UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

DAVI ARAÚJO LOPES SILVA

APLICATIVO PARA GESTÃO DE APOSTAS E BOLÕES DE JOGOS DA MEGA-SENA

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

DAVI ARAÚJO LOPES SILVA

APLICATIVO PARA GESTÃO DE APOSTAS E BOLÕES DE JOGOS DA MEGA-SENA

Monografia de especialização apresentada ao curso de Especialização em Desenvolvimento para Dispositivos Móveis, da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de especialista.

Orientador: Prof. Dr. Robson Ribeiro Linhares.

CURITIBA

2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

"Aplicativo para Gestão de Apostas e Bolões de Jogos da Megasena"

por

"Davi Araújo Lopes Silva"

Este Trabalho de Conclusão de Curso foi apresentado às 18:26 do dia 19 de dezembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

______________________________ Prof. Robson Ribeiro Linhares **(Presidente/Orientador - UTFPR/Curitiba)**	______________________________ Prof. Adriano Francisco Ronszcka **(Avaliador 1 – Externo)**
______________________________ Prof. Paulo Mauricio de Lucchi Bordin **(Avaliador 2 – Externo)**	

"A Ata de Aprovação assinada encontra-se na Coordenação do Curso."

AGRADECIMENTOS

É possível que estes parágrafos não agradeçam a todas as pessoas que direta ou indiretamente fizeram parte dessa etapa importante da minha vida acadêmica. Portanto peço desculpas por não os referenciar nestas palavras, mas acreditem que tenho muita gratidão pelo apoio.

Agradeço primeiramente a Deus, pela oportunidade de realizar um trabalho de conclusão de um curso de especialização.

Agradeço à minha família pelo apoio e compreensão sobre o esforço, dedicação e o tempo necessário para a elaboração deste trabalho.

Agradeço ao Professor Robson Ribeiro Linhares pela dedicação de tempo e paciência, e pela orientação necessária para realização deste trabalho.

Agradeço também a todos os professores do DAINF pela dedicação e paciência em lecionar as disciplinas do curso de especialização.

RESUMO

SILVA. Davi Araújo L. Aplicativo para gestão de apostas e bolões de jogos da Mega-Sena. 2017. 80 f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Esta monografia apresenta as etapas e conceitos necessários para o desenvolvimento de um protótipo de aplicativo móvel para o sistema operacional *Android.* O propósito do aplicativo destina-se a gestão de apostas e bolões de jogos da Mega-Sena. Como existe uma grande quantidade de apostadores e com a popularização dos smartphones com sistema operacional *Android* um aplicativo como proposta nesta monografia pode alcançar um grande número de usuários. Este trabalho apresenta os conceitos de levantamento de requisitos, modelagem UML, design de interação para construir a documentação de análise e design do protótipo de aplicativo, e utiliza a ferramenta de desenvolvimento *Android Studio* para realizar a implementação do aplicativo.

Palavras-chave: Android, UML, Meg- Sena, Bolão.

ABSTRACT

SILVA. Davi Araújo L. Application for management of bets and bets in group of Mega-Sena games. 2017. 80 f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

This monograph presents the steps and concepts necessary for the development of a mobile application prototype for the Android operating system. The purpose of the application is for the management of bets and betting on groups of Mega-Sena. As there are a lot of bettors and with the popularization of smartphones with Android operating system an application as proposed in this monograph can reach a large number of users. This work presents the concepts of requirements analysis, UML modeling, interaction design to build analysis documentation and design of the application prototype, and uses the Android Studio development tool to carry out the implementation of the application.

Keywords: Android, UML, Mega-Sena, Bolão.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
APK	Android Package
APP	Application
CRUD	Create, Read, Update and Delete
HTTP	Hypertext Transfer Protocol,
IDE	Integrated Development Environment
JEE	Java Enterprise Edition
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
NDK	Native Development Kit
OCR	Optical Character Recognition
OEM	Original Equipment Manufacturer
REST	Representational State Transfer
SD	Secure Digital
SDK	Software Development Kit
SMS	Short Messenger Service
SO	Sistema Operacional
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VM	Virtual Machine
WEB	World Wide Web
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

Existe um grande número de apostadores em bolões da Caixa, 22 milhões em 2016, a arrecadação do governo através dos jogos de loteria chega a ultrapassar 10 bilhões por ano (CAIXA, 2016).

