade	Média
Descrição	Permitir que o aluno avalie uma disciplina que esteja cursando (ou cursada) com uma nota de 1 a 5, proporcionando feedback para os professores
Entrada	Valor real entre 1 e 5
Operações executadas	(1.) Verificação se a nota está entre 1 e 5 (2.) Enviar nota para servidor através do web service (3.) Atualizar média de nota da disciplina
Saída	Nota atribuída pelo aluno é exibida no componente barra de avaliação (estrelas), que mostrará o número de estrelas selecionado de acordo como a nota

Tabela A.6: Requisito funcional RFN02.

RFN02	Editar nota atribuída à determinada disciplina
Prioridade	Média
Descrição	Permitir que o aluno altere uma avaliação concedida anteriormente a uma disciplina
Entrada	Valor real entre 1 e 5
Operações executadas	(1.) Verificação se a nota está entre 1 e 5 (2.) Enviar nota para servidor através do web service (3.) Alterar nota salva no banco de dados remoto e local (4.) Atualizar média de nota da disciplina
Saída	Nota atribuída pelo aluno é exibida no componente barra de avaliação (estrelas), que mostrará o número de estrelas selecionado de acordo como a nota

Tabela A.7: Requisito funcional RFN03.

RFN03	Marcar evento como favorito para utilizar como filtro posteriormente
Prioridade	Média
Descrição	Permitir que o aluno marque um evento como favorito, clicando em um botão de “coração” (favorito)
Entrada	Pressionar o botão de favorito em detalhes do evento
Operações executadas	Inserir no banco de dados na tabela de eventos favoritos código do evento selecionado
Saída	Ícone de favorito muda o aspecto visual para um ícone com preenchimento de cor, indicando que está selecionado

Tabela A.8: Requisito funcional RFN04.

RFN04	Avaliar desempenho acadêmico através de gráficos intuitivos baseados nas notas individuais e média coletiva
Prioridade	Média
Descrição	Permitir que o aluno visualize suas notas plotadas em gráficos que permitem fazer análises de desempenho do mesmo
Entrada	Selecionar opção “Desempenho view” no menu de opções na tela de notas
Operações executadas	(1.) Ordenar notas em ordem decrescente (2.) Calcular média da turma para todas as atividades com notas disponíveis (3.) Plotar gráficos
Saída	Gráficos com os valores das notas plotados na tela

Tabela A.9: Requisito funcional RFN05.

RFN05	Buscar por registros nas categorias: disciplinas, eventos e notas
Prioridade	Média
Descrição	Permitir que o aluno busque por um registro específico na lista de itens nas categorias: disciplinas, eventos e notas
Entrada	Termo a ser buscado na lista do tipo <i>string</i>
Operações executadas	(1.) Clicar no botão de lupa (pesquisar) na Toolbar (2.) Buscar itens similares ao termo digitado
Saída	Lista com resultado da busca exibida na tela

Tabela A.10: Requisito funcional RFN06.

RFN06	Configurar a aplicação conforme sua preferência
Prioridade	Alta
Descrição	Permitir que o aluno configure a aplicação Android baseado na sua preferência de uso
Entrada	Tipo boolean – elementos “checados” ou habilitados correspondem ao valor booleano verdadeiro; do contrário, o valor atribuído é falso
Operações executadas	(1.) Abrir tela de configurações do aplicativo (2.) Marcar/desmarcar configuração
Saída	(1.) Elemento gráfico (checkbox ou switch) marcado ou desmarcado, desabilitado ou habilitado (2.) Aplicação lê as configurações salvas e adapta seu comportamento

Tabela A.11: Requisito funcional RFN07.

RFN07	Enviar feedback para desenvolvedor
Prioridade	Baixa
Descrição	Possibilitar que o usuário reporte um problema da aplicação para o desenvolvedor do produto
Entrada	(1.) E-mail do usuário remetente da mensagem (2.) Texto do tipo <i>string</i> com mensagem para desenvolvedor
Operações executadas	(1.) Campos de dados validados para evitar que o campo de texto esteja vazio (2.) Mensagem enviada ao servidor
Saída	Desenvolvedor recebe e-mail com mensagem de feedback

Tabela A.12: Requisito funcional RFN08.

RFN08	Cadastrar eventos em suas respectivas disciplinas
Prioridade	Alta
Descrição	Possibilitar que o professor cadastre um evento em suas respectivas disciplinas com informações relevantes: nome, disciplina relacionada, data de criação, anexos e data limite
Entrada	(1.) Nome do evento do tipo <i>string</i> (2.) Descrição do evento do tipo <i>string</i> (3.) Data limite do tipo <i>date</i> (4.) Tipo de evento do tipo <i>int</i> (chave estrangeira) (5.) Arquivo de anexo do tipo <i>file</i>
Operações executadas	(1.) Campos de dados validados de acordo com o tipo esperado de dado (2.) Evento é adicionado no banco de dados remoto (3.) Servidor GCM é notificado a respeito de uma atualização disponível no servidor
Saída	Aplicativo Android é notificado a respeito do novo evento adicionado

Tabela A.13: Requisito funcional RFN09.

RFN09	Editar registros
Prioridade	Alta
Descrição	Permitir que professores editem registros previamente adicionados, como informações sobre eventos, disciplinas e notas
Entrada	(1.) Alterar campos que deseja modificar
Operações executadas	(1.) Campos de dados validados de acordo com o tipo esperado de dado (2.) Registro é alterado no banco de dados do servidor (3.) Servidor GCM é notificado a respeito de uma atualização disponível no servidor
Saída	Usuário é notificado sobre conteúdo disponível

Tabela A.14: Requisito funcional RFN10.

RFN10	Fazer upload de material para suas disciplinas
Prioridade	Alta
Descrição	Possibilitar que o professor da disciplina adicione documentos como anexo dos eventos
Entrada	(1.) Clicar no botão “Arquivo” no formulário de cadastro de evento (2.) Carregar documento que deseja anexar
Operações executadas	(1.) Tipo de arquivo carregado é validado (pdf, doc, ppt, xls) (2.) Arquivo é enviado para pasta uploads no servidor
Saída	Arquivo é disponibilizado para download na tela de detalhes do evento no aplicativo Android

Tabela A.15: Requisito funcional RFN11.

