ENADE SIMULADO – APLICATIVO ANDROID PARA SIMULAÇÃO DAS PROVAS DO ENADE

Rafael Vieira Martins[1]
Rafael Oliveira Vasconcelos[2]

cadernos de graduação
ciências exatas e tecnológicas

Ciência da Computação

ISSN IMPRESSO 1980-1777
ISSN ELETRÔNICO 2316-3135

RESUMO

A questão da qualidade de ensino no Brasil é enfatizada pela Lei de Diretrizes e Bases da Educação Nacional. A avaliação da educação superior tem destaque dentre as políticas educacionais com a criação de avaliações periódicas das instituições e cursos superiores. A avaliação do desempenho dos estudantes da educação brasileira é uma modalidade avaliativa implantada desde meados da década de 1990 e um componente curricular obrigatório aos cursos de graduação, conforme determina a Lei nº10.861/2004 é o Exame Nacional de Desempenho dos Estudantes (ENADE). Como uma maneira de auxiliar os estudantes a se preparar e servir como um auxílio aos estudos, o aplicativo Enade Simulado, traz questões retiradas das provas de edições anteriores do ENADE para que os candidatos possam ter uma ideia de como são as questões e por se tratar de um aplicativo tem a vantagem da mobilidade, pois em qualquer lugar o candidato pode praticar suas habilidades e conhecimentos. Possui um sistema semelhante à de um quiz, o aluno responde questões de múltipla escolha e ao final do questionário o aluno visualiza uma tela mostrando seus erros e acertos no simulado.

PALAVRAS-CHAVE

Enade. Android. Aplicativo. Simulado.

1 INTRODUÇÃO

A permanente busca pela melhoria da qualidade da educação superior, pela sua expansão e pelo acompanhamento da oferta dos cursos de graduação do sistema federal, é um dos objetivos do Sistema Nacional de Avaliação da Educação Superior (SINAES), instituído em 14 de abril de 2004. Com a implementação do Sinaes, o Exame Nacional de Desempenho dos Estudantes (ENADE) passou a substituir o Exame Nacional de Cursos (ENC), conhecido como Provão, criado pela Lei nº 9.131/1995 e aplicado, no período de 1996 a 2003, apenas aos estudantes concluintes e com o objetivo de avaliar os respectivos cursos de graduação (FURB).

A partir da implantação do Sinaes, em 2004, o objetivo do Enade é ampliado e passa a integrar a avaliação de cursos e instituições e a expressar o desenvolvimento da aprendizagem dos estudantes ingressantes e concluintes dos cursos de graduação, em consonância com as Diretrizes Curriculares Nacionais (DNC) para os cursos ofertados (R7).

O impacto que a tecnologia causou em nossas vidas durante os últimos anos foi muito grande, mudamos a maneira como nos relacionamos, seja com outras pessoas, com produtos ou até mesmo como aprendemos. Se antes era preciso imprimir páginas e páginas de provas de questões anteriores para poder fazer um simulado para praticar e estudar, hoje você faz tudo isso em segundos, precisando apenas de um smartphone ou tablet conectado à internet. Os aplicativos são uma realidade e já fazem parte do nosso dia a dia, tornando nossas tarefas mais otimizadas e com a praticidade de resolver tudo de forma ágil e rápida, nos dando mais tempo livre para gastarmos com o que realmente importa.

Uma das áreas que vem ultimamente se beneficiando bastante com esse movimento é a da educação. É a partir dessa perspectiva que o presente artigo consiste em desenvolver um aplicativo capaz de simular provas de edições anteriores do ENADE, com um sistema semelhante à de um quiz onde se torna mais simples e rápida a utilização, mantendo o aproveitamento no aprendizado.

2 REFERENCIAL TEÓRICO

2.1 TECNOLOGIA ANDROID

Hoje em dia os sistemas operacionais encontrados em computadores estão presentes também em dispositivos móveis, principalmente em celulares e tablets. Dentre eles destaca-se o sistema operacional Android, um sistema operacional baseado em Linux que conta com a assinatura de criação da empresa Google. A ideia inicial da empresa era criar um sistema operacional *open source* e flexível, de forma que várias empresas pudessem e fossem motivadas a aderir ao mesmo.

Fato é que o Android ganhou destaque, por ser uma plataforma de fácil desenvolvimento, que permite encontrar um Software de Desenvolvimento (SDK) bem

amigável, que, além disso, possibilita ao desenvolvedor criar aplicativos para vários dispositivos, publicá-los e vendê-los na loja própria do sistema, denominada *Google Play.* Dentre as principais características desse sistema destacam-se, a plataforma adaptada para dispositivos VGA maiores, gráficos 2D, bibliotecas gráficas 3D baseadas em *OpenGL* ES especificação 2.0 e layouts mais tradicionais de smartphones, possuir navegador baseado no framework de código aberto conhecido como *WebKit*, ter suporte a vários formatos de áudio e vídeo como MPEG-4, H.264, MP3 e AAC, incluir no sistema de desenvolvimento um emulador, ferramentas de debugging, memória e análise de desempenho.

Um ponto que chama a atenção e torna o desenvolvimento mais agradável, são os nomes atribuídos a cada versão do Android, pois são curiosamente relacionados à comida como pode ser visto na Figura 1. Desta forma, acredita-se que, para ao utilizar a tecnologia Android, seja possível alcançar um grande número de usuários. Para implementação deste aplicativo são utilizadas técnicas e padrões de desenvolvimento como o Padrão MVC, que será explicada a seguir.

Figura 1 – Demonstração das versões do sistema Android

Fonte: Redação Optclean (2016).

2.2 PADRÃO MVC

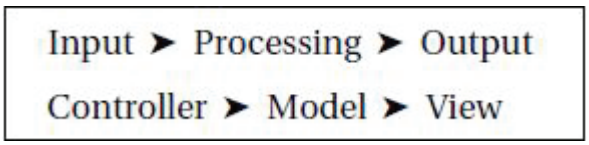
Na fase de Projeto começamos a nos preocupar com a arquitetura da aplicação, mesmo não possuindo uma definição consensual, afinal diversos livros, artigos ou autores definem o que é arquitetura da sua forma, a arquitetura de software de um sistema computacional pode ser definida como a suas estruturas, que são compostas de elementos de software, de propriedades externamente visíveis desses componentes, e do relacionamento entre eles. Ou seja, a arquitetura define os elementos de software e como eles interagem entre si.