É o mais popular entre os apostadores e principal produto das Loterias CAIXA, a Mega-Sena é a loteria que distribui as maiores premiações ao acertador dos seis números sorteados, premiando também quem acertar cinco ou quatro números.

Dentro desse universo das loterias existe um grande grupo de pessoas que reúnem amigos, colegas do trabalho, colegas de faculdade, e do futebol por exemplo, para realizar bolões da Mega-Sena, onde são feitas muitas apostas. Criar um aplicativo para dispositivos moveis para gerenciar os participantes, quantidades de bilhetes, quantidade de apostas e números apostados, irá facilitar o gerenciamento de apostas e bolões desses apostadores.

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

O objetivo geral consiste em realizar o desenvolvimento de uma aplicação para dispositivos móveis que permita o usuário cadastrar e gerenciar suas apostas da mega-sena, trazendo algumas facilidades na gestão e manutenção do histórico do apostador.

1.2.2 Objetivos Específicos

- Gerar uma lista de requisitos em alto nível a partir do entendimento do escopo e entrevistas com pessoas que costumam apostar na mega-sena.
- Efetuar a Análise dos requisitos da aplicação através da metodologia ágil, transformando-os em *User Stories*.
- Elencar as *User Stories* mais importantes levando em consideração a complexidade/tempo de implementação versus a importância da funcionalidade.

- Criar um esboço do aplicativo com a arquitetura da informação.
- Projetar um design de interface de usuário levando em conta as melhores práticas de *UX* para os dispositivos *Android*.
- Definir a arquitetura sistêmica do aplicativo, no que diz respeito a tecnologias e ferramentas que serão utilizadas.
- Modelar as principais funcionalidades em UML.
- Realizar a implementação do aplicativo de acordo com as especificações.
- Criar e executar os planos de testes e validar a implementação do sistema.
- Documentar a implementação e os resultados/dificuldade obtidos.

1.3 JUSTIFICATIVAS

Só em Bolão a CAIXA registrou-se em 2016 mais de 22,37 milhões de apostas, em sua maioria, nas modalidades Mega-Sena, Loto fácil e Quina (CAIXA, 2016). A falta de aplicativos semelhantes no mercado, os números expressivos que esse jogo movimenta são as principais motivações para a construção do aplicativo. Existem diversas pessoas que organizam com um grupo de pessoas a compra de centenas de bilhetes para tentar aumentar a chance de acerto. O aplicativo poderia servir para auxiliar a leitura, gravação e histórico dessas centenas de jogos.

Outro dado importante é sobre a escolha da plataforma *Android* para o desenvolvimento desse protótipo de aplicação. O número de dispositivos móveis com o sistema operacional *Android* representa cerca de 92.1% do mercado nacional, com base no mês de março de 2017 (KANTAR WORD PANEL, 2017).

Como existe uma grande quantidade de dispositivos com o sistema operacional *Android*, muito provavelmente, desenvolver um aplicativo para essa plataforma poderá alcançar um número muito maior de usuários.

1.4 ESCOPO

O escopo deste trabalho de conclusão consiste na criação de um aplicativo móvel para o sistema operacional *Android*, e uma aplicação secundária web, responsável por prover informações para o aplicativo através de WebService. O aplicativo móvel será uma ferramenta para auxiliar jogadores da mega-sena, em especial, realizadores de bolões. Neste sentido, as principais funcionalidades do aplicativo serão:

1. Gerenciar os jogos/bolões do usuário, mantendo dados dos participantes, números apostados e etc.
2. Capturar os bilhetes apostados por meio da câmera do celular e com isso realizar a leitura dos jogos por meio de uma biblioteca OCR.

O Aplicativo não fará consultas ou operações diretamente na Caixa Econômica, todas as informações relevantes serão providas através de WebService contido na aplicação secundária.

O aplicativo também disponibiliza algumas informações estatísticas relevantes as apostas realizadas pelo apostador. Neste projeto não está prevista a criação de uma versão do aplicativo para a web, apenas será desenvolvido um aplicativo para o sistema operacional Android.

