UNIVERSIDADE ESTADUAL PAULISTA

"JÚLIO DE MESQUITA FILHO"

CAMPUS UNIVERSITÁRIO DE BAURU

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

MARCOS VINÍCIUS ALVES SARTORI

FERRAMENTAS PARA O DESENVOLVIMENTO DE PROTÓTIPO DE LMS PARA DISPOSITIVOS MÓVEIS

UNESP – Bauru
Novembro de 2017

MARCOS VINÍCIUS ALVES SARTORI

FERRAMENTAS PARA O DESENVOLVIMENTO DE PROTÓTIPO DE LMS PARA DISPOSITIVOS MÓVEIS

Monografia apresentada junto à disciplina Projeto e Implementação de Sistemas II, do curso de Bacharelado em Ciência da Computação, Faculdade de Ciências, Unesp, campus de Bauru, como parte do Trabalho de Conclusão de Curso.

Orientadora: Prof. Dr. ª Andréa Carla Gonçalves Vianna.

UNESP – Bauru
Novembro de 2017

MARCOS VINÍCIUS ALVES SARTORI

FERRAMENTAS PARA O DESENVOLVIMENTO DE PROTÓTIPO DE LMS PARA DISPOSITIVOS MÓVEIS

Banca Examinadora

Orientadora: Prof. Dr. ª Andréa Carla Gonçalves Vianna
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Prof. Dr. João Pedro Albino.
Departamento de Computação
Faculdade de Ciências
UNESP – BAURU

Prof. Dr. ª Simone das Graças Domingues Prado.
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

UNESP – Bauru
Novembro de 2017

RESUMO

Com o aumento da disponibilidade de informação e seus novos meios de acesso, os estudantes encontram novas oportunidades para aprender e melhorar suas habilidades em cada vez menos tempo. As empresas, escolas e faculdades também se utilizam da tecnologia e da internet para diminuir custos e aumentar a produtividade, tornando possível o aprendizado em qualquer hora e lugar. Este trabalho visa coletar as ferramentas necessárias para a construção de um protótipo de um Sistema de Gerenciamento de Aprendizado (LMS) para dispositivos móveis *Android*, com o desenvolvimento baseado nas principais tecnologias da atualidade, com a intenção do protótipo final ser escalável para grandes grupos de alunos e professores.

Palavras-chave: Aprendizado, LMS, *Android*.

ABSTRACT

With increasing availability of information and their new means of access, students have found new opportunities to learn and improve their skills in less and less time. Businesses, schools, and colleges also use technology and the Internet to lower costs and increase productivity, making learning possible anytime, anywhere. This work aims to collect all the necessary tools for the construction of a prototype of a Learning Management System (LMS) for Android mobile devices, with the development based on the main current technologies, with the intention of the final prototype being scalable for large groups of students and teachers.

Palavras-chave: Learning, LMS, *Android*.

Lista de Figuras

Sumário

1. Introdução

O acesso à informação nunca foi tão fácil e rápido antes do advento da internet, que possibilitou qualquer pessoa obter uma gama de conteúdo cada vez maior através de um computador sem sair de casa. O crescente avanço de novas tecnologias tem gerado novos meios de acesso à informação, como os telefones inteligentes (*smartphones)*, que possuem uma tecnologia mais avançada, uma capacidade de processamento maior em comparação com os outros telefones celulares e um sistema operacional próprio. (BATISTA, 2011)

Com o aumento da disponibilidade de informação e seus novos meios de acesso, os estudantes encontraram novas oportunidades para aprenderem e melhorarem suas habilidades em cada vez menos tempo. As empresas, escolas e faculdades também se utilizam da tecnologia e da internet para diminuir custos e aumentar a produtividade, tornando possível o aprendizado em qualquer hora e em qualquer lugar.

Uma das novas tecnologias que as instituições de ensino passaram a utilizar são os Sistemas de Gerenciamento de Aprendizagem (*Learning Management Systems* – LMS), que consistem em uma plataforma eletrônica capaz de lançar e acompanhar cursos presenciais, gerenciar as aulas, bem como as atividades propostas, e manter um registro de todas essas interações (MCINTOSH, 2011).

Segundo Cavus (2010), em um LMS os estudantes seriam capazes de compartilhar recursos, realizarem provas através da internet, acessarem suas notas, colaborarem com outros estudantes de sua classe e também com seu instrutor, ter um acesso rápido e fácil a materiais do curso.

Existe uma variedade de LMS utilizados hoje em dia por escolas e faculdades. Os mais comuns são *Blackboard*, *Canvas* e *Moodle*. Todos eles foram desenvolvidos para computadores e notebooks. Porém os estudantes hoje preferem utilizar como principal ferramenta para suas interações digitais o *smartphone*.

Esses dispositivos se tornaram o portal global para tudo o que as pessoas precisam fazer digitalmente, como a troca de e-mails, mensagens, fotos, resolução de negócios, atividades sociais como um todo. O aprendizado não poderia ficar de fora.

O aprendizado, através de dispositivos móveis, tem atributos tecnológicos únicos que resultam em melhorias pedagógicas. Pea e Maldonado (2006) enunciam sete características de se ter um dispositivo móvel a mão em sala de aula: portabilidade, tamanho da tela pequena, o poder de processamento computacional, diversas redes de comunicação, uma ampla gama de aplicações, sincronização de dados entre computadores, e entrada de caneta do dispositivo.

Nas salas de aulas é natural encontrar alunos que desistiram de usar papel e caneta para copiar as anotações dos professores e, ao invés disso, se utilizam de *smartphones* para registrar o conteúdo, seja digitando, gravando vídeos ou fotografando a lousa. Isso acaba acontecendo com maior frequência entre alunos

dos cursos relacionados a área de informática onde eles têm mais contato com as novas tecnologias e por isso desenvolvem uma maior confiança em utilizá-los.

Uma pesquisa realizada em 2015 pela Ericsson apontava que 2,6 bilhões de pessoas possuem um *smartphone* e a previsão é que para 2020 esse número chegue em 6,1 bilhões, aproximadamente 70% da população mundial. Um indicativo de que conforme a sociedade for se acostumando com uma tela menor como a de um celular os outros dispositivos digitais serão deixados de lado (CRAIG, 2015).

Os estudos de Cavus (2010) mostraram que os estudantes gostam de usar os dispositivos móveis como ferramenta de aprendizado, por eles serem flexíveis e terem acesso a materiais de aprendizado a qualquer hora e em qualquer lugar. Zhang, Perris e Young (2005) dizem que flexibilidade de tempo e lugar são um grande avanço para cursos a distância.

Segundo Dahlstrom, Brooks e Bichsel (2014), a próxima geração de LMS, para atender as necessidades e expectativas de seus usuários deve ser amigável para dispositivos móveis, personalizável, intuitivo e projetado para o aprendizado dos estudantes.

1.1 Problema

Os LMS mais difundidos não foram feitos ou preparados para os dispositivos moveis como os *smartphones,* e a partir do momento em que tais aparelhos passam a constituir a preferência dos usuários em suas atividades digitais e de aprendizado, isso se torna um problema tanto para visualização da plataforma, como sua utilização.

1.2. Justificativa

O desenvolvimento de um protótipo de LMS para dispositivos móveis visa sanar as dificuldades das plataformas mais utilizadas apresentadas anteriormente a fim de agilizar e melhorar as interações de professores e alunos através da praticidade que os smartphones oferecem.

1.3 Objetivos

Listar todas as ferramentas necessárias para a criação de um protótipo de LMS para *smartphones* com o intuito de facilitar o acesso e modernizar a utilização da plataforma de aprendizado por seus usuários.