RFN11	Buscar registros (professor)
Prioridade	Média
Descrição	Possibilitar que o professor pesquise por registros na aplicação web
Entrada	(1.) Termo a ser pesquisado do tipo <i>string</i>
Operações executadas	(1.) Termo digitado é buscado no banco de dados (2.) Lista de ocorrências é retornada
Saída	Lista com os resultados é exibida conforme a palavra é digitada no campo de busca

Tabela A.16: Requisito funcional RFN12.

RFN12	Excluir itens do servidor
Prioridade	Alta
Descrição	Possibilitar que o professor exclua registros relacionados às suas disciplinas. Somente usuários do tipo professor têm permissão para excluir registros
Entrada	(1.) Clicar no botão “excluir” no item da lista que deseja excluir
Operações executadas	(1.) Janela de diálogo é acionada confirmando exclusão (2.) Se confirmado, registro é excluído do banco de dados
Saída	Banco de dados local do aplicativo Android é sincronizado quando a aplicação for executada, atualizando a lista de registros e removendo os que foram excluídos

Tabela A.17: Requisito funcional RFN13.

RFN13	Cadastrar disciplinas com suas respectivas informações: ementa, carga horária, tipo, professor responsável
Prioridade	Alta
Descrição	Permitir que o professor adicione disciplinas na aplicação web
Entrada	(1.) Ementa da disciplina do tipo <i>string</i> (2.) Carga horária do tipo <i>int</i> (3.) Tipo de disciplina do tipo <i>int</i> (chave estrangeira) (4.) Professor responsável do tipo <i>int</i> (chave estrangeira) (5.) Aluno do tipo <i>int</i> (chave estrangeira)
Operações executadas	(1.) Campos de dados são validados conforme tipo de dado esperado (2.) Disciplina é adicionada no servidor (3.) Servidor GCM é acionado, enviando mensagem para os dispositivos Android interessados
Saída	Notificação é lançada no aplicativo Android informando da atualização disponível

Tabela A.18: Requisito funcional RFN14.

RFN14	Cadastrar notas para as disciplinas
Prioridade	Alta
Descrição	Permitir que o professor adicione notas para as disciplinas que leciona
Entrada	(1.) Selecionar disciplina desejada (2.) Marcar alunos que terão notas atribuídas (3.) Nota do aluno do tipo <i>float</i>
Operações executadas	(1.) Campo de nota é validado com o tipo de dado esperado (2.) Nota é enviada para o servidor (3.) Servidor GCM é acionado, enviando mensagem para os dispositivos Android interessados
Saída	Notificação é lançada no aplicativo Android informando da atualização disponível

Tabela A.19: Requisito funcional RFN15.

RFN15	Matricular aluno em disciplina
Prioridade	Alta
Descrição	Possibilitar que o professor cadastre alunos em disciplinas que os mesmos estão cursando
Entrada	(1.) Marcar disciplinas em que o aluno está matriculado; opções marcadas do tipo <i>int</i> 0 ou 1 (2.) Clicar botão “Matricular”
Operações executadas	(1.) Formulário de cadastro é validado (2.) Se confirmado, o aluno é registrado no servidor na internet (3.) Servidor GCM é acionado, enviando mensagem para os dispositivos Android interessados
Saída	Aluno começa a receber notificações das disciplinas em que está registrado

Tabela A.20: Requisito funcional RFN16.

RFN16	Enviar mensagem para aplicativo Android sobre atualização
Prioridade	Alta
Descrição	Permitir que os alunos recebam notificações acerca de atualizações no servidor, alertando os usuários interessados através de notificação
Entrada	Mensagem a ser enviada para o aplicativo Android
Operações executadas	(1.) Enviar mensagem através do servidor GCM para aplicação Android (2.) Identificar o tipo de mensagem (3.) Extrair mensagem recebida (4.) Montar e lançar notificação no aplicativo Android
Saída	Aplicação Android recebe notificação informando a respeito de atualizações no servidor

Tabela A.21: Requisito funcional RFN17.

RFN17	Destacar os campos obrigatórios aos usuários através de mensagens indicativas
Prioridade	Média
Descrição	Permitir que o usuário identifique quais campos são de preenchimento obrigatório, indicando-os com mensagens diretas a respeito do tipo de dado esperado
Entrada	(1.) Clicar em um campo de texto do aplicativo Android (2.) Digitar algo e pressionar o botão de ação
Operações executadas	Texto digitado é validado de acordo com o tipo de informação esperado para campo de texto
Saída	Mensagem de erro é mostrada informando o tipo de dado esperado e destacando o campo em questão

Tabela A.22: Requisito funcional RFN18.

RFN18	Ser capaz de exibir/ocultar a senha sendo digitada
Prioridade	Baixa
Descrição	Permitir que o aluno, ao acessar a tela de login no aplicativo Android, possa exibir e ocultar sua senha
Entrada	(1.) Clicar no botão de “olho” (visível/invisível) dentro da caixa de texto de senha
Operações executadas	(1.) Ícone muda de aparência, indicando que a senha está visível ou invisível
Saída	Caracteres digitados da senha são exibidos ou ocultados

Tabela A.23: Requisito funcional RFN19.

RFN19	Exibir eventos disponíveis através de lista simples e calendário
Prioridade	Média
Descrição	Possibilitar que o aluno alterne entre o modo lista e calendário de eventos
Entrada	(1.) Abrir tela de lista de eventos (2.) Clicar no menu de opções em “Ver calendário”
Operações executadas	(1.) Lista de eventos é ocultada (2.) Calendário é exibido no lugar
Saída	Datas dos eventos são marcadas no calendário exibido

Tabela A.24: Requisito funcional RFN20.

RFN20	Mostrar média de nota das avaliações dos alunos
Prioridade	Média
Descrição	Exibir a média das notas atribuídas pelos alunos a cada disciplina
Entrada	Abrir a tela de lista de disciplinas
Operações executadas	(1.) Calcular a média entre o número de alunos que votou e a soma das notas
Saída	Exibir ao lado do nome da disciplina sua média correspondente na tela de lista de disciplinas

Tabela A.25: Requisito funcional RFN21.