1.5 METODOLOGIA

Para a realização desse trabalho será necessário a pesquisa sobre aplicativos semelhantes existentes no mercado de dispositivos moveis, e também comentários na Play Store, para mapear seus pontos positivos e negativos. Logo em seguida serão utilizados os mesmos princípios da metodologia de trabalho científico para revisar os fundamentos teóricos, com ênfase em material online / livros.

Para a construção do aplicativo será utilizada uma metodologia ágil para desenvolvimento de software que é uma metodologia mais dinâmica, flexível e com maior produtividade, usando-se do framework Scrum que tem alguns elementos interessantes para o desenvolvimento de um projeto, como por exemplo a quebra de requisitos em *User Story* que são agrupadas em uma lista chamada *Product Backlog*.

1.6 ORGANIZAÇÃO DO TRABALHO

O trabalho será organizado em mais cinco capítulos, além do capítulo atual que é a introdução. Nos próximos três capítulos veremos os aspectos técnicos necessários para o desenvolvimento da aplicação seguido pelo capítulo de conclusão onde veremos um resumo de todos o desenvolvimento do trabalho e por fim um capítulo de referências apenas para informar as fontes de pesquisas utilizadas no decorrer do trabalho. Segue abaixo mais detalhes sobre os capítulos subsequentes:

2º Capítulo - Referencial teórico estudo: Neste capítulo serão descritos todos os estudos necessários para a construção da aplicação em um em nível técnico de detalhe sobre os conceitos, fundamentos e tecnologias utilizados.

3º Capítulo - Desenvolvimento do trabalho: Este capítulo é a parte do trabalho onde será demonstrado todas as etapas de desenvolvimento necessárias para a construção do aplicativo, sendo dividida em três partes:

Análise **-** Levantamento e especificação de requisitos, descrição e desenho da arquitetura do sistema.

Design **-** Modelagem UML das principais funcionalidades, projeto de interface de usuário.

Implementação - Descrição do artefato, manual de utilização, código fonte de partes essenciais dos sistemas.

4º Capítulo - Estudo de caso: Descrição do plano de teste, cenário de teste, resultados dos testes e considerações

5º Capítulo - Conclusões: Resumo de todo desenvolvimento do trabalho incluindo dificuldades, pontos fortes e trabalhos futuros.

6º Capítulo - Referências: Este capítulo deverá apresentar as fontes de todas as pesquisas realizadas na construção deste trabalho.

2 REFERENCIAL TEÓRICO

2.1 APLICATIVOS SEMELHANTES NO MERCADO

Nesta seção, são apresentados dois aplicativos semelhantes existentes no mercado que possuem funcionalidades semelhantes às propostas nesta monografia. Também são apresentados os principais pontos positivos e negativos dessas aplicações, tendo como base as avaliações e os comentários das lojas de aplicativos onde estes aplicativos estão publicados.

2.1.1 Mega-sena

Aplicativo com o um bom número de downloads, tem uma interface de usuário amigável e menu lateral conforme Figura 1 e Figura 2, traz informações sobre os últimos resultados e estatísticas dos sorteios da Mega-sena.

Tem como pontos positivos a opção para geração de números aleatórios (jogo conhecido como surpresinha) e análise dos números, exibindo quantas vezes o número já foi sorteado e há quantos sorteios fazem que o número não é sorteado novamente. Conforme algumas avaliações no Google Play, um ponto negativo é que o aplicativo não salva os números jogados, funcionalidade que poderia ser interessante em conjunto com o conferencia automática da aposta. No desenvolvimento do aplicativo proposto neste trabalho será disponibilizado uma opção para salvar a aposta feita pelo usuário, essa é uma das opções que se diferenciam das funcionalidades do aplicativo Mega-Sena.