1.4 Método de Pesquisa

O desenvolvimento deste trabalho consistiu em apurar e definir as ferramentas para a construção de um protótipo LMS, bem como as principais funcionalidades que o compõe. O protótipo deve ser composto por um *web service* capaz de processar todas as requisições feitas pelo aplicativo do *smartphone.*

A modelagem do *web service* precisa ser estruturada cuidadosamente a fim de suprir todas as funcionalidades do protótipo e atender a qualquer plataforma que queira utilizá-lo. Precisa ser construído de forma que possíveis melhorias futuras sejam adicionadas sem dificuldades facilitando a adaptação de projetos maiores a partir do protótipo.

O aplicativo do *smartphone* precisa ser simples e objetivo a fim de facilitar a sua utilização pelos usuários finais.

2. Fundamentação Teórica

Este capítulo apresenta uma sucinta descrição dos conceitos e teorias pesquisados necessários para o entendimento e desenvolvimento do projeto.

2.1 Banco de Dados

Toda aplicação que exige armazenamento de dados precisa de uma estrutura que gerencie esses dados. Essas estruturas se dividem em arquivos e banco de dados. É de conhecimento geral que, para aplicações que movimentam muitas informações, banco de dados é a escolha certa, por ser uma estrutura mais robusta, e oferecer maior segurança e integridade.

Segundo Date (2004), “Um banco de dados é uma coleção de dados persistentes, usadas pelos sistemas de aplicação de uma determinada entidade”. Por persistentes, o autor pretende dizer que após os dados entrarem no banco, eles só saem por meio de uma requisição explicita do SGBD (Sistema de Gerenciamento de Banco de Dados), que consiste em um conjunto de *softwares* que ficam entre o banco de dados físico e os usuários do sistema.

Um banco de dados consiste de tabelas e as tabelas por sua vez consistem de atributos. Para pesquisas e manipulação de dados do banco é utilizada uma linguagem universal chamada SQL (*Structured Query Language* ou Linguagem de Consulta Estruturada).

2.1.1 MySQL

O MySQL possui mais de 10 milhões de instalações registradas (incluindo alguns líderes da indústria como Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia e Booking.com – fonte tirada do site *mysql.com*). É provavelmente o SGBD mais popular para servidores *web* nos dias de hoje (NIXON, 2014). Hoje é desenvolvido e distribuído pela *Oracle* e, seu sucesso se deve a algumas características. Segundo SUEHRING (2002):

- Funciona em várias plataformas;
- Possui um baixo TCO (Custo total de posse);
- Documentação intuitiva vasta;
- Baixo custo de processamento, rodando no mais básico dos *hardwares*;
- A licença do *software* é gratuita.

2.2 HTTP

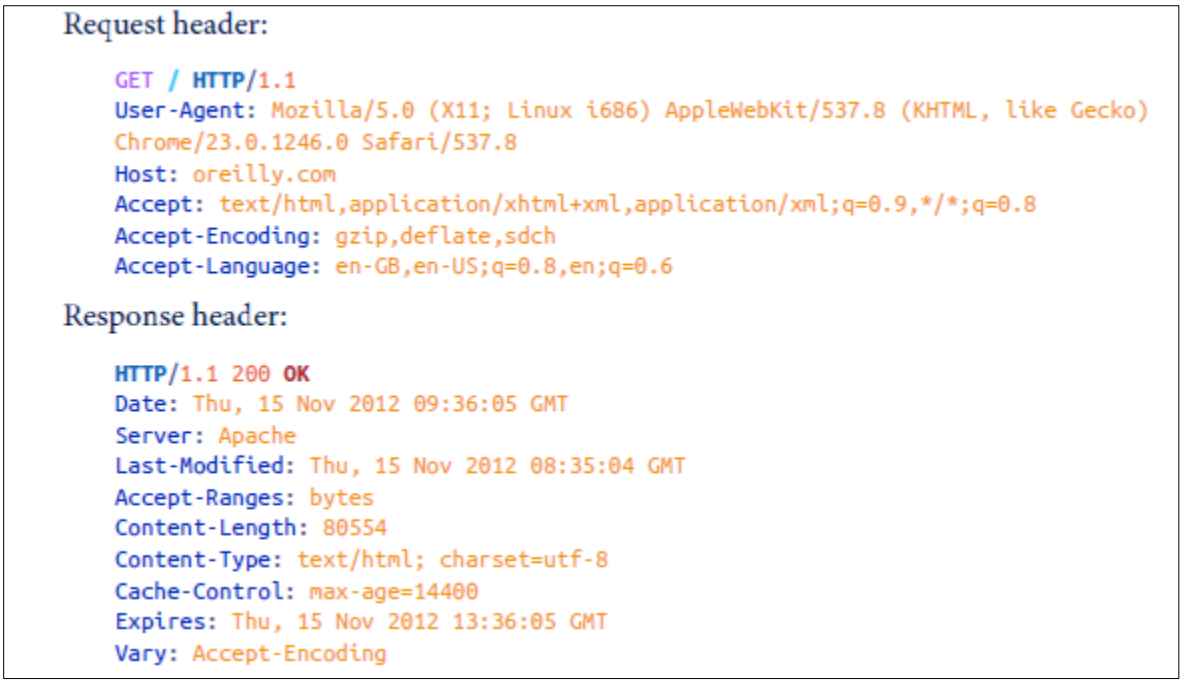
HTTP significa *Hyper Text Transfer Protocol* (Protocolo de Transferência de Hipertexto), que consiste em um protocolo de rede responsável pela transferência

de dados e pela comunicação entre cliente e servidor no ambiente da Internet, o qual permite a transferência de dados em hipermídia (texto, imagem e som).

Segundo Mitchell (2013), o HTTP é composto de muitas partes: o endereço de rede onde se encontra o recurso requisitado (URL), o método (ou verbo) usado, alguns cabeçalhos (*headers*), códigos de status (*status code)* e o corpo das respostas. Um usuário comum navega pela *web* acessando em *links* e abrindo imagens sem o conhecimento do que realmente acontece. Basicamente, a ação de acessar um *link* implica automaticamente em uma requisição/resposta. O cliente (navegador*)* faz uma requisição para o servidor e este manda uma resposta que é exibida pelo próprio navegador.

Quando um usuário faz uma requisição (http://google.com, por exemplo) visualiza somente o corpo da resposta do HTTP. Porém, como citado anteriormente, o protocolo é composto, também, por *headers*, *status code* e verbo. A Figura 1 ilustra as outras partes de uma suposta requisição HTTP.

Figura 1: Requisição HTTP.



Fonte: Mitchell (2013, p.2).

2.2.1 Verbos

O verbo (ou método) é uma parte muito importante do protocolo, pois é ele quem consegue transmitir as intenções da requisição. Eles se resumem em, basicamente, dois: *GET* e *POST* (apesar de existirem outros tipos, este trabalho foca apenas nos citados).

De acordo com Mitchell (2013), o método *GET* possui as seguintes particularidades:

- Suas requisições podem ser armazenadas no navegador;
- Permanecem no histórico do navegador;

- Pode ser "favoritada", pois as requisições feitas por esse método são enviadas via URL;
- Possuem restrições quanto ao tamanho da requisição;
- Nunca devem ser usadas para fornecer conteúdo.

Enquanto o método *POST* possui as seguintes características:

- Suas requisições nunca são cacheadas;
- Não ficam no histórico do navegador;
- Não pode ser feita de favorita, pois as requisições feitas por esse método são enviadas no corpo da requisição;
- Não possuem restrições de tamanho.