A arquitetura de um sistema tem diversos elementos como: elementos utilitários, elementos de interação, elementos que fazem parte do domínio do problema, elementos de conexão, elementos de persistência etc. Dessa forma, na arquitetura sempre definimos os seus elementos que precisarão ser utilizados no software e como eles se conectam. Uma arquitetura complexa exige mais tempo para desenvolvimento, porém, por meio de geração automática de aplicações isso pode se tornar mais produtivo. Por isso algumas equipes definem um framework para uma determinada aplicação, assim podemos utilizar muitas coisas pré-prontas que facilitam o desenvolvimento.

No entanto, alguns padrões arquiteturais já foram pensados para resolver problemas corriqueiros. Alguns projetos ou organizações combinam esses padrões arquiteturais, pois atendem melhor às suas necessidades ou deram certo para o seu tipo de aplicação. Por isso é sempre interessante entender as características básicas de cada um dos estilos e escolher ou combinar aqueles que atendem melhor às necessidades de um projeto específico, isso tudo, deve ser feito após uma análise do sistema a ser desenvolvido. Entre as arquiteturas existentes temos: Cliente-servidor, P2P - Peer to Peer, Dados compartilhados, Máquina virtual, Camadas, MVC e muitos outros. Nessa parte do artigo veremos mais sobre o padrão de arquitetura *Model – View – Controller* (MVC) que é um dos mais antigos e mais utilizados atualmente.

O padrão arquitetural MVC é uma forma de quebrar uma aplicação, ou até mesmo um pedaço da interface de uma aplicação, em três partes: o modelo, a visão e o controlador (RAMOS, 2015). O MVC inicialmente foi desenvolvido no intuito de mapear o método tradicional de entrada, processamento e saída que os diversos programas baseados em GUI utilizavam. No padrão MVC, teríamos então o mapeamento de cada uma dessas três partes para o padrão MVC conforme ilustra a Figura 2.

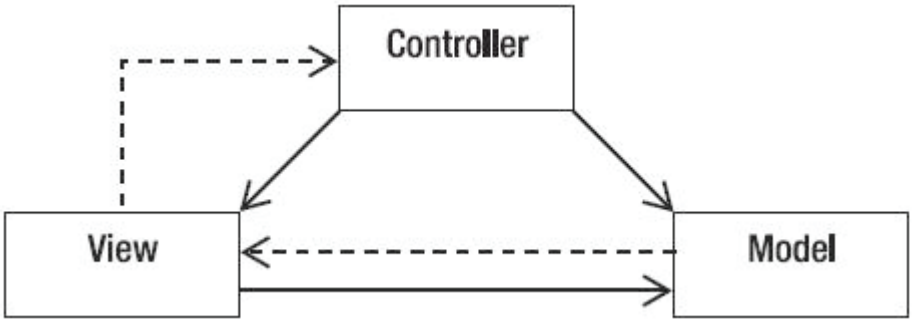
Figura 2 - Mapeamento das três partes de uma aplicação para o MVC



Fonte: Acervo do Autor (2017)

A Figura 3 demonstra que a entrada do usuário, a modelagem do mundo externo e o feedback visual para o usuário que são separados e gerenciados pelos objetos Modelo Model, Visão View e Controlador Controller, para melhor organizar e facilitar o desenvolvimento do projeto.

Figura 3 - Objetos utilizados no MVC e suas interações.



Fonte: Acervo do autor (2017)

Tem-se, explicando cada um dos objetos do padrão MVC, primeiramente o controlador Controller que interpreta as entradas do mouse ou do teclado enviado pelo usuário e mapeia essas ações do usuário em comandos que são enviados para o modelo Model e/ou para a janela de visualização View para efetuar a alteração apropriada. Por sua vez o modelo Model gerencia um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado.

O modelo sabe o que o aplicativo quer fazer e é a principal estrutura computacional da arquitetura, pois é ele quem modela o problema que está se tentando resolver. Por fim, a visão View gerencia a área retangular do display e é responsável por apresentar as informações para o usuário por meio de uma combinação de gráficos e textos. A visão não sabe nada sobre o que a aplicação está atualmente fazendo, tudo que ela realmente faz é receber instruções do controle e informações do modelo e então exibir elas. A visão também se comunica de volta com o modelo e com o controlador para reportar o seu estado.

Tão importante quanto explicar cada um dos objetos do padrão arquitetural MVC é explicar como é o seu fluxo tipicamente. Primeiramente o usuário interage com a interface (por exemplo, pressionando um botão) e o controlador gerenciar esse evento de entrada da interface do usuário. A interface do usuário é exibida pela visão view, mas controlada pelo controlador. O controlador não tem nenhum conhecimento direto da View, ele apenas envia mensagens quando ela precisa de algo na tela atualizado. O controlador acessa o modelo, possivelmente atualizando-a de forma apropriada para as ações do usuário (por exemplo, o controlador solicita ao modelo que o carrinho de compras seja atualizado pelo modelo, pois o usuário incluiu um novo item).

Isto normalmente causa uma alteração no estado do modelo tanto quanto nas informações. Por fim, a visão usa o modelo para gerar uma interface com o usuário apropriada. A visão recebe as informações do modelo. O modelo não tem conhecimento direto da visão. Ele apenas responde a requisições por informações de quem quer que seja e requisita por transformações nas informações feitas pelo controlador. Após isso, o controlador, como um gerenciador da interface do usuário, aguarda por mais interações do usuário, onde inicia novamente todo o ciclo.

Portanto, a principal ideia do padrão arquitetural MVC é a separação dos conceitos – e do código. O MVC é como a clássica programação orientada a objetos, ou seja, criar objetos que escondem as suas informações e como elas são manipuladas e então apresentar apenas uma simples interface para o mundo. Entre as diversas vantagens do padrão MVC estão a possibilidade de reescrita da GUI ou do Controller sem alterar o nosso modelo, reutilização da GUI para diferentes aplicações com pouco esforço, facilidade na manutenção e adição de recursos, reaproveitamento de código, facilidade de manter o código sempre limpo etc.

Existem diversos *frameworks* para Java que implementam o padrão MVC e são muito utilizados em diversos projetos. Entre eles temos o JSF, *Struts* 1 e *Struts* 2, Spring MVC, Play Framework, *Tapestry* e diversos outros. Existem diversos artigos e sites especializados que comparam as diferenças e vantagens entre esses diferentes *frameworks*. No entanto, o melhor é sempre verificar o que cada *framework* disponibiliza para os desenvolvedores e analisar se ele atende às nossas expectativas.