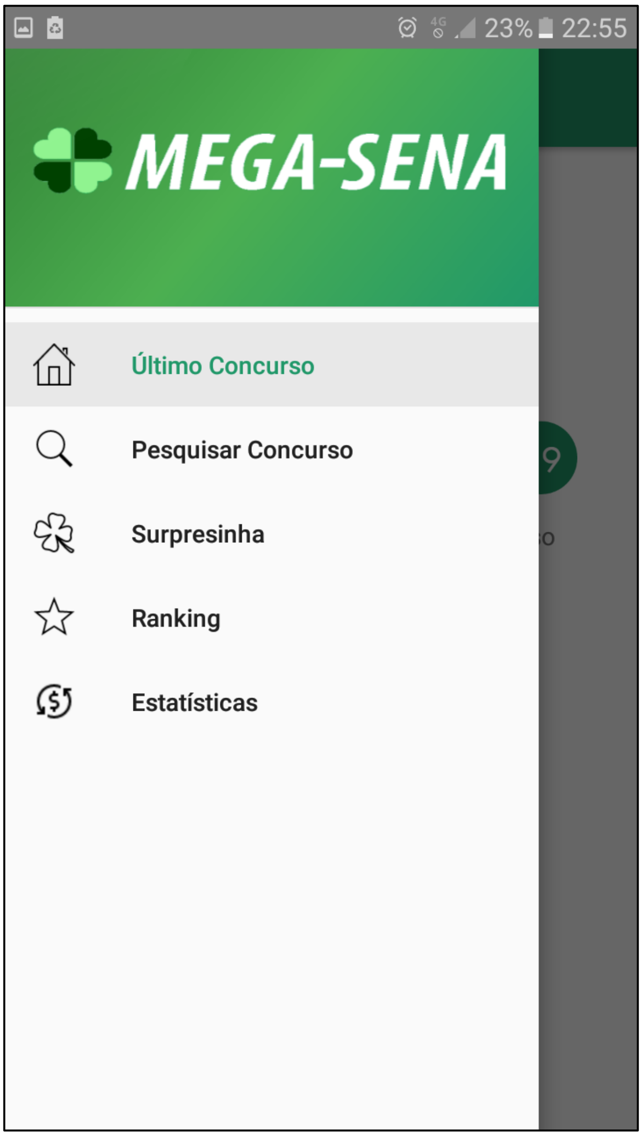

Figura 1 – Aplicativo Mega-Sena – Menu lateral
Fonte: Captura de tela do aplicativo Mega-Sena.

Figura 2 – Aplicativo Mega-Sena – Pagina Inicial
Fonte: Captura de tela do aplicativo Mega-Sena.

2.1.2 Ganhe Mega-Sena

Aplicativo com menos funcionalidades que o aplicativo Mega-Sena, e com interface de usuário simples conforme Figura 3, um ponto interessante é a geração de números com algumas técnicas de geração como por exemplo números das laterais, números na horizontal e vertical conforme apresentado na Figura 4.

Os pontos negativos desse aplicativo são não ter opções de análise de números igual ao Mega-Sena, a consulta do último resultado não é direta no aplicativo ele simplesmente direciona para o navegador no site da caixa, e também assim como o aplicativos Mega-Sena não tem opção para salvar os jogos.

O aplicativo proposto neste trabalho trará uma interface de usuário mais rica e com menu lateral para facilitar a usabilidade. Outro ponto de melhoria será a consulta direta dos resultados da Mega-sena, sem a necessidade de ser direcionado para um navegador.