A partir das características apresentadas, é possível definir quando e onde é adequado usar o método *POST*, assim como para o método *GET*. Por exemplo, requisições de busca devem ser feitas via *GET*, pois o usuário tem a possibilidade de registrar como favorita aquela busca. Já o método *POST* pode ser usado para registrar um usuário, uma vez que é considerado um método mais seguro.

2.3 PHP

Segundo Dall'Oglio (2007) o PHP (*Personal Home Page Tools*) foi criado em 1994 por Rasmus Lerdorf como um conjunto de *scripts* em linguagem C feitos para monitorar o acesso ao seu currículo. Desde então, o PHP vem crescendo ao longo dos anos e adicionando mais e mais recursos, se consolidando como uma das linguagens de programação orientada a objetos que mais cresce no mundo.

O PHP é uma linguagem de programação de ampla utilização, interpretada, ou seja, o código não é compilado, o interpretador o executa diretamente, o que a torna especialmente interessante para desenvolvimento para a *web*. Seu objetivo principal é gerar páginas dinamicamente.

Uma das grandes vantagens do PHP é que ele é gratuito e é um *software* com código-fonte aberto, podendo ser obtido através do site *http://www.php.net* juntamente com sua documentação (Niederauer, 2009).

2.3.1 Lumen

Lumen é um *Framework* simples que usa um subconjunto dos mesmos componentes do Laravel *Framework*. Juntos Laravel e Lumen oferecem uma poderosa combinação de ferramentas: uma estrutura leve para escrever *API's* e uma estrutura *web* completa para aplicativos *web*. Lumen também tem um subconjunto de ferramentas de console disponíveis do Laravel. Outros recursos poderosos do Laravel estão incluídos como migrações do banco de dados, Modelos Eloquent (pacote ORM), filas de trabalho, trabalhos agendados e uma solução de teste focada (Redmond, 2016).

O *Framework* Lumen é um conjunto de bibliotecas que facilitam o desenvolvimento de API's em PHP e fornecem uma estrutura básica e simples para a criação de recursos e rotas para serem consumidos por outras aplicações,

no caso deste projeto, por um aplicativo *Android*. A Figura 2 ilustra a criação de um recurso para retornar todas as informações de um dado usuário.

Figura 2: Exemplo de recurso simples

Fonte: Documentação oficial do Lumen. (Disponível em https://lumen.laravel.com)

2.4 JSON

JSON (*JavaScript Object Notation* - Notação de Objetos JavaScript) é uma formatação de texto que facilita a troca de informações estruturadas entre todas as linguagens de programação. (ECMA, 2013)

O JSON é caracterizado com uma formatação leve de dados, pois é fácil para os seres humanos ler e escrever, e também é fácil para a máquina gerar e interpretar. Esta notação é baseada em um subconjunto da linguagem de programação JavaScript, porém é um formato de texto completamente independente de uma linguagem pois suas convenções são familiares às linguagens C, C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados ao invés de usar a antiga formatação XML.

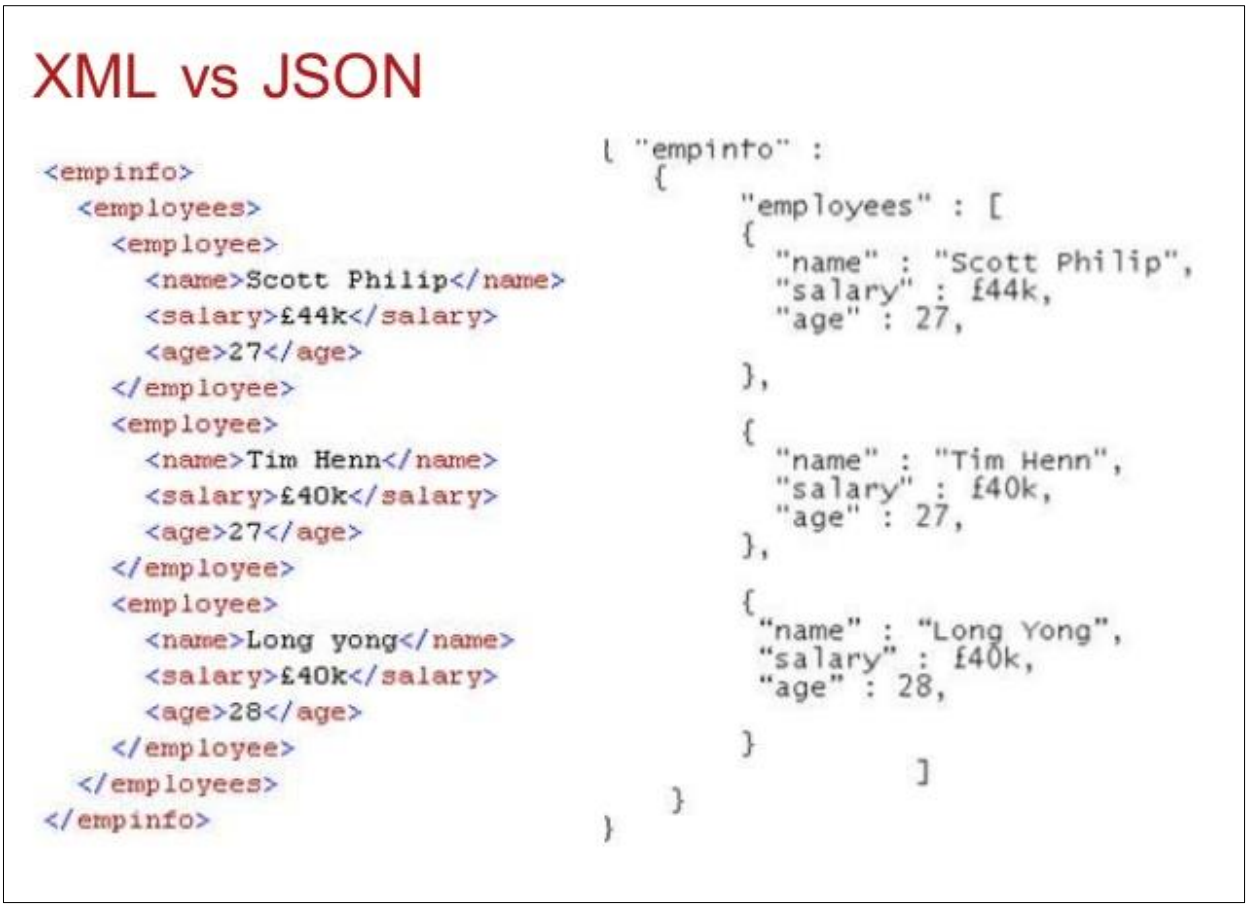
JSON é constituído por uma das duas estruturas a seguir:

- Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um objeto, *record*, *struct*, dicionário, *hash tabela*, *keyed* lista, ou *arrays* associativas.

- Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

A Figura 3 ilustra como é simples escrever e entender um conjunto de informações em JSON.

Figura 3: XML x JSON.



Fonte: Elaborada pelo autor.

2.5 Vagrant

Segundo Hashimoto (2013), Vagrant é uma ferramenta para construir completamente um ambiente de desenvolvimento portátil, de fácil configuração, reprodutível, construído com alta tecnologia e controlado por um único fluxo consistente para ajudar a maximizar produtividade e flexibilidade.

Após a configuração inicial, é possível:

- Criar uma máquina virtual com o sistema operacional a escolha;
- Modificar as propriedades físicas da máquina virtual;
- Estabelecer uma rede para poder acessar a máquina virtual de seu próprio computador, ou qualquer outro dispositivo na rede;
- Compartilhar diretórios entre os computadores da rede possibilitando a edição dos mesmos e essas alterações refletirem na máquina virtual;
- Configurar os *hostname* da máquina virtual, visto que muitas aplicações dependem destas configurações serem feitas corretamente;

A utilização da ferramenta Vagrant permite a unificação de todas as configurações do projeto em uma única máquina virtual, tornando possível a replicação do ambiente de desenvolvimento para futuros desenvolvedores, e assim ajudando a garantir a escalabilidade do projeto.