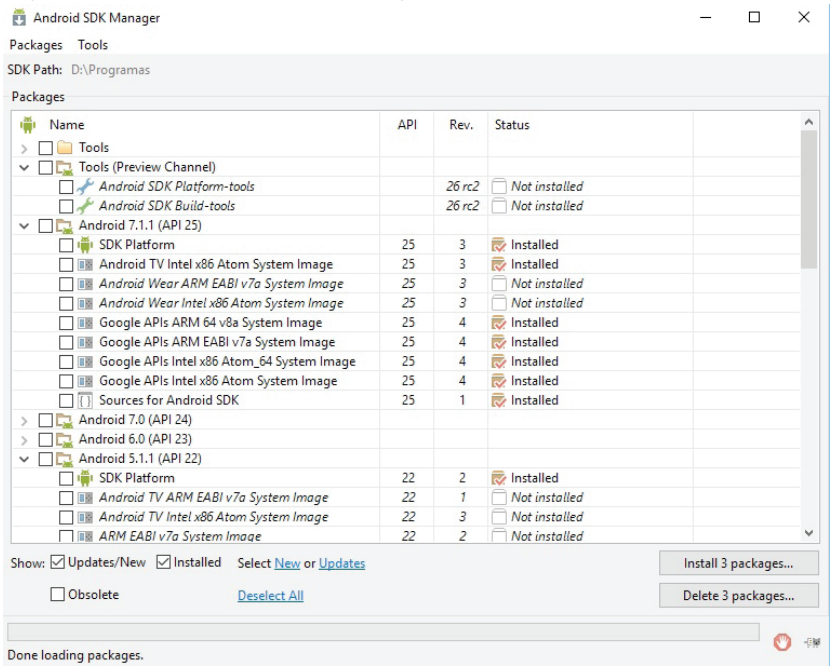
Outras linguagens/plataformas também possuem *frameworks* que aderem ao padrão arquitetural MVC. Isso não inviabiliza que uma equipe crie o seu próprio *framework*, mas é preciso lembrar que um desenvolvedor novo precisa de tempo para aprender a desenvolver em determinada arquitetura e caso a empresa/projeto já utilize um *framework* bastante popular, a curva de aprendizado será bem menor ou praticamente nula. Isso inclusive ajuda na contratação de novos funcionários, onde a empresa já pode exigir como pré-requisito conhecimentos neste *framework*. No projeto desse aplicativo é utilizado o padrão MVC no *framework* android, no ambiente de desenvolvimento conhecido como Android SDK, utilizando a IDE IntelliJ IDEA,ambiente que serviu de base para criação do Android Studio.

2.3 AMBIENTE DE DESENVOLVIMENTO

O Android SDK fornece as bibliotecas de API e as ferramentas necessárias para construir e desenvolver, realizar teste e depuração de aplicativos para o Android. Com um simples download, o pacote SDK inclui tudo que você precisa para começar a desenvolver aplicativos. Foi utilizado para o desenvolvimento desse projeto Android o ambiente de desenvolvimento IntelliJ IDEA.

O Android SDK permite que os desenvolvedores elaborem as aplicações a partir de um dispositivo virtual para os aparelhos de celular e tablet, desde jogos a utilitários que façam uso das funções oferecidas pelos aparelhos, como touchscreen, telefonia GSM, Câmera, GPS, bússola, acelerômetro, Bluetooth, EDGE, 3G e WiFi. O SDK não fornece bancos de dados de outras empresas, mas oferece pacotes de ferramentas que podem ser instalados logo ao abrir o programa pela primeira vez. Como pode ser visto na Figura 4.

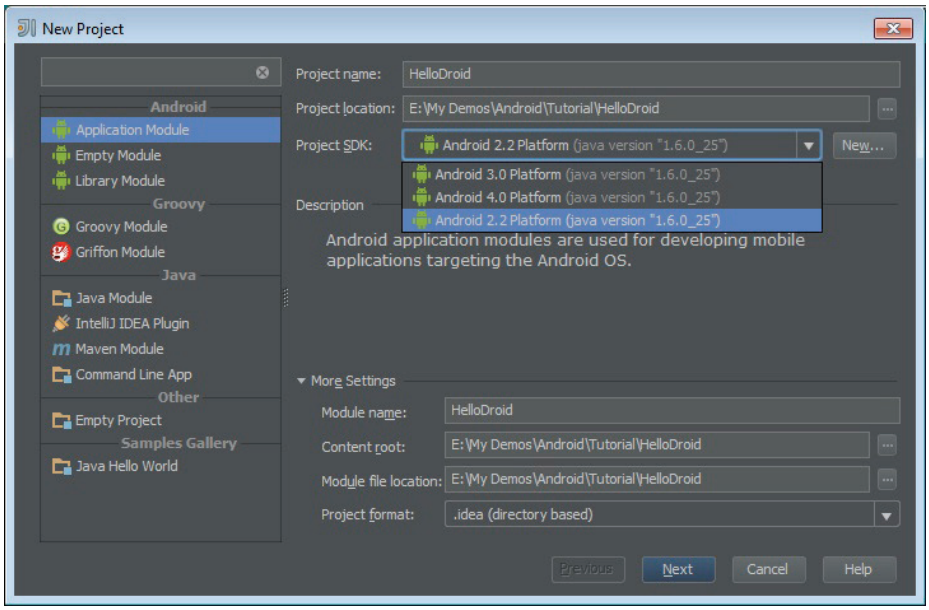
Figura 4 – Android SDK Manager



Fonte: Acervo do autor (2017).

Feito a atualização dos pacotes Android, utiliza-se o próprio IntelliJ para o desenvolvimento das aplicações, de forma mais simples, já que o IntelliJ é bem completo e tem várias funções que facilitam a criação e edição dos códigos.

Figura 5 – Ambiente de desenvolvimento Android IntelliJ



Fonte: Acervo do autor (2017).