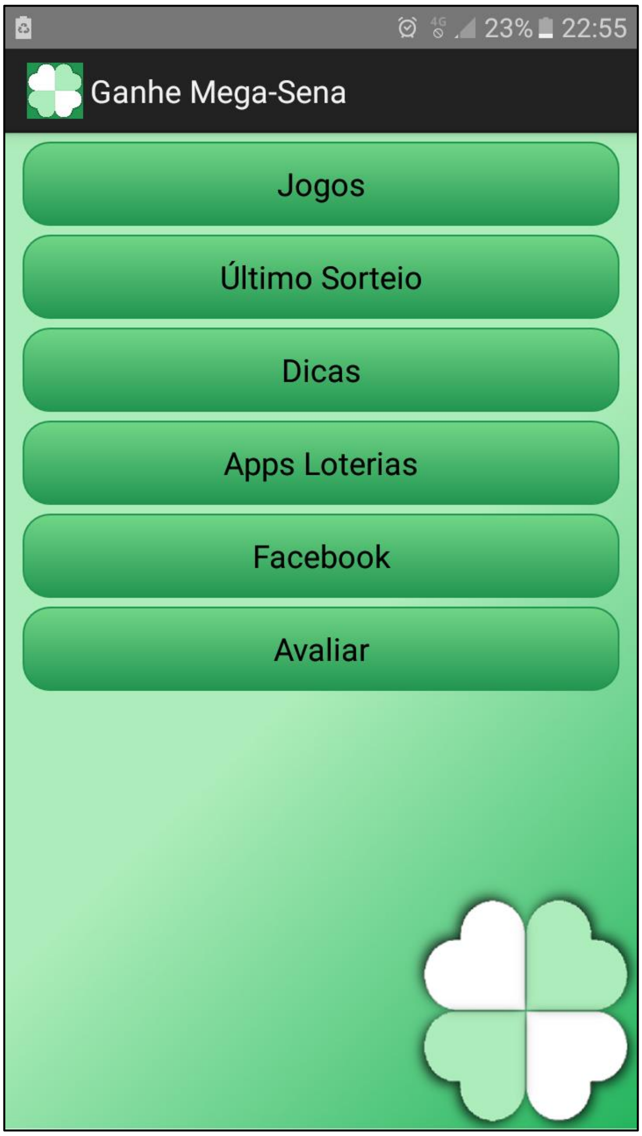

Figura 3 – aplicativo ganhe mega-sena.
Fonte: Captura de tela do aplicativo Ganhe Mega-Sena.

Figura 4 – aplicativo ganhe mega-sena – Gerar Jogos.
Fonte: Captura de tela do aplicativo Ganhe Mega-Sena.

2.2 METODOLOGIAS ÁGEIS

A metodologia ágil surgiu após uma equipe de profissionais americanos da área de software se reunirem para discutir uma forma de melhorar o desempenho de seus projetos. (HIGHSMITH, 2001).

Embora cada envolvido tivesse seus princípios e teorias de como fazer um projeto de software ter sucesso, cada um com suas particularidades, todos eles perceberam e concordaram que, existiam alguns pouco princípios que pareciam ser respeitados quando projetos davam certo. Com base nisso, eles criaram um manifesto que descreviam alguns princípios básicos para o desenvolvimento de software ágil, que são apresentados a seguir:

- Indivíduos e interação entre eles mais que processos e ferramentas.
- Software em funcionamento mais que documentação abrangente.
- Colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

2.2.1 User Stories

User Stories ou histórias de usuário, são uma forma de definir e organizar os requisitos de um sistema que está sendo gerido por um projeto de desenvolvimento de software ágil. As *User Stories* focam nos objetivos do usuário e de que forma o sistema alcança esses objetivos. Uma característica das *User Stories* é que elas devem ser uma fração de um requisito, ou seja, devem ser pequenas, para facilitar o entendimento e a estimativa do esforço. Normalmente *User Stories* são escritas do ponto de vista do usuário (GLAUCO, 2011).

2.3 ANDROID

Android é um sistema operacional baseado em *Linux* e atualmente desenvolvido pela empresa de tecnologia *Google*. O *Android* está presente em múltiplos aparelhos de diversas fabricantes, como Samsung, Motorola, LG, e Sony, é a plataforma para dispositivos móveis mais popular do mundo. É conhecido por ser baseado no núcleo do Linux, ter um código aberto e uma série de possibilidades de personalização.

Um aspecto interessante é que celulares não são os únicos dispositivos a serem equipados com o sistema operacional *Android*. O *Android* é implantado também em tablets, consoles móveis, e agora uma série de novos produtos, como TVs, computadores e dispositivos vestíveis, já estão sendo fabricados com ele. (BARROS, 2015).

O *Android* é uma pilha de software conforme Figura 5, de código aberto criada para uma ampla gama de dispositivos com diferentes tamanhos de tela. Os principais objetivos do *Android* são criar uma plataforma de software aberta disponível para operadoras, OEM’s e desenvolvedores para tornar suas ideias inovadoras uma realidade e apresentar um produto bem-sucedido e real que melhore a experiência móvel para usuários. (GOOGLE, 2017).