2.5.1 Homestead

Segundo o LARAVEL.COM, a Laravel Homestead é uma Vagrant Box oficial, pré-configurada, que oferece um ambiente de desenvolvimento completo sem exigir nenhuma instalação de PHP, um servidor *web* e qualquer outro *software* de servidor na sua máquina local.

Alguns *softwares* incluídos na Laravel Homestead importantes para o desenvolvimento deste projeto:

- Ubuntu 16.04
- Git
- PHP 7.1
- MySQL
- Composer

De acordo com Redmond (2016), a Laravel Homestead é a melhor escolha como ambiente de desenvolvimento para aplicações Lumen por fornecer um ambiente completo com tudo que é preciso para o desenvolvimento do projeto. A Homestead provém alguns benefícios sólidos como ambiente de desenvolvimento:

- Ambiente isolado em uma máquina virtual
- Funciona em Windows, Mac e Linux
- Fácil de configurar todos os projetos em um único lugar

2.6 Composer

Segundo GETCOMPOSER.ORG, Composer é uma ferramenta para gerenciar as dependências em PHP. Ele permite, a partir da declaração das dependências do projeto, o gerenciamento delas (instalar/atualizar).

O Composer não é um gerenciador de pacotes da mesma maneira que o Yum ou Apt. Apesar de cuidar de pacotes e bibliotecas, o Composer os gerencia em uma escala menor a do projeto a qual foi designado. Ele instala as dependências em um diretório reservado (*vendor*) dentro do projeto. Por padrão, ele não instala nada globalmente na máquina, porém ele é totalmente capaz de fazê-lo através de um comando global.

Através do Composer é possível gerenciar e documentar todas as dependências, bibliotecas e *plug-ins* do projeto e mantê-las atualizadas ou em uma versão estável a fim de manter todas funcionando em conjunto.

2.7 *Android*

Segundo Pereira e Silva (2009), “O *Android* é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middleware*, aplicativos e interface do usuário”. Ele é o primeiro projeto de uma plataforma *open source* para dispositivos móveis em conjunto com a *Open Handset Alliance* (OHA). Ele foi criado com a finalidade

de fornecer aos desenvolvedores as ferramentas necessárias para explorar todas as funções do celular. Por exemplo, uma aplicação pode ter uma funcionalidade para fazer chamadas, ou utilizar a câmera, ou mandar mensagens, ou utilizar o GPS, etc.

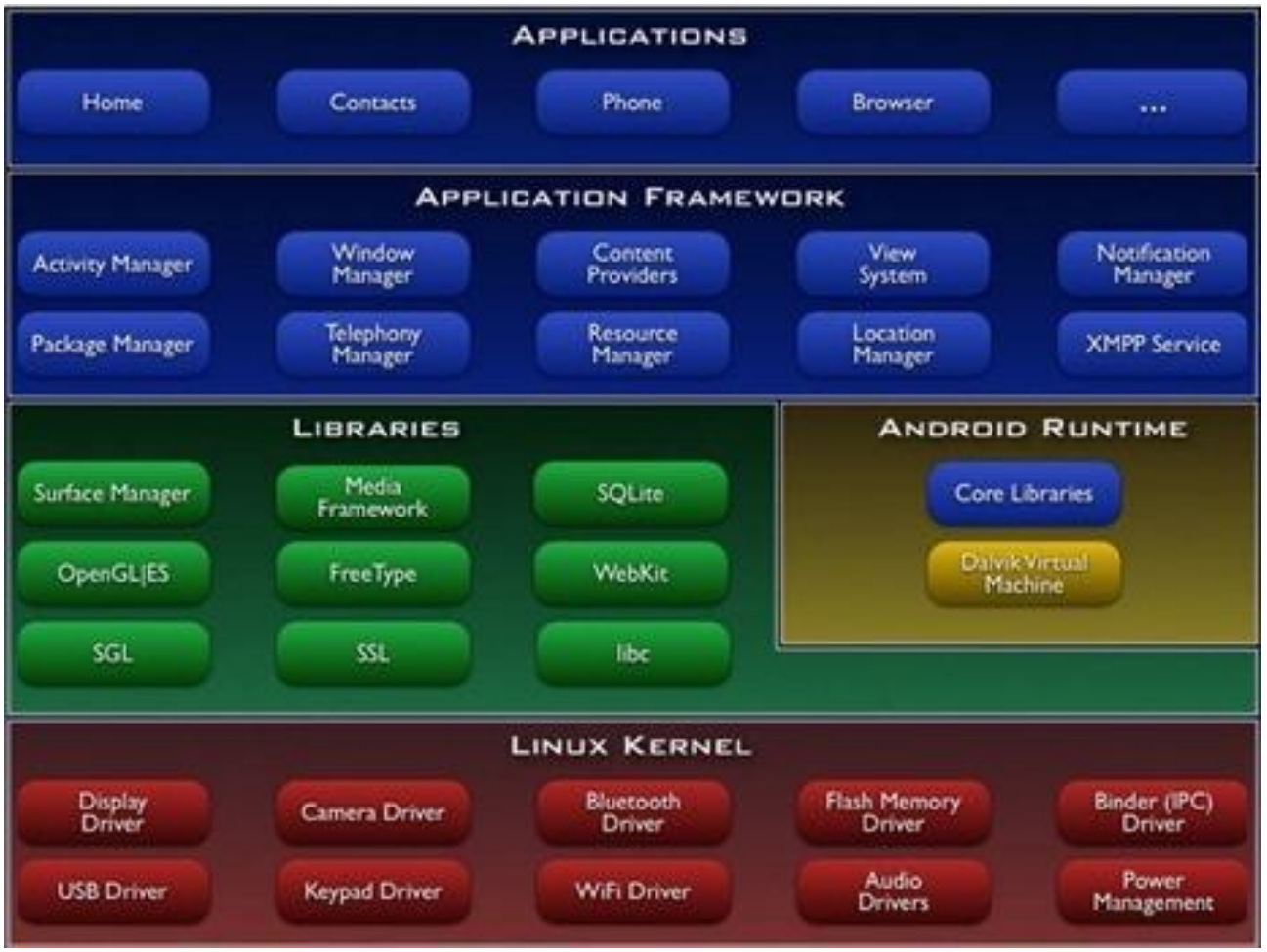
Por ser *open source*, e desenvolvido com base no sistema operacional Linux, ele pode ser sempre adaptado e melhorado de acordo com a ascensão de novas tecnologias e a contribuição da comunidade, ou seja, sempre estará em evolução.

Apesar de ser *open source*, o *Android* tem uma capacidade de segurança muito forte. Como é executado em um núcleo Linux, toda vez que um aplicativo é instalado, cria-se um novo usuário Linux especialmente para aquele programa, com diretórios que serão usados pelo aplicativo, mas somente para aquele usuário Linux. Como os aplicativos ficam isolados uns aos outros, qualquer tentativa de acesso a informações de outro aplicativo é barrada na negação (ou não) do usuário. Caso uma aplicação faça uso do GPS, por exemplo, é preciso fazer uma solicitação de permissão ao usuário para o uso de tal recurso.

2.7.1 Arquitetura

O *Android* possui uma arquitetura dividida em cinco camadas: Aplicações, *Framework* de Aplicação, Bibliotecas, *Android Runtime* e Linux *Kernel*, conforme ilustrado na Figura 4.

Figura 4: Arquitetura *Android.*



Fonte: PEREIRA; SILVA (2009, p.5).