2.4 FIREBASE

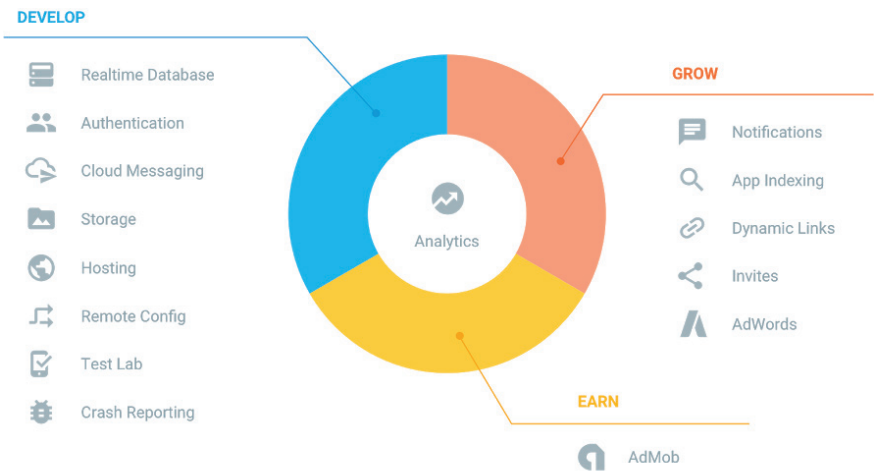
O Firebase é uma plataforma móvel que ajuda você a desenvolver rapidamente aplicativos de alta qualidade, crescer sua base de usuários, e ganhar mais dinheiro. Composto de características complementares que você pode misturar e combinar para atender às suas necessidades (MAES; JEFFERSON, 2017).

A Google é uma das grandes empresas do mercado de tecnologia a apostar com muita força no segmento de serviços baseados na nuvem. Assim, para tentar dar um direcionamento melhor aos desenvolvedores que dependem de Cloud para potencializar seus aplicativos, a Gigante das Buscas transformou seu Firebase em uma plataforma unificada para os profissionais da área mobile.

Esse plano foi revelado durante o Google I/O e mostra que a companhia realmente acredita no potencial da ferramenta – que foi integrada ao acervo da marca em 2014. Ele incorpora diversos recursos da casa para oferecer um leque bem mais robusto de funcionalidades. As funções de análise, por exemplo, se tornam bem mais detalhadas e poderosas com a reformulação, bastando à adição de algumas linhas de código para que apps já lançados possam conversar de forma muito mais natural com o *Google Analytics*. O controle dos desenvolvedores em cima desse tipo de integração é tão alto que permite dizer exatamente como os dados do usuário passam do aplicativo para o banco de dados ou filtrar quais informações devem ser armazenadas ou verificadas durante o uso do software.

De forma geral, essa nova versão da plataforma e do SDK visa solucionar em um só lugar as necessidades dos programadores. A ideia principal é evitar que eles tenham que recorrer a diferentes produtos para tratar de cada elemento das aplicações mobile, como avaliação da audiência e acompanhamento da performance de rendimentos com propagandas.

Figura 6 – Firebase



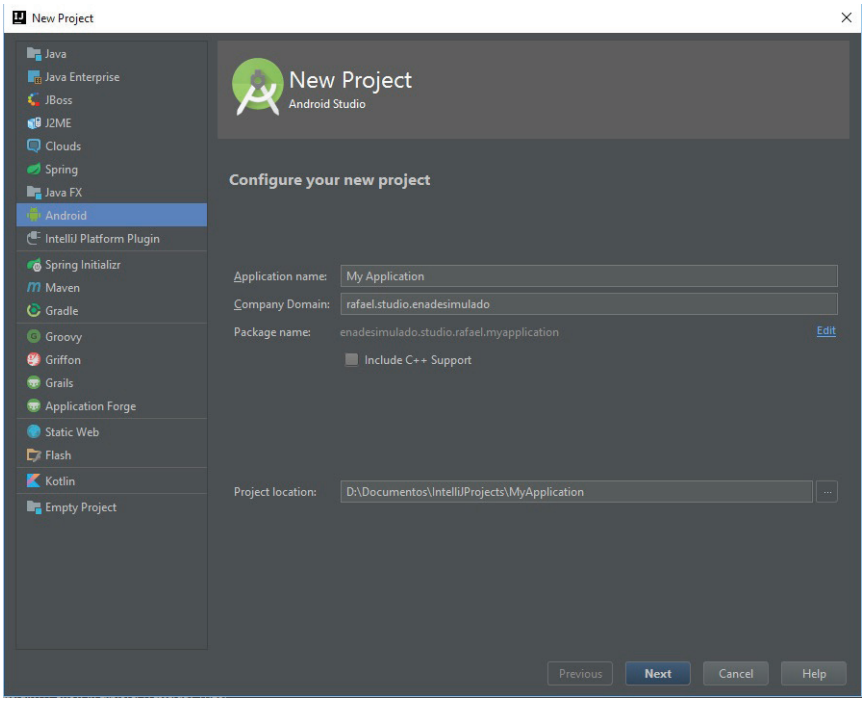
Fonte: Acervo do autor.

3 METODOLOGIA E DESENVOLVIMENTO

3.1 INÍCIO E CRIAÇÃO DAS TELAS – ACTIVITY

Como pode ser visto na Figura 7, na primeira tela é definido o nome da aplicação, o nome da empresa caso possua alguma e exibe o local onde é salvo o projeto, após preencher esses campos basta pressionar o botão Next no canto inferior direito, no canto esquerdo podemos visualizar o quão completo é essa IDE IntelliJ IDEA, pois com ela podemos desenvolver vários tipos de *frameworks*, como por exemplo, *Java*, *Java Enterprise*, JBoss, J2ME, *Spring*, *Java FX, Clouds*, dentre outros, mas nesse projeto somente utilizaremos Android.

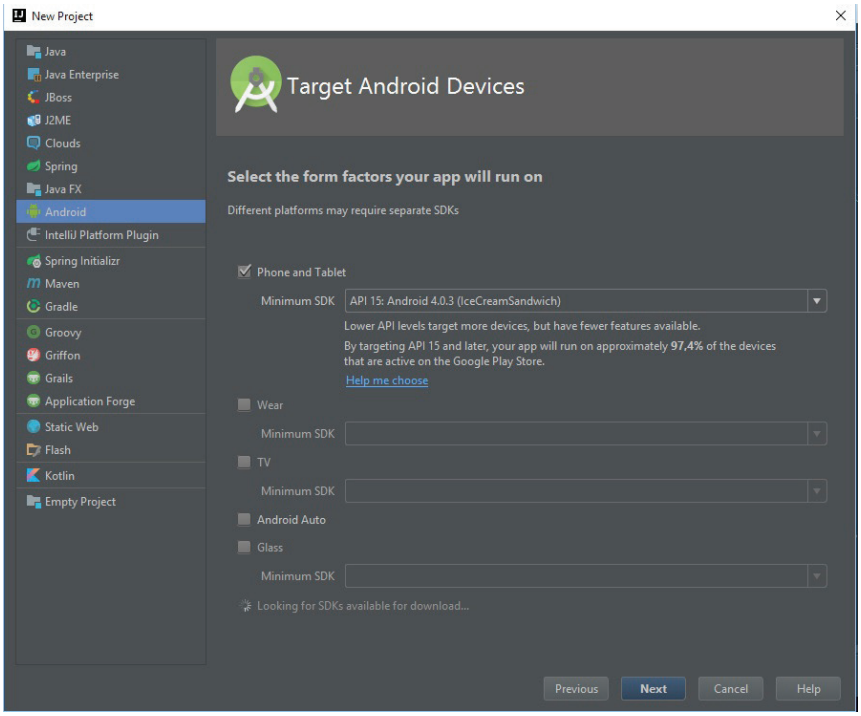
Figura 7 – Novo projeto Android no *IntelliJ*