RFN21	Agrupar os materiais disponíveis em todos os eventos por disciplina
Prioridade	Alta
Descrição	Agrupar na aba “Materiais” todos os anexos disponíveis por eventos que estão associados àquela disciplina
Entrada	(1.) Clicar em uma disciplina na tela de lista de disciplinas (2.) Clicar na aba “Materiais” na tela de detalhes da disciplina
Operações executadas	(1.) Selecionar os anexos dos eventos que pertencem à determinada disciplina (2.) Agrupá-los para serem disponibilizados em conjunto
Saída	Lista de anexos agrupados daquela disciplina é mostrada a aba de “Materiais”

Tabela A.26: Requisito funcional RFN22.

RFN22	Mostrar grupo de alunos participantes de uma disciplina
Prioridade	Alta
Descrição	Exibir os alunos participantes (matriculados) em uma disciplina
Entrada	(1.) Clicar em uma disciplina na tela de lista de disciplinas (2.) Clicar na aba “Alunos” na tela de detalhes da disciplina
Operações executadas	Listar alunos matriculados na mesma turma
Saída	Lista dos alunos na mesma turma é exibida na aba “Alunos”

Tabela A.27: Requisito funcional RFN23.

RFN23	Exibir notas disponibilizadas pelos professores das disciplinas
Prioridade	Alta
Descrição	Exibir notas das atividades por disciplina em que o aluno está matriculado
Entrada	(1.) Abrir a tela de notas (2.) Seleciona uma disciplina na lista de disciplinas disponíveis
Operações executadas	Selecionar atividades relacionadas a uma disciplina
Saída	Lista de atividades disponíveis por disciplina, com suas respectivas notas

Tabela A.28: Requisito funcional RFN24.

RFN24	Mostrar informações geral do curso
Prioridade	Média
Descrição	Exibir detalhes do curso, como: nome, carga horária, período e área
Entrada	Clicar em curso no menu de opções
Operações executadas	Carregar informações acerca do curso em que o aluno está matriculado
Saída	Tela com detalhes do curso

Tabela A.29: Requisito funcional RFN25.

RFN25	Receber notificações quando houver atualização disponível no servidor
Prioridade	Alta
Descrição	Alertar o usuário através de uma notificação no aplicativo Android quando houver atualizações de eventos, disciplinas e notas disponíveis no servidor
Entrada	Não se aplica
Operações executadas	(1.) Enviar mensagem para servidor GCM quando conteúdo for adicionado (evento, disciplina ou nota) na aplicação web (2.) Servidor GCM envia mensagem para todos os dispositivos Android registrados no servidor, que tenham a aplicação instalada, e que estejam matriculados nas disciplinas correspondentes
Saída	Notificação é disparada no aplicativo Android alertando o usuário que algo novo foi adicionado no servidor

Tabela A.30: Requisito funcional RFN26.

RFN26	Atualizar dados da aplicação utilizando gestos
Prioridade	Média
Descrição	Permitir que o aluno atualize manualmente a lista de eventos, disciplinas e notas utilizando gestos de cima para baixo com o dedo na tela do dispositivo
Entrada	Tocar em qualquer ponto da lista de itens, manter o dedo pressionado, arrastar para baixo e soltar o dedo
Operações executadas	(1.) Verifica no servidor a existência de atualizações (2.) Sincroniza com o banco de dados local do aplicativo Android
Saída	Atualiza a lista de itens exibida ao usuário

Tabela A.31: Requisito funcional RFN27.

RFN27	Sincronizar com o servidor automaticamente quando conectado à internet
Prioridade	Média
Descrição	Sincronizar a aplicação Android automaticamente com o servidor caso o dispositivo perca a conexão com a internet durante a interação com o usuário
Entrada	Não se aplica
Operações executadas	(1.) Verificar a conexão com a internet (2.) Se estiver conectado, consultar o servidor por atualizações (3.) Se houver algo novo disponível, sincronizar aplicativo Android
Saída	Lista de registros é automaticamente atualizada ao retomar conexão com a internet se houver conteúdo novo no servidor

Tabela A.32: Requisito funcional RFN28.

RFN28	Carregar dados de forma automática – ao invés de exibir notificação – se o usuário estiver com a aplicação aberta e houver alguma atualização no servidor
Prioridade	Média
Descrição	Atualizar lista de registros (eventos, disciplinas e notas) de maneira automática, sem disparar notificação, se o aluno já estiver com o aplicativo executando
Entrada	Não se aplica
Operações executadas	(1.) Receber mensagem do servidor GCM acerca de atualizações disponíveis (2.) Não disparar notificação (3.) Sincronizar aplicativo com o servidor
Saída	Banco de dados local da aplicação Android é sincronizado com o servidor

Tabela A.33: Requisito funcional RFN29.

RFN29	Informar o usuário quando o mesmo não estiver conectado à internet
Prioridade	Média
Descrição	Avisar o usuário quando não houver conexão com a internet, para que o mesmo tome alguma decisão
Entrada	Não se aplica
Operações executadas	(1.) Verificar se existe algum tipo de conexão (WI-FI ou dados móveis) disponível
Saída	Se não houver conexão, envia uma mensagem através da Snackbar (seção 3.15.4) informando o usuário acerca do ocorrido

Tabela A.34: Requisito funcional RFN30.

RFN30	Informar o usuário se a aplicação estiver configurada para baixar anexos via WI-FI somente
Prioridade	Média
Descrição	Avisar o aluno se o aplicativo Android estiver configurada para baixar anexos somente via WI-FI, se o mesmo tentar realizar o download de um documento
Entrada	Clicar para baixar um documento com somente os dados móveis ativados, sendo a aplicação configurada para baixar via WI-FI
Operações executadas	(1.) Verificar as configurações do aplicativo (2.) Checar o tipo de conexão do aparelho, e se está de acordo com as preferências do usuário
Saída	Se estiver conectado via dados móveis, enviar mensagem através da Snackbar alertando o usuário

Tabela A.35: Requisito funcional RFN31.