A seguir é explicado brevemente o papel de cada camada.

a) Aplicações

O *Android* inclui em seu sistema operacional alguns aplicativos que interagem com o celular e dão a funcionalidade básica do sistema. Entre eles estão navegadores, programa de SMS, gerenciador de contatos, agenda, calendários, etc. (PEREIRA; SILVA, 2009).

b) *Framework* de Aplicação

Framework é um conjunto de classes que incorporam um *design* abstrato a fim de prover uma solução para uma família de problemas semelhantes. Sendo assim, o *Framework* de Aplicação do *Android* contém todas as APIs (Interface de Programação de Aplicativos) e recursos utilizados pelos aplicativos, as chamadas classes visuais (listas, grades, caixas de texto).

Possui também o provedor de conteúdo (*Content Provider*), que faz com que uma aplicação possa acessar ou até mesmo compartilhar informações de outra, gerenciador de localização e notificação, etc (PEREIRA; SILVA, 2009).

c) Bibliotecas

A camada de bibliotecas possui um conjunto de bibliotecas C/C++ que acabam por poupar trabalho do desenvolvedor. Algumas rotinas fundamentais como abertura de arquivo, exibição de mensagens, cálculo aritmético já estão implementadas. Possui também bibliotecas das áreas de multimídia, funções para gráficos, funções para navegadores *web* e muitos outros utilitários (PEREIRA; SILVA, 2009).

d) *Android Runtime*

Assim como o Java, o *Android* possui uma máquina virtual. Esta chamada de *Dalvik Virtual Machine* (DVM), que é responsável por executar todos os aplicativos no celular. Por se tratar de dispositivos móveis, cujo desempenho era extremamente limitado se comparado a computadores pessoais, ela precisou se adaptar a tais limitações, o que acabou tornando-a muito eficiente, pois consegue trabalhar em sistemas com baixa frequência de CPU (processador) e pouca memória RAM disponível. Além disso, a *Dalvik* é otimizada para consumir pouca bateria.

O código do aplicativo é compilado em um .*class*, depois transformado em arquivo binário (.dex) e interpretado pela Dalvik. Quem faz essa transformação é uma ferramenta chamada DX disponível no SDK do *Android* (Kit de Desenvolvimento de Software). É importante apontar que é exatamente por isso que os aplicativos podem ser executados em qualquer dispositivo *Android*, independentemente da arquitetura do seu processador (PEREIRA; SILVA, 2009).

e) Linux *Kernel*

Utiliza a versão 2.6 do *kernel* (núcleo) do Linux. O núcleo de um SO (Sistema Operacional) atua como uma camada de abstração entre o *hardware* e as camadas seguintes da arquitetura. É ele que gerencia os serviços fundamentais do sistema como segurança, gestão de memória, gestão de processos, modelo de *drivers* e pilha de protocolos de rede. Um recurso interessante encontrado nesse *kernel* é o *Binder*. Este consiste num mecanismo de comunicação de processos (PEREIRA; SILVA, 2009).

2.7.2 Requerimentos para Desenvolvimento

Como é de conhecimento geral, as aplicações *Android* são feitas em cima da plataforma Java, consequentemente, se faz necessária a instalação dela, do

JDK (*Java Development Kit*), que contém o compilador, *debugger* e outras ferramentas para o desenvolvimento do software, bem como a JRE (*Java Runtime Environment*), responsável por executar tais ferramentas.

Atualmente, a fim de otimizar o tempo do desenvolvedor com instalações e reduzir as chances de erros, a Google disponibiliza o *Android Development Tools – Bundle* (ADT). Como o próprio nome diz, ele é um pacote que contém os componentes essenciais do SDK (*Software Development Kit* - será explicado mais detalhadamente em capítulos seguintes); a IDE Eclipse; a última plataforma *Android* etc. Nota-se que antes do ADT *Bundle*, todos os componentes eram instalados manualmente.

Depois de instalar o ADT corretamente, basta configurar um aparelho *Android* para testes futuros (ou um emulador, que não será o foco deste trabalho) e o ambiente estará pronto para desenvolver.

2.7.3 Configurando um Aparelho para Desenvolver

A grande maioria dos aparelhos *Android* podem ser usados para desenvolvimento. Para habilitá-los, é necessário acessar a área desenvolvedor/programador (o local específico onde tal área se encontra depende da versão do SO, porém ela normalmente se encontra dentro de Configurações) e habilitar a opção Depuração USB (USB *Debugging*). Em seguida, é necessária a instalação do *driver* do ADB (*Android Debug Bridge*) que consiste em uma ferramenta a qual permite a comunicação com um emulador ou aparelho *Android*. Trata-se de um programa que inclui três componentes: cliente, que executa na máquina de desenvolvimento; servidor, que é executado como um processo em segundo plano na máquina de desenvolvimento; *daemon*, que roda como um processo em segundo plano no aparelho ou emulador.

2.7.4 SDK

Como definido anteriormente, o SDK (*Software Development Kit ou ainda* Kit de Desenvolvimento de *Software*) é essencial para o desenvolvimento, é composto por diversos componentes de *software* agregados. Normalmente inclui várias bibliotecas, documentação da API, códigos de amostra, IDE, etc.

O *Android* SDK tem uma particularidade por possuir uma estrutura modular, ou seja, é composto de pacotes. Isso implica em benefícios, uma vez que o programador pode instalar apenas os componentes necessários para sua aplicação, desperdiçando menos espaço. É dividido em:

- SDK *Tools*: ferramentas para *debug* e testes mais outros utilitários requeridos na construção de um aplicativo.
- *Platform Tools*: ferramentas adicionais que são atualizadas conforme a versão do *Android*.
- SDK *Platform*: há uma SDK *Platform* para cada versão do SO inclui um *android.jar* que contém uma biblioteca *Android*.

- *Google* APIs: bibliotecas necessárias para utilizar os recursos exclusivos do *Google*, i.e, *Google Cloud Messaging* ou *Maps*.
- *Samples and Documentation*: amostras de projetos com suas respectivas documentações para cada versão da plataforma (PEREIRA; SILVA, 2009).

2.7.5 *User Interface – Layout*

Assim como em páginas *web*, a interface com o usuário em um aplicativo é determinante para sucesso ou fracasso do mesmo. O layout de uma aplicação *Android* é criado a partir de um mecanismo simples e intuitivo: arquivos XML (*Extensible Markup Language*).

Por serem separados do código da aplicação, é permitido ao desenvolvedor mudar o *layout* sem ter que recompilar ou editar a lógica do programa. Ele é feito através da hierarquia entre dois objetos:

- *View*: normalmente são os *widgets* (caixa de texto, botões).
- *ViewGroups*: objetos invisíveis (*containers*, *forms*) que definem como os objetos filhos (*Views*) serão exibidos na tela (PEREIRA; SILVA, 2009).

A Figura 5 mostra um arquivo XML básico para uma aplicação *Android*. É possível notar a hierarquia entre os objetos e a facilidade de entendimento.

Figura 5: Layout para Uma Aplicação *Android.*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Fonte: Documentação oficial do *Android*. (Disponível em http://developer.android.com/guide/topics/ui/declaring-layout.html).