Fonte: Acervo do autor (2017).

Ao pressionar o botão *Next*, passamos para outra tela de configuração como pode ser visto na Figura 8, onde é definido o tipo de plataforma que será utilizada, podendo ser Phone ou Tablet, *Wear* que seria algum dispositivo vestível como um *Smart Watch*, por exemplo, TV pois também é muito utilizado aplicativos Android em *Smart* TV's, dentre outros, no caso desse projeto será utilizada a plataforma phone e tablets, onde nesse campo é definido o SDK mínimo para rodar o aplicativo, ou seja a versão mínima do sistema operacional instalado no dispositivo, podendo ser smartphone ou tablete, para que o aplicativo funcione corretamente, nesse projeto o sistema mínimo será o Android 4.0.3 (Ice Cream Sandwich).

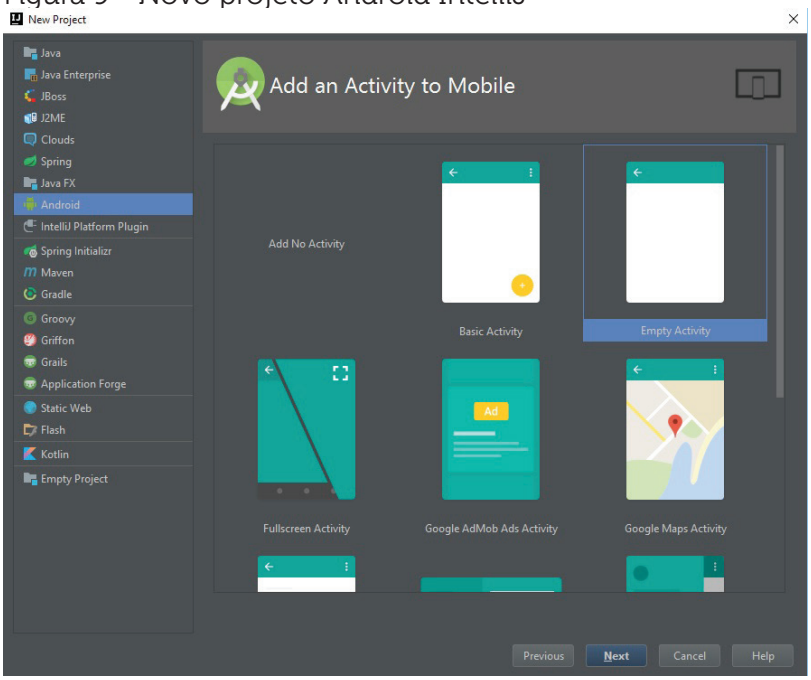
Figura 8 – Novo projeto Android *IntelliJ*



Fonte: Acervo do autor (2017).

Após clicar em Next na tela anterior, iremos para a tela onde escolheremos a *Activity* para iniciar no desenvolvimento do aplicativo, a Figura 9 mostra os tipos de *Activity* que podem ser selecionados. Uma *Activity* é basicamente uma classe gerenciadora de Interface com o usuário (UI). Todo aplicativo android começa por uma *Activity*. Falaremos bastante sobre ela, suas características, seu ciclo de vida e como manipulá-la corretamente. Nesse projeto foi utilizado a *Empty Activity*.

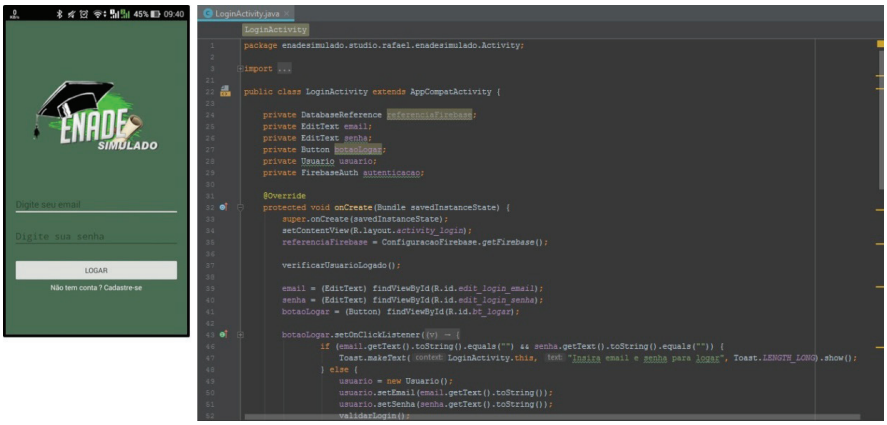
Figura 9 - Novo projeto Android IntelliJ



Fonte: Acervo do autor (2017).