Na figura 5 é demonstrado a maioria dos componentes da plataforma Android. O Kernel do Linux é a fundação da plataforma Android, uma das suas funções é o encadeamento e gerenciamento de memória de baixo nível.

A camada de abstração de hardware (HAL), fornece as interfaces que expõe as capacidades de hardware para a estrutura do *JAVA API*. A camada Android Runtime é responsável por criar várias maquinas virtuais de forma que cada aplicativo seja executado em uma instância própria.

A camada *JAVA API*, fornece um conjunto completo dos recursos do *Android SO.* Essa *API* forma os blocos de programação necessários para o desenvolvimento de aplicativos *Android.*

E por fim a camada de aplicativos do sistema, que são um conjunto de aplicativos padrões do sistema *Android* para executar diversas tarefas como enviar *SMS*, e-mail, navegar na internet e etc.

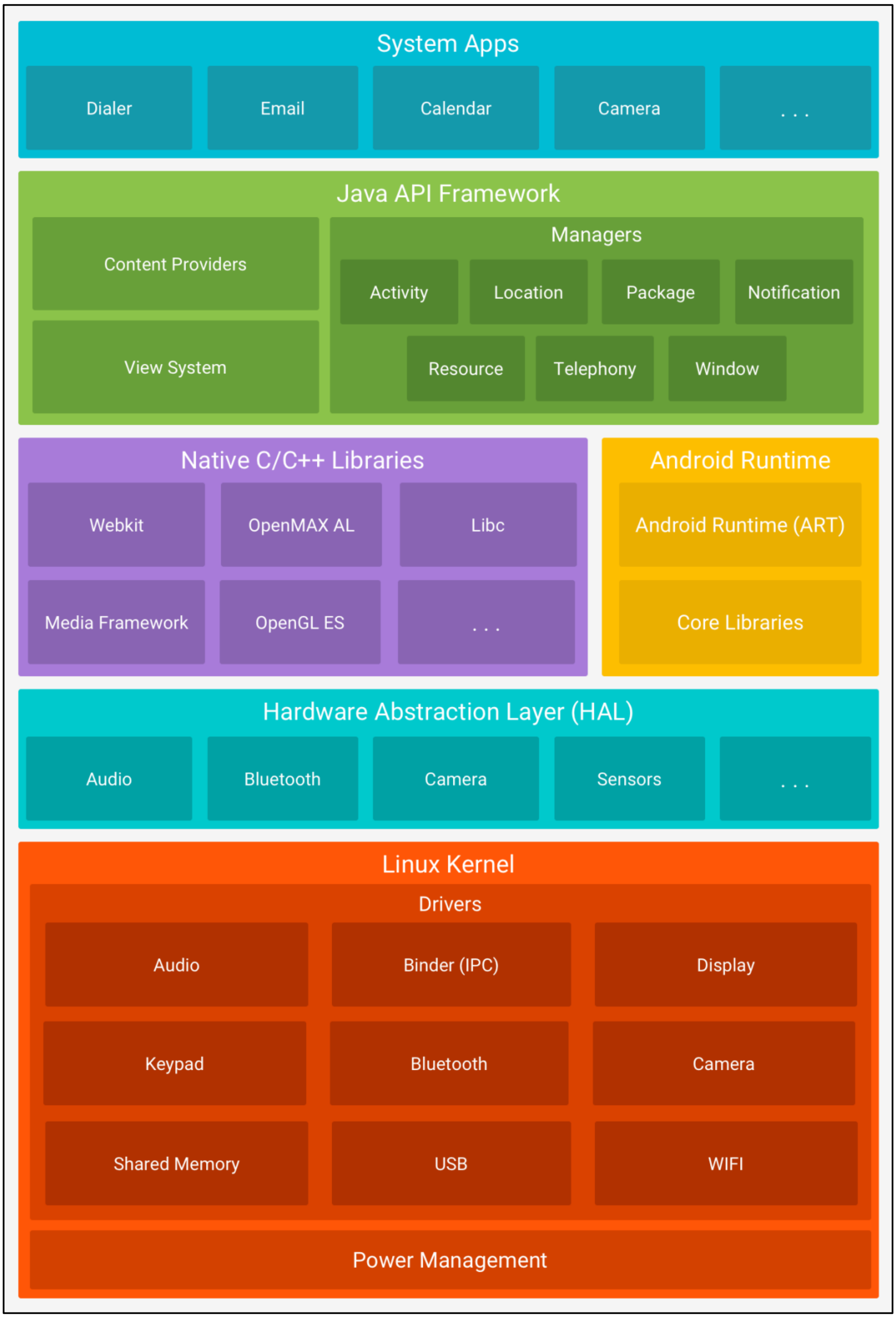

Figura 5 – A pilha de software do Android.
Fonte: Google Developers (2017).