RFN31	Abrir o menu lateral de opções (Navigation Drawer) utilizando cliques ou gestos de deslizar
Prioridade	Alta
Descrição	Abrir o menu principal do aplicativo clicando no menu “sanduíche” ou utilizando gesto de deslizar o dedo no canto da tela, no sentido esquerda para direita
Entrada	Clicar no menu “sanduíche” na Toolbar ou utilizar gesto na tela
Operações executadas	Captar interação do usuário com clique ou gesto
Saída	O menu principal da aplicação é aberto

Requisitos não funcionais

Estes requisitos definem critérios externos à aplicação, que não estão ligados às funcionalidades do sistema e, portanto, não são enquadrados pela análise dos requisitos funcionais. Alguns tipos de requisitos não funcionais podem incluir: usabilidade, portabilidade, performance, disponibilidade e segurança. Todos esses aspectos citados – entre outros –, embora não estejam diretamente ligados às funcionalidades do sistema, são imprescindíveis para o sucesso de qualquer projeto.

Usabilidade

Definem os requisitos de usabilidade do sistema, como cores, padrões de design, estrutura de menus, entre outros. A tabela A.3.5 representa a lista de requisitos não funcionais para esta categoria.

Tabela A.36: Requisitos não funcionais de usabilidade.

Req. #	Descrição
RNF01	Utilizar os conceitos do material design na composição da sua interface gráfica
RNF02	Abrir o menu de opções lateral automaticamente pela primeira vez que o usuário executa o aplicativo, introduzindo o recurso para os usuários que não conhecem o mesmo
RNF03	Alternar o tipo de teclado do sistema dependendo do tipo de entrada de dados; por exemplo: campo contendo RGA do aluno abre o teclado numérico; campo com e-mail abre o teclado com ‘@’

Portabilidade

Representa as restrições para o ambiente de execução da aplicação. Neste trabalho, definiremos apenas uma restrição para a execução do aplicativo Android, como mostrado a seguir:

RNF04 – Executar em dispositivos Android com versão acima da 4.2.

Performance

Os requisitos específicos de desempenho que o sistema deverá atender. Neste projeto, o seguinte requisito do tipo performance deverá ser atendido:

RNF05 – Retornar requisições do servidor em até 3 segundos.

Disponibilidade

Determina a frequência ou como a aplicação e seu conteúdo ficam disponíveis para o usuário. Neste trabalho, o seguinte requisito deve ser aplicado nesta categoria:

RNF06 – Disponibilizar o conteúdo do aplicativo Android offline.

Segurança

Define os critérios de segurança do sistema que devem ser seguidos em sua implementação. A tabela A.3.5 exibe a lista de requisitos de segurança que serão aplicados neste projeto.

Tabela A.37: Requisitos não funcionais de segurança.

Req. #	Descrição
RNF07	Validar os dados entrados pelo usuário
RNF08	Utilizar <i>prepared statements</i> nas interações com o banco de dados
RNF09	Autenticar o usuário através de um algoritmo AES de criptografia

Banco de dados

A aplicação Android irá utilizar o banco de dados nativo da plataforma, o SQLite 3 (seção 3.12.1). Já o portal web irá utilizar o MySQL (seção 3.12.2) para armazenar dados no servidor.

Sistema operacional

A aplicação móvel utilizará o sistema operacional Android na sua execução. Já o sistema web não tem restrição, uma vez que pode ser acessado a partir de qualquer sistema operacional.

B.0.8 Diagrama Relacional

O diagrama relacional de banco de dados representa as tabelas – e suas ligações – onde os dados deste projeto são armazenados. O esquema relacional desta aplicação é representado na figura B.3.

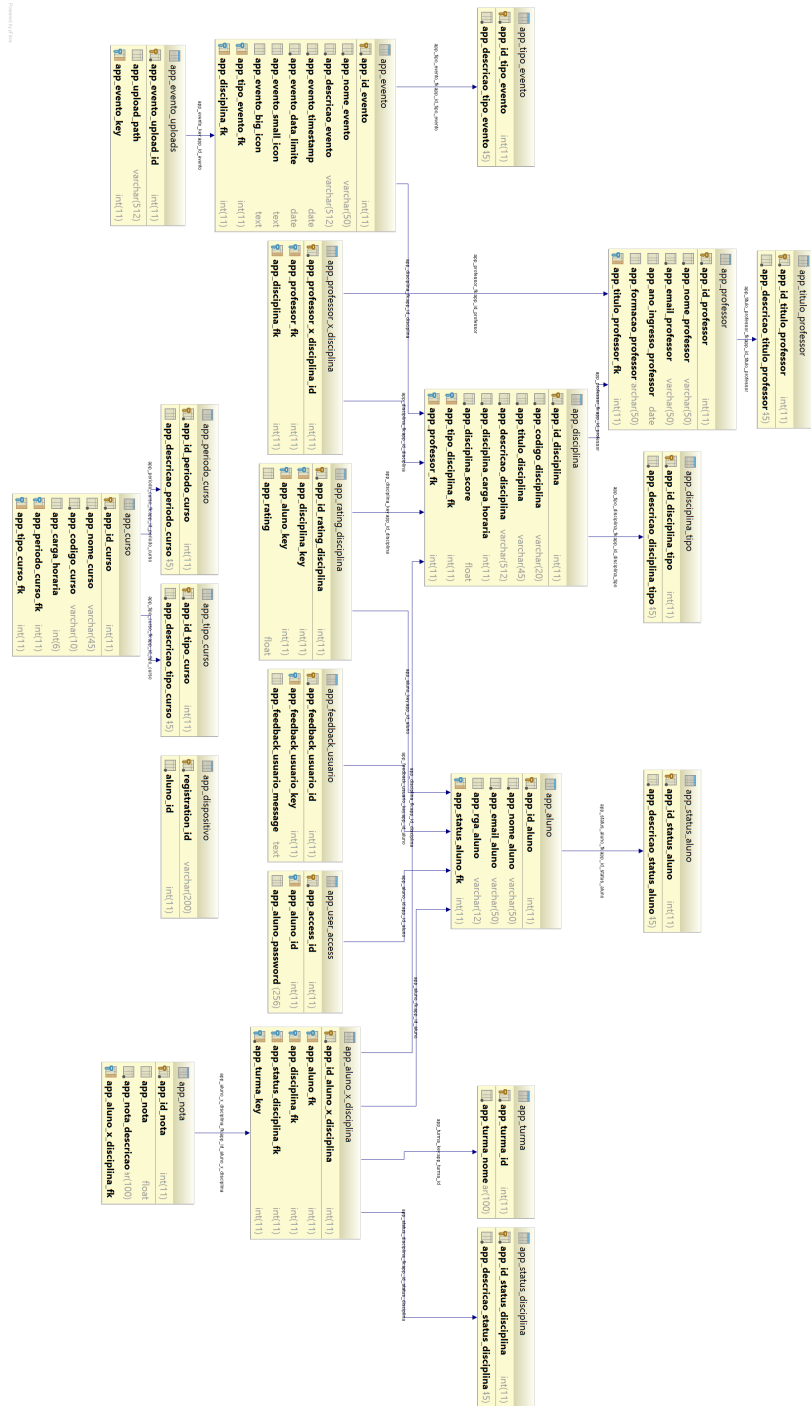


Figura B.3: Diagrama relacional de banco de dados do projeto.