2.7.6 *Activity*

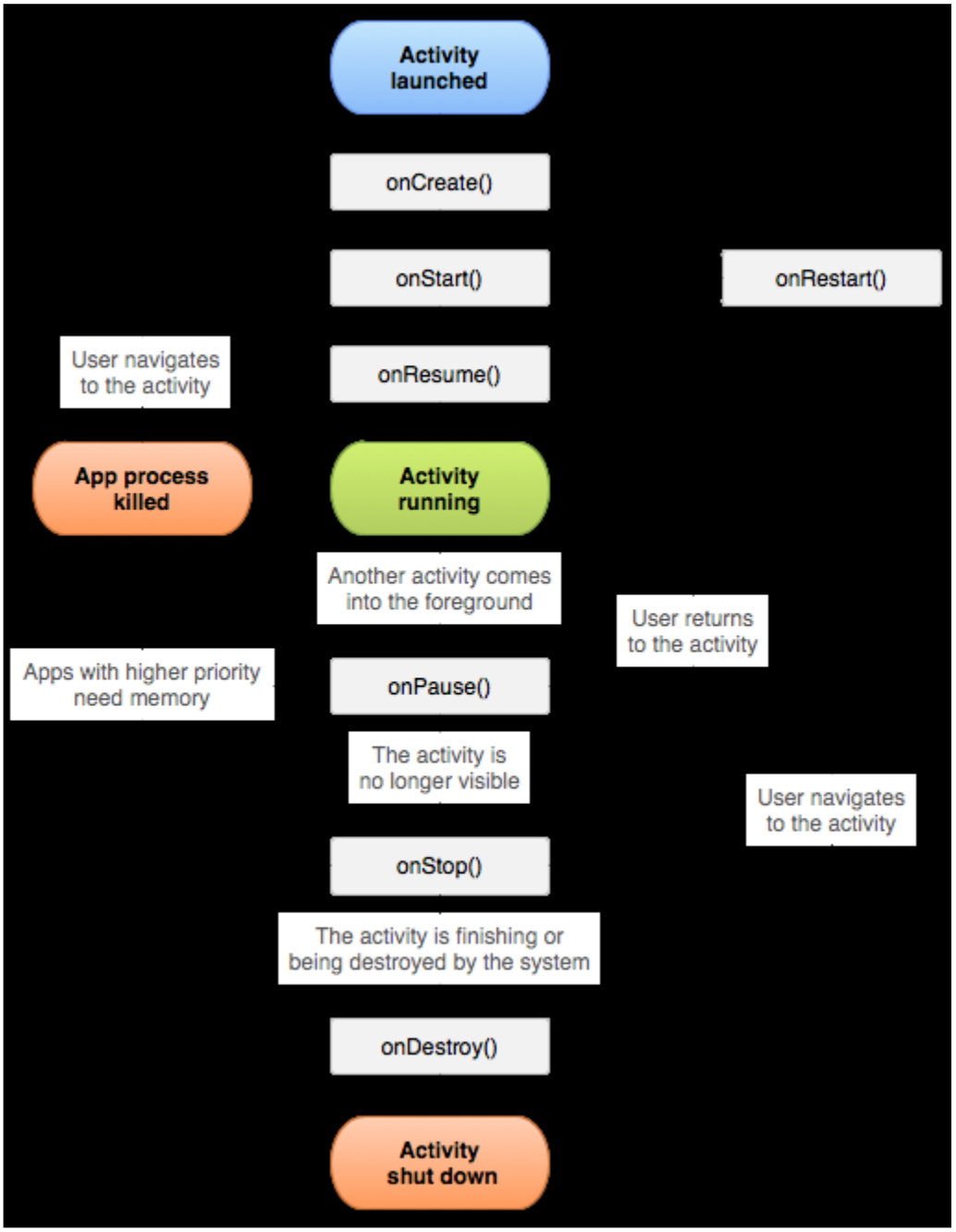
Segundo o Android Developers (2015), *Activity* é o componente principal de uma aplicação. Ela é composta de uma tela onde é desenhada uma interface. O

usuário pode interagir com essa interface através de ações como, por exemplo, enviar email, tirar foto, visualizar mapa, etc. O fluxo de uma aplicação *Android* começa por uma *activity.* A fim de criar uma *activity*, é necessário criar uma subclasse da classe *Activity* e implementar alguns métodos. Tais métodos compõem o ciclo de vida da *activity*. Sabendo disso, alguns deles se tornam obrigatórios dependendo do comportamento da *activity* no sistema. São eles:

- *onCreate*: esse método é o único que deve ser implementado em todas as ocasiões, pois é chamado quando a *activity* está sendo criada. Para se ter uma idéia de sua importância, é nele que é definido qual *layout* será utilizado para a *activity* em questão;
- *onRestart*: chamado quando a *activity* foi pausada e está prestes a ser chamada novamente. Esse método é sempre precedido pelo *onStart;*
- *onStart*: chamado quando a *activity* se torna visível para o usuário;
- *onResume*: chamado quando a *activity* está prestes a se interagir com o usuário;
- *onPause*: chamado quando o sistema está prestes a retornar a *activity* anterior;
- *onStop*: chamado quando a *activity* não está mais visível para o usuário;
- *onDestroy*: chamado antes da *activity* ser destruída. A razão da chamada pode ser ou liberação de espaço ou término da mesma.

A Figura 6 ilustra o ciclo de vida de uma *activity* e a interação entre os métodos.

Figura 6: *Activity Lifecycle.*



Fonte: Documentação oficial do *Android*. (Disponível em http://developer.android.com/guide/components/activities.html).

2.7.7 *Fragment*

De acordo com o *Android Developers* (2015), os *Fragments* representam um comportamento ou um pedaço de interface de uma *activity*. É possível combinar vários *fragments* dentro de uma *activity*. O intuito em usá-los no sistema se deve ao fato de serem reutilizáveis, ou seja, um mesmo *fragment* pode ser incorporado em várias *activities*. É importante notar que, apesar de ter um ciclo de vida proprio (muito similar ao de uma *activity*, com métodos de chamada do tipo *onCreate, onPause, onStart,* etc), um *fragment* depende totalmente do ciclo de

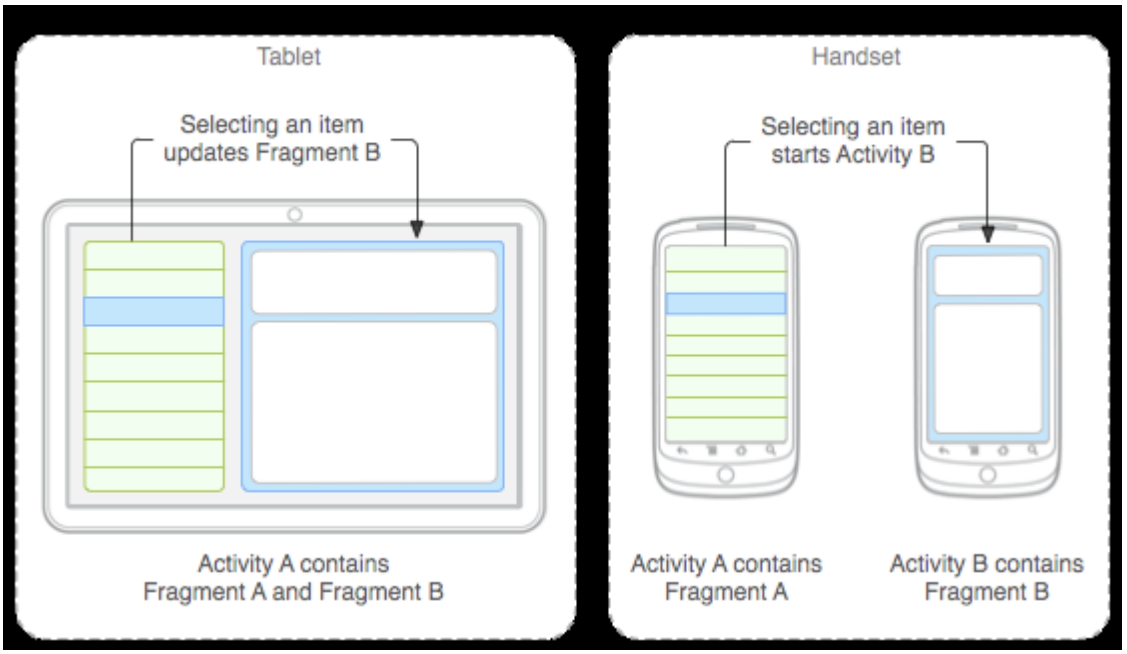
vida de sua respectiva *activity*. Por exemplo, se uma *activity* é pausada, todos os *fragments* relacionados a ela serão pausados também.