2.3.1 Fundamentos de Aplicativos Android

Os aplicativos para Android são desenvolvidos em um ambiente de linguagem de programação Java. As ferramentas do *Android SDK* compilam o código em conjunto com todos os arquivos de dados e recursos em um *APK*, um pacote Android, que é um arquivo de sufixo. *apk*. Os arquivos de *APK* contêm todo o conteúdo de um aplicativo do Android e são os arquivos que os dispositivos desenvolvidos para Android usam para instalar o aplicativo.

Depois de instalado em um dispositivo, cada aplicativo do Android é ativado na própria *sandbox,* que é uma área separada do restante do S.O e de outras aplicações de modo que garanta a segurança e não prejudique o S.O em caso de falha. A *sandbox* onde o aplicativo é ativado possui algumas características em decorrência do Android ser baseado em Linux:

- O sistema operacional *Android* é um sistema Linux multiusuário em que cada aplicativo é um usuário diferente.
- Por padrão, o sistema atribui a cada aplicativo um ID de usuário do Linux exclusivo (o ID é usado somente pelo sistema e é desconhecido para o aplicativo). O sistema define permissões para todos os arquivos em um aplicativo, de modo que somente o ID de usuário atribuído àquele aplicativo pode acessá-los.
- Cada processo tem a própria máquina virtual (VM), portanto, o código de um aplicativo é executado isoladamente de outros aplicativos.
- Por padrão, cada aplicativo é executado no próprio processo do Linux. O Android inicia o processo quando é preciso executar algum componente do aplicativo, em seguida, encerra-o quando não mais é necessário ou quando o sistema precisa recuperar memória para outros aplicativos.

O sistema Android implementa o princípio do privilégio mínimo. Ou seja, cada aplicativo, por padrão, tem acesso somente aos componentes necessários para a execução do seu trabalho e nada mais. Isso cria um ambiente muito seguro em que o aplicativo não pode acessar partes do sistema para o qual não tem permissão.

No entanto, existe uma maneira de um aplicativo compartilhar dados com outros aplicativos e acessar serviços do sistema, mas são necessárias algumas configurações específicas como por exemplo atribuir o mesmo certificado para os aplicativos:

- É possível fazer com que dois aplicativos compartilhem o mesmo ID de usuário do Linux, caso em que eles são capazes de acessar os arquivos um do outro. Para preservar os recursos do sistema, os aplicativos com o mesmo ID de usuário também podem ser combinados para serem executados no mesmo processo Linux e compartilharem a mesma VM (também é preciso atribuir o mesmo certificado aos aplicativos).
- Um aplicativo pode solicitar permissão para acessar dados de dispositivo como contatos do usuário, mensagens SMS, o sistema montável (cartão SD), câmera, Bluetooth etc. O usuário precisa conceder essas permissões de forma explícita. (ANDROID DEVELOPERS, 2017).

2.3.2 Componentes de aplicativo

Os componentes de aplicativo são os blocos de construção de um aplicativo Android, esses componentes são desenvolvidos de forma desacoplada do aplicativo para que possam ser reutilizados por outro aplicativos, por exemplo um aplicativo de e-mail pode usar o componente de foto de um outro aplicativo para tirar uma foto e depois enviá-la por anexo. Cada componente é um ponto diferente pelo qual o sistema pode entrar em seu aplicativo. Nem todos os componentes são pontos de entrada reais para o usuário e alguns dependem uns dos outros, mas cada um existe como uma entidade independente e desempenha uma função específica — cada um é um bloco de construção exclusivo que ajuda a definir o comportamento geral do aplicativo.