B.0.9 Pacotes do projeto

O esquema apresentado na figura B.4 mostra os pacotes do projeto Android no Android Studio. Os pacotes são separados por categoria: **código java** e **recursos da aplicação**. Os pacotes java incluem classes auxiliares, activities, fragments, objetos do java (POJO – **Plain Old Java Object**), preferências, etc. Os pacotes de recurso incluem as imagens, estilos, layouts, strings, menus, tela de preferência, etc. utilizadas na aplicação.

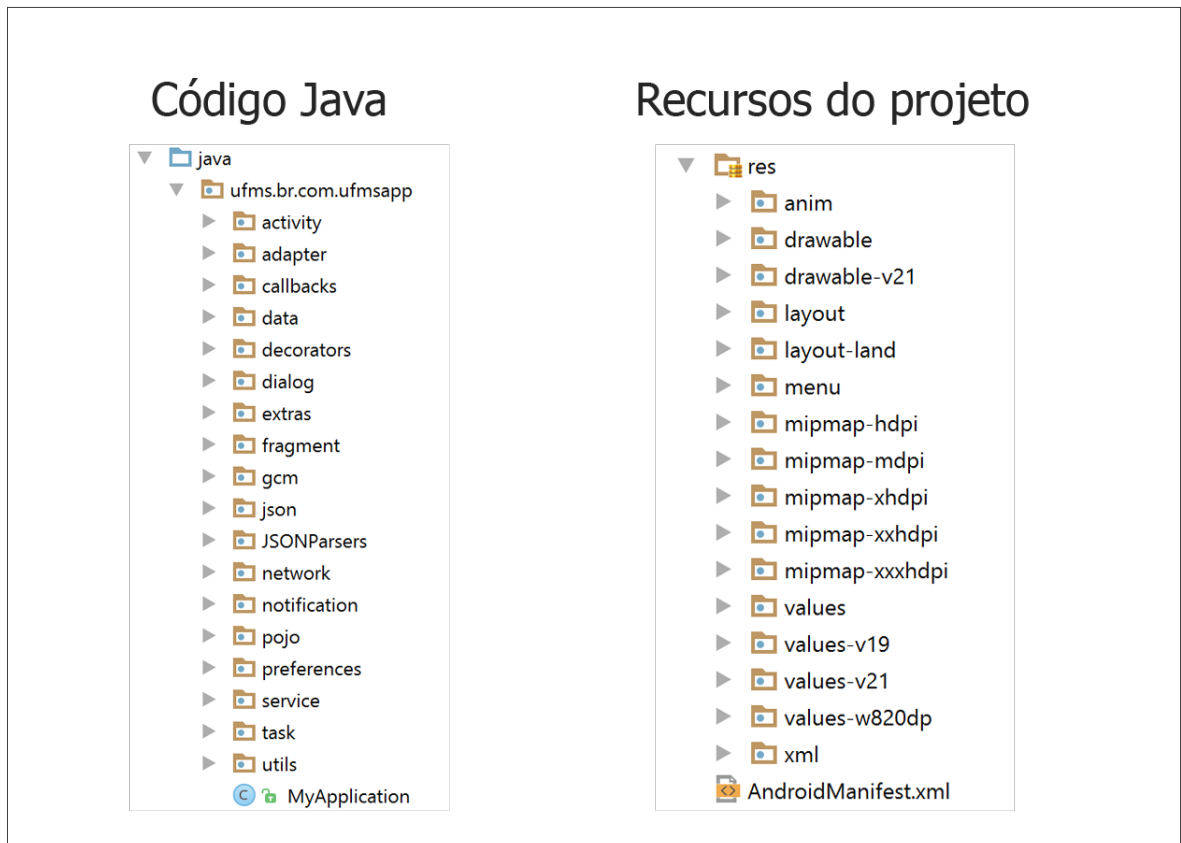


Figura B.4: Pacotes criados no desenvolvimento deste projeto.

Apêndice C

Desenvolvimento Iterativo

Para compreender de maneira mais detalhada os problemas a serem resolvidos, o desenvolvimento foi dividido em etapas de curto prazo de duração, onde ao final de cada estágio, uma nova versão do projeto foi gerada.

Iterações

As iterações duravam em média de 5 a 7 dias e eram executadas por prioridade. Tarefas complexas ou que demandavam muito tempo foram realocadas para futuras iterações. Abaixo segue um esquema representando as iterações executadas neste trabalho.

Tabela C.1: Atividades executadas na iteração 1.

Iteração 1
1. Instalação do ambiente
2. Modelagem da base de dados
3. Criação do projeto no Android Studio
4. Download e configuração das máquinas virtuais e instalação do driver usb para rodar em dispositivo real

Tabela C.2: Atividades executadas na iteração 2.

Iteração 2
1. Criação do banco de dados SQLite
2. Implementação das telas da aplicação
3. Criar adapter customizados para lista de eventos, disciplinas e notas
4. Implementar navigation view para menu da aplicação
5. Desenvolver funcionalidade de download de anexos

Tabela C.3: Atividades executadas na iteração 3.

Iteração 3
1. Aprimorar funcionalidade de download para baixar somente via WIFI
2. Criar servidor GCM para disciplinas e notas
3. Carregar objetos do banco de dados local (quando não houver conexão)
4. Arrumar conteúdo da tela explorar para mostrar mensagem se lista estiver vazia
5. Função filtrar nas telas de listas (movido)
6. Funcionalidade de busca(movido)
7. Criar notificações para a aplicação Android

Tabela C.4: Atividades executadas na iteração 4.