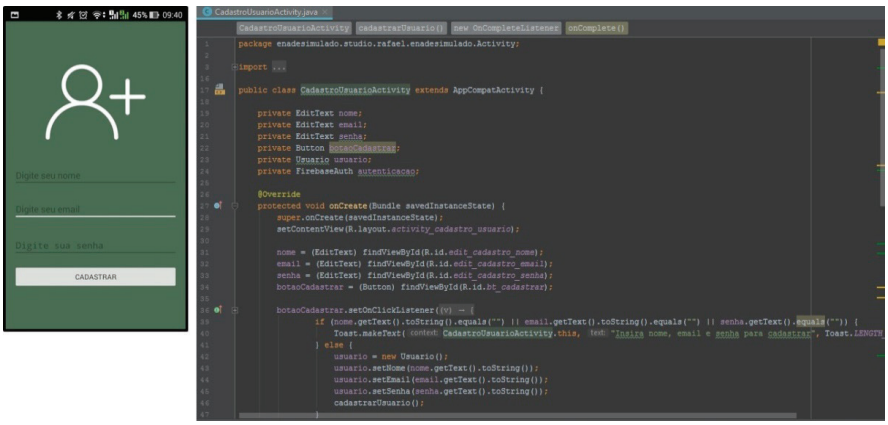
Após selecionar a *activity* basta clicar em *Next* e escolher o nome da *activity*. Logo após basta finalizar e iniciar o desenvolvimento do app. Para esse projeto foram desenvolvidas 5 telas(Activitys) que são: Login, Cadastro de Usuário, Main, Questões e Resultado. Ao ser criada as *Activitys* a IDE *IntellyJ* cria também um arquivo XML correspondente para cada tela onde pode ser editada o formato do seu app, como imagem, botões etc. Nas Figuras abaixo podem ser vistas as activitys do projeto.

Figura 10 – Activity Login



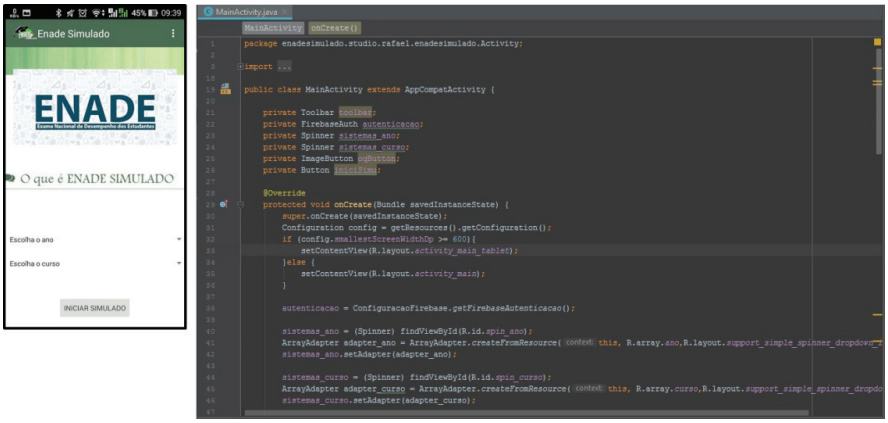
Fonte: Acervo do autor (2017).

Figura 11 – Activity Cadastro de Usuário



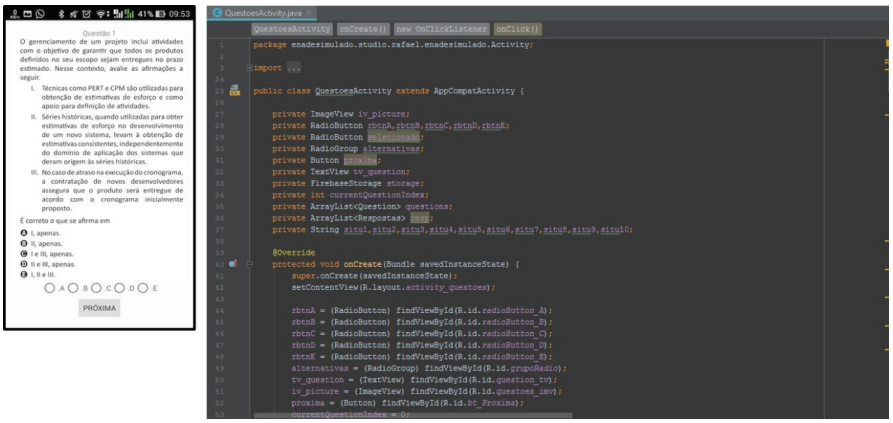
Fonte: Acervo do autor (2017).

Figura 12 – Activity Main



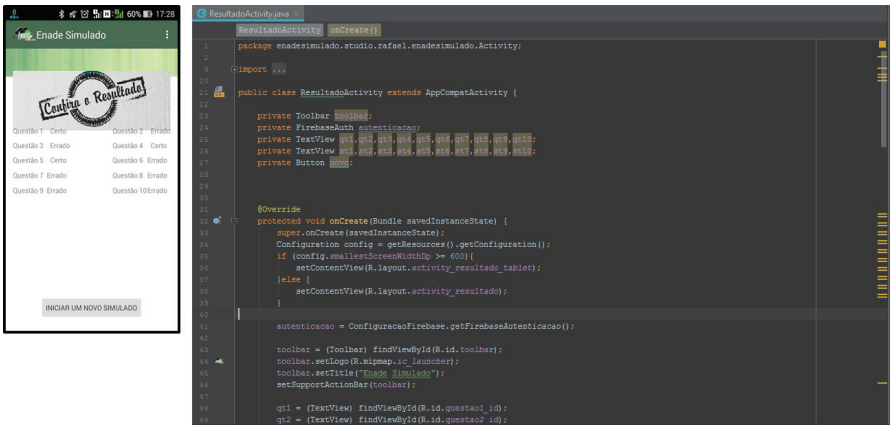
Fonte: Acervo do autor (2017).

Figura 13 – Activity Questões



Fonte: Acervo do autor (2017).

Figura 14 – Activity Resultado



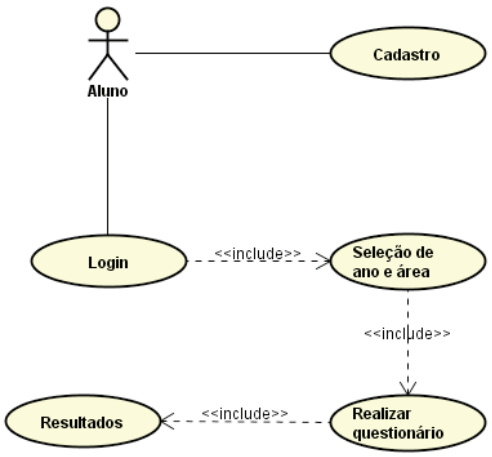
Fonte: Acervo do autor (2017).

3.2 DIAGRAMA E LÓGICA DA PROGRAMAÇÃO DO PROJETO

O diagrama de uso do projeto pode ser visto na Figura 15, nele podemos ver que para o usuário utilizar o aplicativo necessita primeiramente criar um cadastro, informando o nome, e-mail e senha, após preencher todos os campos basta tocar no botão "Cadastrar", logo após poderá efetuar o login, informando o e-mail e senha cadastrados anteriormente. Quando o login é efetuado o usuário verá a tela principal do aplicativo chamada de "Main", é a tela principal onde o usuário pode ter mais informações sobre o aplicativo tocando no campo "O que é ENADE SIMULADO" onde esclarece a quem utilizar o aplicativo, qual o ponto principal e como funciona o app.