Esse componente foi introduzido na versão 3.0 para suportar interfaces mais flexíveis e dinâmicas em dispositivos com telas grandes, como os *tablets*. Os *fragments* permitem fazer o *design* de telas, sem a necessidade de mudanças complexas no código. Ao dividir o *layout* de uma *activity* em vários desses componentes, fica muito mais fácil e dinâmico modificar sua aparência.

Em suma, os *fragments* devem ser projetados de forma modular e reutilizável, possibilitando o desenvolvimento de aplicativos com suporte para vários *layouts* diferentes, ou seja, o posicionamento desses elementos é diferente de acordo com o tamanho da tela do dispositivo.

A Figura 7 representa um exemplo de uso dos *fragments*. O *tablet*, por possuir uma tela maior, comporta os dois *fragments* numa única *activity.* Já o *handset*, por possuir uma tela com tamanho reduzido, necessita de duas *activities* para mostrar os dois *fragments* em questão.

Figura 7: Exemplo de Utilização dos *Fragments.*

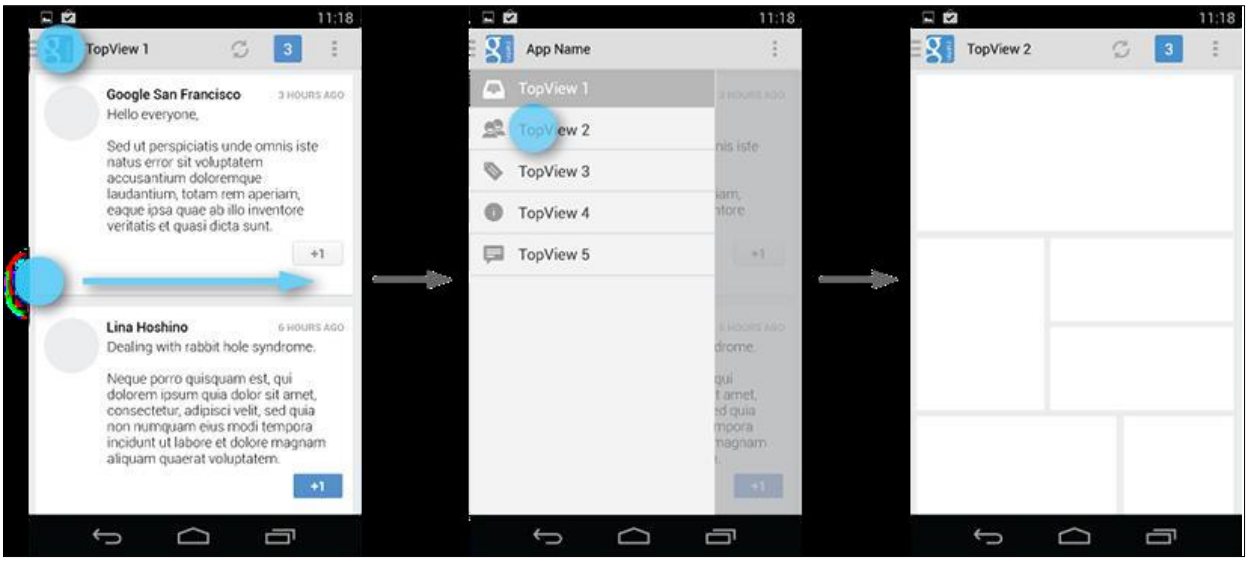


Fonte: Documentação oficial do *Android.* (Disponível em http://developer.android.com/guide/components/fragments.html).

2.7.8 *Navigation Drawer*

É um tipo de menu muito utilizado nos dias de hoje. Ele é representado por três traços horizontais. Acaba por poupar muito espaço da tela, uma vez que inicialmente fica escondido e só é acionado quando o usuário toca no ícone ou o arrasta da lateral da tela. De acordo com o *Android Developers* (2015), é recomendado quando o menu contém mais de três itens. A Figura 8 demonstra um exemplo de *Navigation Drawer.*

Figura 8: Funcionamento do *Navigation Drawer.*



Fonte: Documentação oficial do Android. (Disponível em https://developer.android.com/design/patterns/navigation-drawer.html)

É importante notar que esse componente faz uso dos *fragments* em sua construção. Basicamente, a tela em si (primeira imagem da figura 8) é uma *activity* que contém um *layout*, o menu (inicialmente escondido) é um *fragment* dessa *activity* e, ao acessar um dos itens do menu (segunda imagem), a aplicação procura pelo *fragment* correspondente (terceira imagem). Ou seja, trata-se de uma única *activity* que varia de *fragment* de acordo com a escolha do usuário.

3 Materiais e Métodos

Este capítulo consiste de uma breve introdução dos materiais e tecnologias que podem ser utilizadas para o desenvolvimento de um protótipo de LMS para dispositivos móveis, bem como as justificativas das escolhas.

3.1 e-Learning

Com aparecimento das Tecnologias de Informação e Comunicação (TICs), surgiram novas perspectivas em relação a forma e ao alcance da Educação a Distância (EaD). No Brasil, o aprendizado eletrônico, ou em inglês *Electronic Learning* (*e-Learning*), parece ser um caminho sem volta. Alonso (2010), com base nos dados do Anuário Brasileiro Estatístico de EaD, afirma que em 2008 mais de 2,5 milhões de brasileiros realizavam cursos na modalidade à distância.

Nos últimos anos presenciamos um considerável aumento da oferta de *e-Learning* no Brasil e, mais recentemente com a venda em larga escala de uma rica variedade de dispositivos móveis como, por exemplo, celulares, *smartphones* e *tablets*; a Educação não pode fechar seus olhos a esse movimento educacional e tecnológico. Neste contexto uma das formas recentes de potencializar o *e-Learning* é através do uso de dispositivos móveis, modalidade conhecida como aprendizagem móvel ou, em inglês, *mobile learning* (*m-Learning*). Segundo Yau e Joy (2010), este meio de oferecer ensino permite que estudantes e professores possam tirar vantagens dos recursos oferecidos pelas tecnologias móveis. Os autores destacam a possibilidade de acessar, visualizar e prover conteúdo independentemente da hora e, principalmente, do lugar.

3.2 *m-Learning*

Cavus (2011) descreve a aprendizagem móvel (*m-learning*) como um tipo de modelo de aprendizagem que permite que os alunos obtenham materiais de aprendizagem em qualquer lugar e a qualquer momento usando tecnologias móveis e a Internet. Ainda, conforme Cavus (2011), a aprendizagem móvel aumenta e proporciona sentimentos de liberdade aos estudantes, e é utilizado para aumentar o desempenho dos alunos, tornando a aprendizagem acessível. As tecnologias de aprendizagem móvel eliminam limites geográficos e fornecem um ambiente de aprendizagem colaborador entre grupos estrangeiros.

3.3 *LMS*

Segundo Turan (2010), uma das tecnologias modernas é o Sistema de Gerenciamento de Aprendizado (LMS), que é um aplicativo de *software* a ser usado para integrar recursos tecnológicos e pedagógicos em um ambiente virtual bem desenvolvido de aprendizagem.