Iteração 4
1. Carregar tela explorar do banco de dados local
2. Funcionalidade de busca(movido)
3. Adicionar biblioteca de calendário para visualização de eventos em modo calendário
4. Agrupar anexos dos eventos em sua respectiva disciplina
5. <i>Bug</i> : Objetos são destruídos quando a tela é girada, gerando NullPointerException .

Tabela C.5: Atividades executadas na iteração 5.

Iteração 5
1. Feedback ao clicar em um item na recyclerview
2. Reter a instância do fragment (chamando o método setRetainInstance(true)) para evitar que o fragment seja destruído junto com a activity (neste caso, o fragment é novamente vinculado à activity quando a mesma é recriada ou até a mesmo pode ser utilizado por outra activity.)
3. Adapter customizado para a tela inicial
4. Funcionalidade de busca(movido)
5. Criar consulta para recuperar informações de um objeto clicado para mostrar nas telas de detalhes
6. Implementar barra de progresso nas listas ao fazer requisição no servidor (feedback para o usuário)
7. Arrumado: Objeto sendo destruído quando a tela é rotacionada (salvar antes de destruir e recuperar depois)

Tabela C.6: Atividades executadas na iteração 6.

Iteração 6
1. Implementar tela de detalhes de eventos e disciplinas utilizando abas
2. Alterar avaliação da disciplina feita pelo usuário
3. Funcionalidade de busca em lista de itens
4. Arrumar: alterar ícones das estrelas de avaliação de disciplinas

Tabela C.7: Atividades executadas na iteração 7.

Iteração 7
1. <i>Bug</i> : Banco de dados local sendo apagado se requisição do servidor retornar vazia ou estiver sem conexão com a internet
2. Armazenar id do servidor localmente para sincronizar corretamente com o banco de dados local
3. Alterar ícone de estrela para avaliação de disciplinas
4. Arrumar tabela local de upload dos eventos
5. Criar tela de curso para mostrar dados do curso

Tabela C.8: Atividades executadas na iteração 8.

Iteração 8
1. Implementar tela de configurações e salvar preferências do usuário
3. Melhorar funcionalidade de atualização das listas de objetos para mostrar mensagem de erro se não tiver conexão com a internet, evitando lançar exceção no código
4. Detectar automaticamente conexão com a internet retomada e atualizar lista
5. Arrumado: tabela local de upload criada e sincronizada com servidor
6. Arrumado: Banco de dados mantido ao fazer requisição com retorno vazio

Tabela C.9: Atividades executadas na iteração 9.

Iteração 9
1. Criar gráfico para mostrar maior nota do usuário
2. Gráfico para mostrar evolução de desempenho nas notas do usuário
3. Gráfico comparativo entre as notas do usuário com a média da turma
4. Ocultar opção de visualizar gráficos se não tiver nota disponível

Tabela C.10: Atividades executadas na iteração 10.

Iteração 10
1. Não exibir notificação do servidor se a aplicação estiver aberta; atualizar automaticamente a lista
2. Identificar se o anexo já foi baixado ao tentar fazer download de um arquivo. Se o anexo já existir, abrir e não baixar novamente
3. Alterar imagem de fundo do navigation view (menu lateral)
4. Permitir que mais de um usuário acesse a aplicação a partir do mesmo aparelho

Tabela C.11: Atividades executadas na iteração 11.

Iteração 11
1. Criar logo para a aplicação e alterar o ícone principal e na tela de login
2. Alterar esquema de cores da aplicação para se adequar às cores da UFMS
3. Criar ícones dinâmicos para os itens da lista de eventos e disciplinas utilizando a biblioteca TextDrawable (seção D.5)
4. Alterar <i>adapter</i> da lista de notas para mostrar média e turma

Tabela C.12: Atividades executadas na iteração 12.

Iteração 12
1. Inspeccionar código e corrigir erros com o <i>framework</i> lint
2. Gerar APK assinada da aplicação Android
3. Publicar aplicação na Google Play Store

Tabela C.13: Atividades executadas na iteração 13.

Iteração 13
1. Montagem do layout utilizando Material Design Lite (seção 3.22.1) e Materialize (seção 3.22.2)
2. Criação da lista de eventos utilizando DataTable (seção 3.22.3)
3. Criar, listar, alterar e deletar eventos
4. Inserção da logomarca adaptada para versão web
5. Implementação da página inicial com eventos recentes e informações sobre o projeto
6. Comunicar servidor GCM sobre atualização feita na base de dados remota (a ser implementado)

Apêndice D

Bibliotecas e Ferramentas Utilizadas

Bibliotecas e APIs nativas e de terceiros foram utilizadas para adicionar funcionalidades à aplicação, para ter acesso a serviços do Google e também personalizar o projeto. As bibliotecas essenciais no desenvolvimento do aplicativo estão listadas nesta seção, assim como ferramentas que auxiliaram na fase de implementação.

D.1 Picasso

Picasso é uma biblioteca disponibilizada pela Square Open Source¹ e permite carregar imagens através da rede de forma simples e eficiente. Alguns de seus recursos incluem:

- Cache de imagem em disco automático;
- Mecanismos de transformação de imagens, como: redimensionar, transformar, etc.;
- Recicla imagens já carregadas anteriormente para minimizar recursos do aparelho.

D.2 DSpec

DSpec é uma biblioteca *open source* disponibilizada por Lucas Rocha². DSpec permite definir em seu arquivo de layout as recomendações do Google para a interface gráfica da aplicação. O funcionamento desta biblioteca é bastante simples, basta definir um componente dspec como elemento raiz do seu arquivo de layout. Ao executar a aplicação, a activity será demarcada com linhas mostrando o posicionamento ideal dos elementos, conforme *guidelines* do Google.

¹Disponível através do endereço: <http://square.github.io/picasso/>.

²Disponível através do endereço: <https://github.com/lucasr/dspec>.

D.3 WilliamChart

WilliamChart é uma biblioteca *open source* disponível³ para Android utilizada na criação de gráficos dinâmicos. Com esta biblioteca é possível criar gráficos em linha, barra horizontal, e barra vertical, além de ser facilmente customizável.