Nessa mesma tela, também, é onde se pode iniciar o simulado, primeiramente o usuário seleciona o ano e depois o curso da prova que deseja fazer o simulado, após ser escolhido basta tocar no botão INICIAR SIMULADO. A tela de questões será exibida, mostrando as questões da prova que foi escolhida anteriormente e dará ao usuário as opções de marcar a resposta correta selecionando uma das opções dentre as cinco exibidas abaixo da questão, após selecionar basta tocar em PRÓXIMA, será exibida a próxima questão onde o usuário também responde e toca em PRÓXIMA novamente, até finalizar as questões, e no final será exibida a tela de resultado, mostrando as questões que foram respondidas corretamente ou não.

Figura 15 – Diagrama de Uso do Projeto



Fonte: Acervo do Autor (2017).

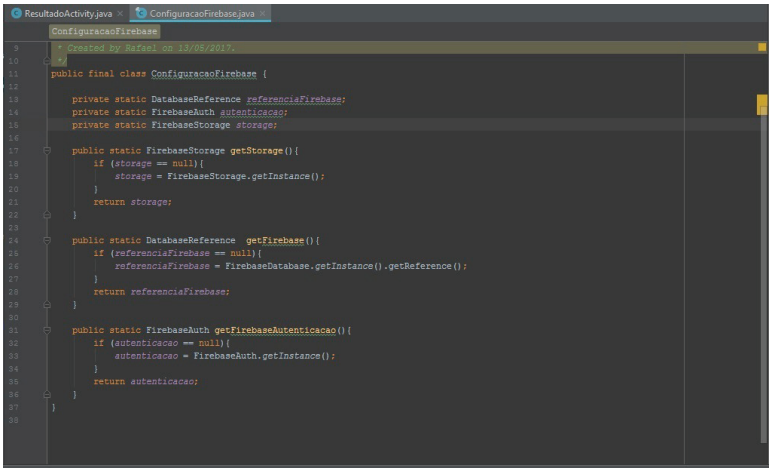
A lógica de programação do projeto foi realizada utilizando os recursos do Firebase, para autenticação, para salvar os dados dos usuários e para guardar as questões das provas do ENADE, como mostra a Figura 16.

Para a autenticação foi utilizada a instância do *FirebaseAuth*, o *Firebase Authentication* fornece serviços de *back-end, SDKs* fáceis de usar e bibliotecas de IU prontas para o uso para autenticar os usuários em seu aplicativo. Ele oferece suporte à autenticação, usando senhas, provedores populares de identidades de identidades federadas, como o *Google*, *Facebook* e *Twitter*, e muito mais (GUIDES, 2017a).

Para salvar os dados dos usuários foi utilizado o banco de dados em tempo real do Firebase por meio da instância Database *Reference*, o *Firebase Realtime* Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria aplicativos multiplataforma com nossos SDKs para iOS, Android e *JavaScript*, todos os seus clientes compartilham uma instância de *Realtime* Database e automaticamente recebem atualizações com os dados mais recentes (GUIDES, 2017b).

Para guardar as questões das provas foi utilizada a instância *Firebase Storage*, o *Firebase Storage* proporciona *uploads* e *downloads* de arquivos seguros para seus aplicativos do *Firebase*, independentemente da qualidade da sua rede. Você pode usá-lo para armazenar imagens, áudio, vídeo ou outro conteúdo gerado por usuários. O *Firebase Storage* é respaldado pelo *Google Cloud Storage*, um serviço de armazenamento de objetos que é simples e econômico (GUIDES, 2017c).

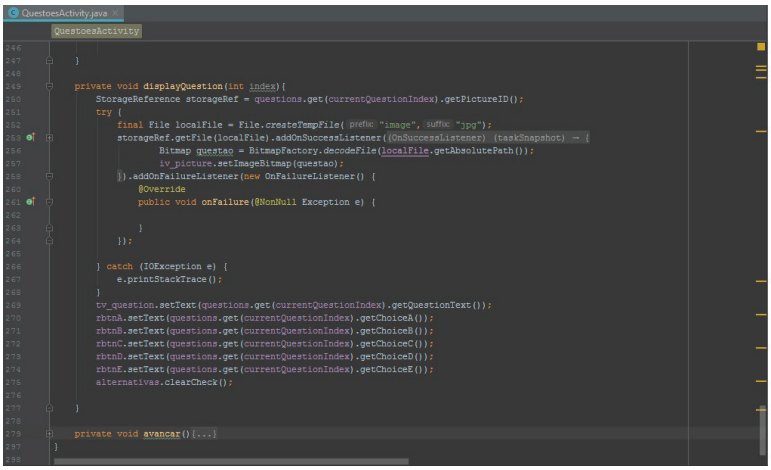
Figura 16 – Instâncias do Firebase



Fonte: Acervo do autor (2017).

As questões das provas de edições anteriores foram obtidas no site do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), e recortadas como imagem, questão por questão, desse modo pôde-se guardar as imagens no *storage* do *Firebase* para ser baixada pelo aplicativo no celular do usuário, como uma imagem, só que temporária, de maneira que não ocupe muito espaço no celular, com isso, sempre que uma imagem é baixada ela sobrescreve a atual, como pode ser visto na Figura 17. O método *display Question* cria um arquivo local temporário com o nome de "image" e sufixo "jpg" que recebe o arquivo referenciado do *storage*, faz o *download* e salva no *Bitmap* onde o arquivo é decodificado e exibido no *Image View*, que é o componente que exibe a imagem na tela de questões.