Como mencionado anteriormente, em um *LMS* os alunos podem compartilhar recursos, fazer testes *on-line*, acessar suas notas e fazer *upload* de tarefas, colaborar com colegas e instrutores, e ter acesso fácil aos materiais do curso. Além do que, os *LMS's* permitem que os instrutores criem e suportem um ambiente de aprendizagem dinâmico.

4 Implementação

Com base no que foi apresentado anteriormente, o protótipo tem como objetivo suprir as necessidades básicas de um *LMS.* Portanto ele deve ser capaz de:

- Cadastrar usuários (alunos ou orientadores);
- Cada usuário (orientador) tem a possibilidade de criar um grupo de estudo e convidar outros usuários (alunos);
- O administrador do grupo pode criar aulas com conteúdo diverso;
- O administrador pode criar atividades para serem feitas;
- O administrador pode avaliar as respostas das atividades de cada aluno;
- Os alunos devem ter acesso as aulas criadas pelo administrador, bem como, poderem dar uma resposta para cada atividade proposta;
- Os alunos acessam as notas após a avaliação do administrador.

A implementação do protótipo deve ser dividida em duas partes compostas por (1) *Web* Service e (2) Aplicativo *Android*.

4.1 *Web service*

O *web service,* desenvolvido em PHP, com ajuda das bibliotecas do *Lumen Framework*, deve conter todos os recursos que a aplicação *Android* precisar acessar e retornar as informações necessárias para construir suas interfaces e atender as requisições dos usuários. As informações dos grupos criados, bem como as atividades e notas, seriam todas armazenadas no banco de dados desenvolvido em MySQL.

4.2 Aplicativo *Android*

O aplicativo é a *interface* utilizada pelo usuário final, desenvolvido para dispositivos móveis *Android*. A aplicação consome os recursos do *web* service tanto para receber as informações que precisa exibir como o conteúdo de uma aula criada no grupo ou as notas de uma atividade, quanto para processar alguma ação realizada pelo usuário como criar um grupo novo ou responder a uma atividade.

5 Conclusão

Com base em todas as ferramentas que foram apresentadas tanto para a construção do *web* sevice quanto para o aplicativo *mobile* e os pontos levantados sobre um LMS, seus benefícios e funcionalidades principais, e também levando em consideração que o objetivo deste trabalho é listar as ferramentas necessárias, bem como descrever e orientar como é a construção de um LMS para dispositivos móveis *Android* e sendo assim é possível dizer que este trabalho cumpre com seus objetivos principais.

Espera-se que com este trabalho seja possível ajudar a comunidade na construção de aplicações *Android,* e também enumerar os benefícios de um LMS como *m-Learning* para os alunos e orientadores que venham a utilizar ou precisar de um sistema assim.

REFERÊNCIAS BIBLIOGRÁFICAS

ALONSO, K. M. (2010) “A Expansão do Ensino Superior no Brasil e a EaD: Dinâmicas e Lugares”. In: **Educação & Sociedade**, v. 31, n. 113. p. 1319-1335.

ANDROID DEVELOPERS. ***App Components*.** 2015. Disponível em: <http://developer.android.com/guide/components/index.html>. Acesso em: 26/02/2017.

BATISTA, G. **Saiba tudo sobre Smartphones**. Artigo Disponível em: <http://www.artigonal.com/telefonia-e-celular-artigos/saiba-tudo-sobre-ossmartphones-4601618.html>. Acesso em: 10 de Abril de 2017.

CAVUS, N. **Investigating mobile devices and LMS integration in higher education: Student perspectives.** 2010. Disponível em: <www.sciencedirect.com>. Acesso em: 05 de Maio de 2017.

CAVUS, N. **Basic elements and characteristics of mobile learning.** 2011. Disponível em: <www.sciencedirect.com>. Acesso em: 05 de Maio de 2017.

CRAIG, R. **Smartphones and the decline and fall of LMS**. 2015. Disponível em: <https://www.higheredjobs.com/blog/postDisplay.cfm?post=730>. Acesso em: 05 de Abril de 2017.

DAHLSTROM, E.; BROOKS, C.; BICHSEL, J. **The Current Ecosystem of Learning Management Systems in Higher Education: Student, Faculty, and IT Perspectives**. 2014. Artigo Disponível em: <https://net.educause.edu/ir/library/pdf/ers1414.pdf>. Acesso em: 10 de Abril de 2017.

DALL’OGLIO, P. **PHP: Programando com Orientação a Objetos**. 3ª Edição. São Paulo: Novatech Editora Ltda, 2007.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. Elsivier Editora Ltda, 2004. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=xBeO9LSlK7UC&oi=fnd&pg=PP23&dq=banco+de+dados+cliente+servidor&ots=xaLxeXDeaG&sig=1qXp_sObMObhagsnW0E5A7IlQtY#v=onepage&q&f=false>. Acesso em 03 de Outubro de 2017.

ECMA INTERNATIONAL. **Standard ECMA-404**: The JSON Data Interchange Fornat. 1ª Edição. Genebra, 2013.

GETCOMPOSER.ORG, **Getting Started**. Disponível em: <https://getcomposer.org/doc/00-intro.md>. Acesso em 10 de Maio de 2017.

HASHIMOTO, M. Vagrant: Up and Running. 1ª Edição. Sebastopol: O'Relly, 2013.

LARAVEL.COM, **Laravel Homestead.** Disponível em: <https://laravel.com/docs/5.5/homestead>. Acesso em: 10/11/2017

MCINTOSH, D. **Learning Management Systems**. Artigo Disponível em: <http://www.trimeritus.com/bookchapter.pdf >. Acesso em: 30 de Março de 2017.

MITCHELL, L. J. . **PHP *Web* Services.** O'Reilly, 2013.

NIEDERAUER, J. **Desenvolvendo *Web*sites com PHP**. 2009. 2ª Edição. São Paulo: Novatec Editora Ltda.

NIXON, R. **Learning PHP, MySQL, JavaScript, CSS & HTML5**. O'Reilly Media Inc, 2014.

PEA, R.; MALDONADO, H. (2006). **WILD for learning: Interacting through new computing devices anytime, anywhere**. In R. K. Sawyer (Ed.), The Cambridge handbook of the learning sciences (p. 427-441). Cambridge, Cambridge University Press.

PEREIRA, L. C. O.; SILVA, M. L. **Android para Desenvolvedores**. Brasport Livros e Multimídia Ltda, 2009. Disponível em: <http://books.google.com.br/books?hl=pt-BR&lr=&id=8u9wJowXfdUC&oi=fnd&pg=PA1&dq=aplicativo+para+android&ots=LSlp441np2&sig=tWNxfIQ5wdt8w8VbG-5AZGhilhM#v=onepage&q=aplicativo%20para%20android&f=false>.

SUEHRING, S. **MySQL Bible**. Wiley Publishing Inc, 2002.

REDMOND, P.; **WRITING APIs WITH LUMEN**. 2016.

TURAN, A. **Student Readiness for Technology Enhanced History Education in Turkish High Schools. Cypriot Journal Of Educational Sciences**. 2010. Artigo Disponível em: <http://www.worldeducation-center.org/index.php/cjes/article/view/75 >. Acesso em: 30 de Março de 2017.

ZHANG, W.; PERRIS, K.; YOUNG, L. (2005). **Online tutorial support in open and distance learning: students perceptions**. British Journal of Educational Technology. Ano 36, n.5, 2005.

YAU, J. K; JOY, M. (2010) “A Context-Aware Personalized m-Learning Application based on m-Learning Preferences”. In: **6th IEEE International Conference on Wireless, Mobile and Ubiquitous Technologies in Education**, p. 11-18.