D.4 Material Calendar View

Material Calendar View é uma biblioteca disponibilizada por Erick C.⁴ para estilizar o componente de calendário padrão no Android. Esta biblioteca adiciona compatibilidade com o Material Design. Esta biblioteca possui eventos de cliques e gestos, que podem ser utilizados para proporcionar interação do usuário com o calendário.

D.5 Text Drawable

Text Drawable é uma biblioteca disponibilizada⁵ por Amulya Khare que permite criar ícones customizados para elementos dentro de uma lista. Esta biblioteca permite criar ícones arredondados, retangulares e retângulo com bordas arredondadas. Além destas formas, é possível aplicar cores de fundo que alteram dinamicamente através de um algoritmo gerador de cores.

D.6 Adobe Photoshop

Adobe Photoshop (<http://www.adobe.com/br/products/photoshop.html>) é um software de edição de imagem muito difundido. É uma aplicação comercial mantida pela Adobe desde 1990 e, neste projeto, uma versão para estudante (<https://goo.gl/6bGZi2>) foi adquirida. Além destas versões do Photoshop, uma versão de avaliação disponível por 30 dias pode ser baixada. Neste trabalho iremos utilizar o editor de imagem para criação da logomarca, diagramas comparativos, telas da aplicação e outras imagens ilustrativas.

D.7 Justinmind Prototyper

Justinmind Prototyper é um software⁶ utilizado na criação de protótipos de telas para diversas plataformas, como: Android, iPhone, Web e Desktop. Esta ferramenta possui recursos

³Disponível através do endereço <https://github.com/diogobernardino/WilliamChart>.

⁴Disponível através do endereço: <https://github.com/prolificinteractive/material-calendarview>.

⁵Disponível através do endereço: <https://github.com/amulyakhare/TextDrawable>.

⁶Disponível em: <http://www.justinmind.com/>.

interessantes, como criar um protótipo responsivo, ou seja, que se adapta a diferentes dispositivos, criação de *templates*, reproduzir gestos de utilização – como cliques – nas telas de protótipos e proporcionar interação entre telas, que pode ser configurada através de cliques e condições lógicas (como, por exemplo, abrir a tela inicial se login e senha do usuário forem válidos). Além disso, diversos exemplos são previamente disponibilizados para iniciar um projeto. Neste trabalho, uma versão gratuita deste software foi baixada em seu site oficial.

Apêndice E

Iterações de design

No decorrer do desenvolvimento da interface gráfica da aplicação Android, as telas foram alteradas algumas vezes para alinhar com o layout dos protótipos criados. Em algumas iterações foi necessária refatoração para adequar a interface gráfica com os requisitos do sistema. Neste apêndice será abordada a evolução da interface gráfica da aplicação desde os protótipos criados até o layout mais recente.



Figura E.1: Iterações de design da tela de configurações.

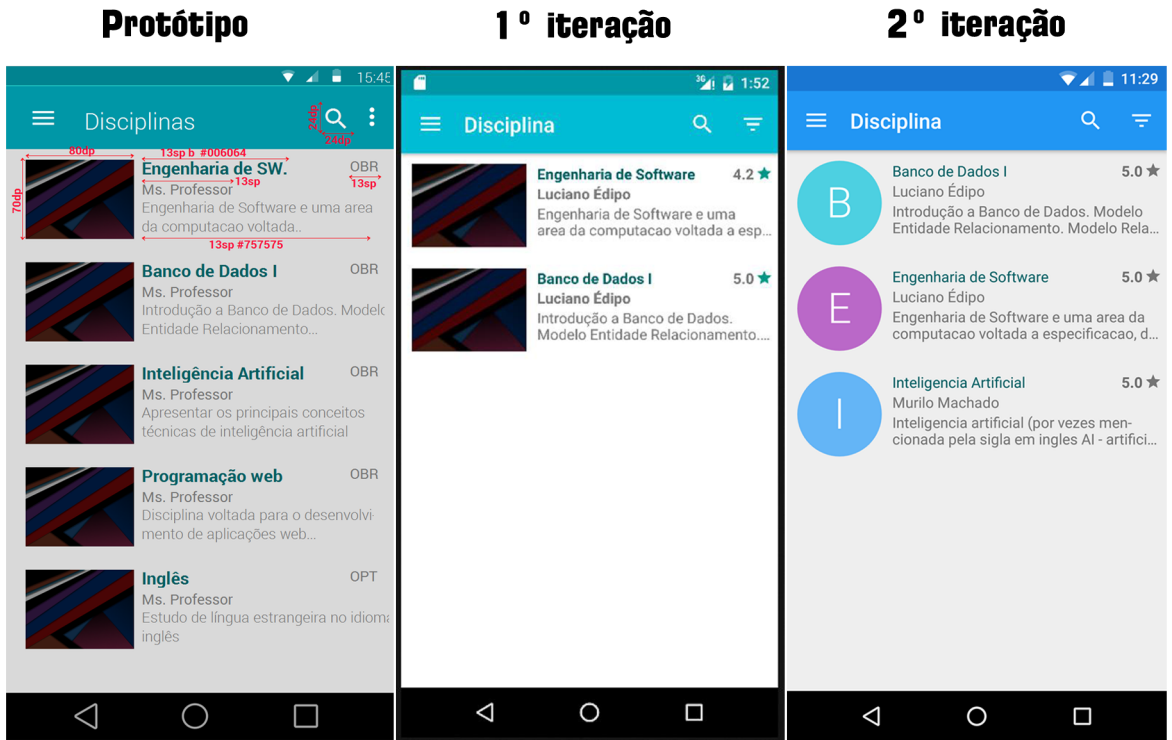


Figura E.2: Iterações de design da tela de lista de disciplinas.



Figura E.3: Iterações de design da tela de detalhes do evento.