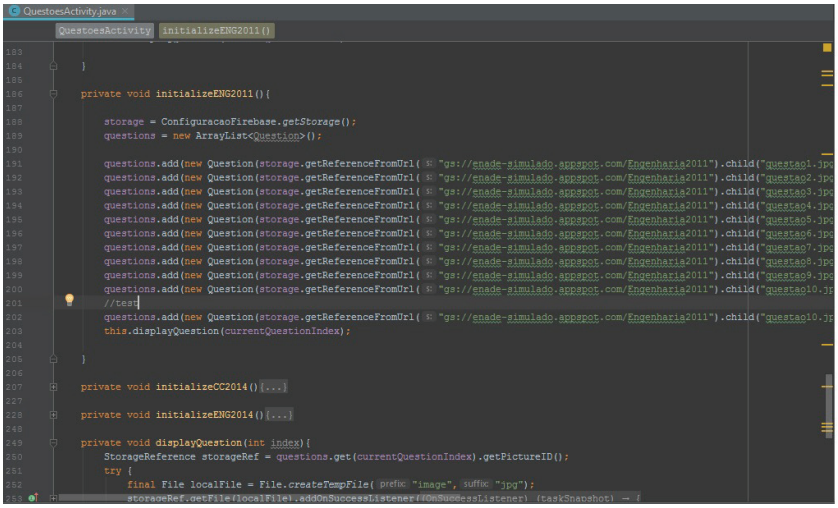
Figura 17 – Código para download das questões



Fonte: Acervo do autor (2017).

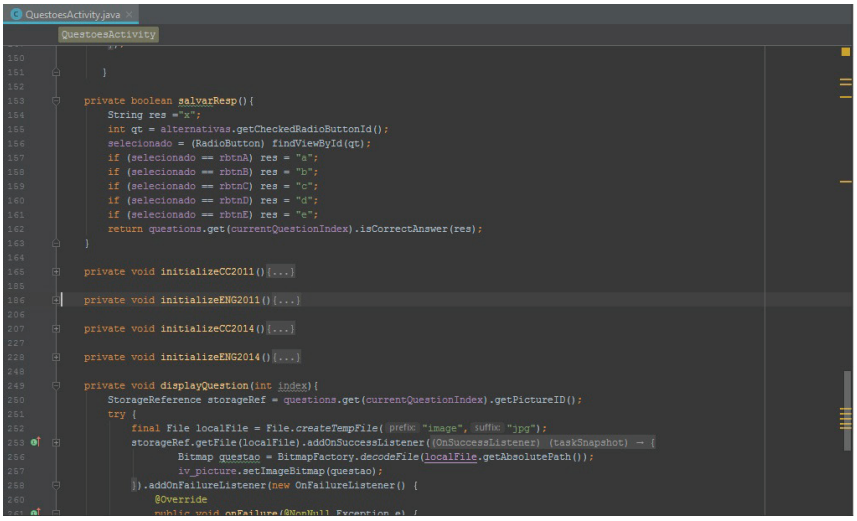
As referências das questões são guardadas em uma *ArrayList* Figura 18, onde possui também as respostas corretas de cada questão, para que ao ser tocado o botão PRÓXIMA seja realizada a correção da resposta dada pelo usuário por meio do método salvarResp, exibido na Figura 19, que faz a correção comparando a resposta dada pelo usuário com a resposta correta salva na *ArrayList*. Uma vez que a resposta está correta, é salvo em uma variável para ser exibido depois na tela de resultado, para que seja feita a conferência pelo usuário. Da mesma maneira ocorre caso a questão esteja errada, sendo salvo na mesma variável para ser exibida na tela de resultado.

Figura 18 – *ArrayList* com a referência das questões



Fonte: Acervo do autor (2017).

Figura 19 – Método de correção das respostas



Fonte: Acervo do autor (2017).

4 CONSIDERAÇÕES FINAIS

Em virtude dos argumentos apresentados, foi possível concluir que a programação realizada na criação do app que auxilia na simulação de uma prova do ENADE, atingiu as metas propostas em termos de funcionalidade. Os resultados obtidos no aplicativo foram condizentes com a expectativa, podendo ajudar vários alunos a exercitar seus conhecimentos para se preparar para a prova do ENADE em qualquer lugar. Por ser um aplicativo gratuito, pode ser utilizado para fins acadêmicos, como forma de revisão ou exercício em sala e até por professores em buscas de questões do ENADE para aplicação em suas provas ou exercícios.

Para trabalho futuro dentro desse escopo, uma possibilidade de ampliação e melhoria pode ser implementada, como a exibição das questões respondidas, ou da média de acerto das questões.

REFERÊNCIAS

FURB. **O que é ENADE?** Disponível em: <http://www.furb.br/web/3237/enade-exame-nacional-de-desempenho-dos-estudantes/o-que-e-o-enade>. Acesso em: 3 jun. 2017.

GUIDES. **Firebase Autentication.** Atualizado em 8 julho 2017. Disponível em: <https://firebase.google.com/docs/auth/?hl=pt-br>. Acesso em: 3 jun. 2017.

GUIDES. **Firebase Realtime Database.** Atualizado em 8 julho 2017. Disponível em: <https://firebase.google.com/docs/database/?hl=pt-br>. Acesso em: 3 jun. 2017.

GUIDES. **Firebase Storage.** Atualizado em 8 julho 2017. Disponível em: <https://firebase.google.com/docs/storage/?hl=pt-br>. Acesso em: 3 jun. 2017.

MAES, Jefferson. **Firebase o que é e para que serve?** Disponível em: <http://digitalprimews.com/google-firebase/>. Acesso em: 3 jun. 2017.

MATTES, FABIO. **Introdução ao desenvolvimento android.** Disponível em: <https://www.vivaolinux.com.br/artigo/Introducao-ao-Desenvolvimento-Android>. Acesso em: 3 jun. 2017.

OPTCLEAN. **Google atualiza dados de distribuição do Android – Marshmallow sobe para 7,5% em maio de 2016.** Disponível em: <https://optclean.com.br/google-atualiza-dados-de-distribuicao-do-android-marshmallow-sobe-para-75-em-maio-de-2016/>. Acesso em: 3 jun. 2017.

R7. **Entenda para que serve o ENADE e como é a prova.** Disponível em: <http://noticias.r7.com/educacao/noticias/entenda-o-que-e-e-para-que-serve-a-prova-do-enade-20091108.html>. Acesso em: 3 jun. 2017.

RAMOS, Allan. **O que é MVC (Model, View, Controller)?** Mar. 2015. Disponível em: <https://pt.stackoverflow.com/questions/55486/o-que-%C3%A9-mvcmodel-view-controller>. Acesso em: 3 jun. 2017.

Data do recebimento: 10 de Junho de 2017
Data da avaliação: 5 de Julho de 2017
Data de aceite: 15 de Julho de 2017

1 Graduando do curso de ciência da computação – UNIT – email: rafael.martins@souunit.com.br
2 Professor do curso de ciência da computação – UNIT – email: rafael.oliveira@souunit.com.br