Protótipo

1º iteração

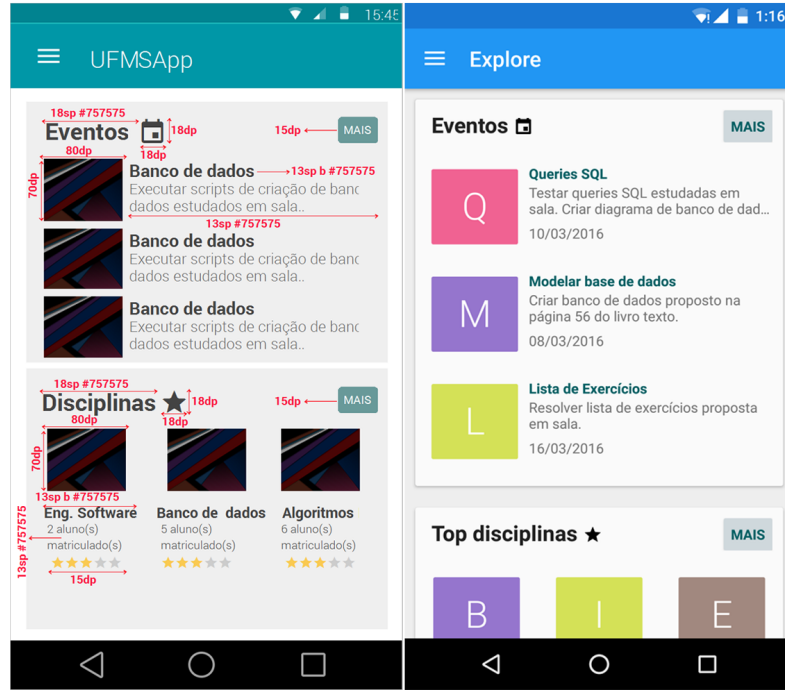


Figura E.4: Iterações de design da tela inicial (Explore).

Protótipo

1º iteração

2º iteração

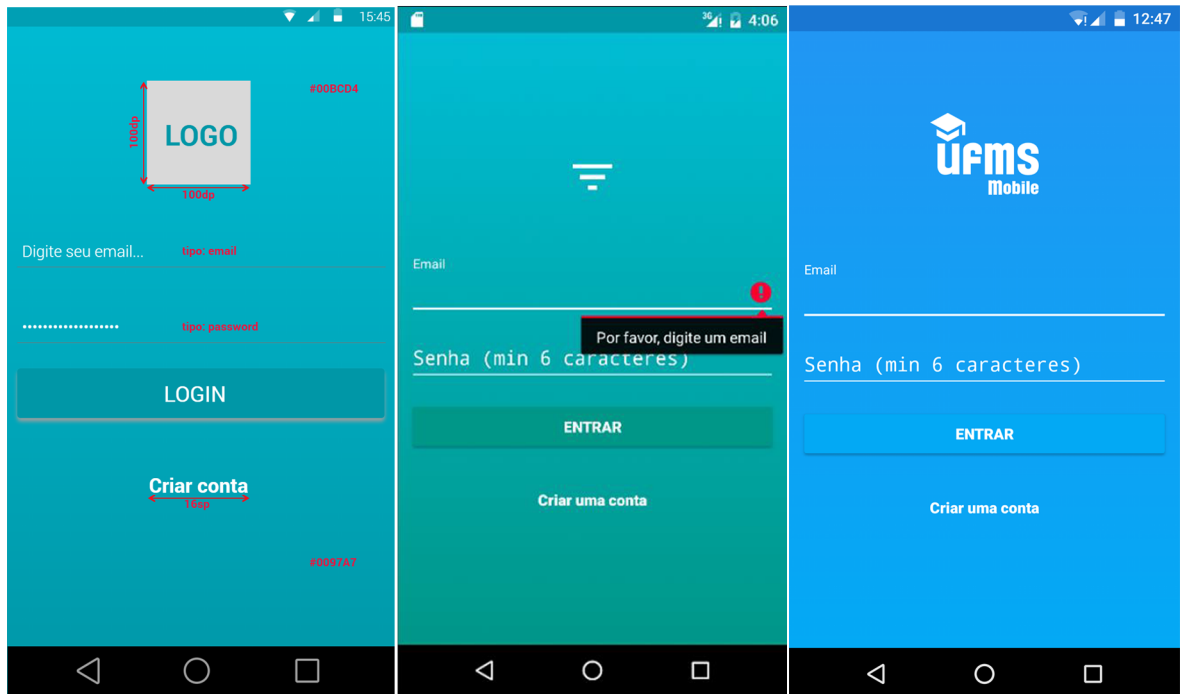


Figura E.5: Iterações de design da tela de acesso: login.

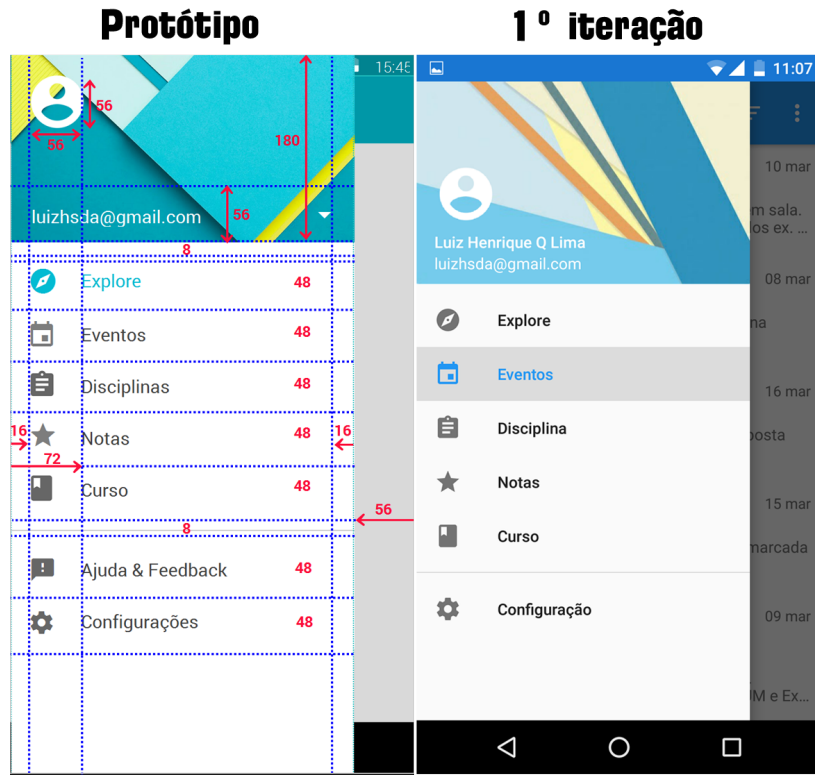


Figura E.6: Iterações de design do menu lateral da aplicação.