Há quatro tipos de componentes de aplicativo. Cada tipo tem uma finalidade distinta e tem um ciclo de vida específico que define a forma pela qual o componente é criado e destruído.

A seguir são apresentados os quatro tipos de componentes de aplicativos.

2.3.2.1 Atividades

Atividades representam uma tela única com uma interface do usuário, basicamente um Atividade é uma classe que gerencia a interação do usuário. Por exemplo, um aplicativo de e-mail pode ter uma atividade que mostra uma lista de novos e-mails, outra atividade que compõe um e-mail e outra ainda que lê e-mails. Embora essas atividades funcionem juntas para formar uma experiência de usuário coesa no aplicativo de e-mail, elas são independentes entre si. Portanto, um aplicativo diferente pode iniciar qualquer uma dessas atividades (se o aplicativo de e-mails permitir). Por exemplo, um aplicativo de câmera pode iniciar a atividade no aplicativo de e-mail que compõe o novo e-mail para que o usuário compartilhe uma foto. Uma atividade é implementada como uma subclasse de *Activity*. (ANDROID DEVELOPERS, 2017).

2.3.2.2 Provedores de conteúdo

O *Content Provider* (Provedor de conteúdo) é um recurso disponibilizado pelo Android para permitir o compartilhamento de dados entre diferentes aplicações. Isso se faz necessário porque o Android não possui um local comum para guardar os dados das aplicações e compartilhá-los entre todos os aplicativos Android.

Esse recurso se trata de uma abstração das fontes de dados de camadas inferiores como, por exemplo, banco de dados *SQLite*, além de disponibilizar alguns mecanismos de segurança de dados, como controle de permissão de leitura e escrita de dados, e disponibilizar uma interface padrão para compartilhar dados entre aplicações.

Cada *Content Provider* possui um caminho único. O modelo de dados utilizado pelo *Content Provider* é como uma simples tabela de banco de dados, onde cada linha é um registro e cada coluna são os dados de um tipo particular. A interface que geralmente é usada é o *ContentResolver*. Sua função é receber requisições de seus clientes e “resolvê-las”, ou seja, envia a requisição para um *Content Provider* distinto. O *ContentResolver* possui métodos CRUD (*create, read, update, delete*) correspondentes aos métodos abstratos do Content Provider (*insert, delete, query, update*). Cada método recebe um URI especificando o endereço do *Content Provider* no qual se deve interagir. (REIS, 2014).

2.3.3 Ativação de componentes

No que diz respeito a componentes três dos quatro tipos (atividades, serviços e receptores de transmissão), são ativados por uma mensagem assíncrona chamada *intent*. Os *intents* vinculam componentes individuais entre si em tempo de execução (como mensageiros que solicitam uma ação de outros componentes), seja o componente pertencente ao aplicativo ou não.

O *intent* é criado com um objeto *Intent*, que define uma mensagem para ativar um componente específico ou um *tipo* específico de componente — os *intents* podem ser explícitos ou implícitos, respectivamente. (GOOGLE, 2017).

2.3.4 O arquivo de manifesto

O Arquivo de manifesto é o arquivo principal do projeto de aplicativo Android, nele ficam todas as configurações do aplicativo. Antes de o sistema Android iniciar um componente de aplicativo, é preciso ler o arquivo `AndroidManifest.xml` (o arquivo de "manifesto") do aplicativo para que o sistema saiba que o componente existe. O aplicativo precisa declarar todos os seus componentes nesse arquivo, que deve estar na raiz do diretório do projeto do aplicativo. O manifesto além de declarar os componentes do aplicativo, realiza outras operações como por exemplo:

- Identifica todas as permissões do usuário de que o aplicativo precisa, como acesso à internet ou acesso somente leitura aos contatos do usuário.
- Declara o nível de API mínimo exigido pelo aplicativo com base nas APIs que o aplicativo usa.
- Declara os recursos de hardware e software usados ou exigidos pelo aplicativo, como câmera, serviços de Bluetooth ou tela multi-toque.
- As bibliotecas de API às quais o aplicativo precisa se vincular (outras além das APIs de estrutura do Android), como a biblioteca Google Maps. (ANDROID DEVELOPERS, 2017).

2.3.5 Linguagem de programação

A linguagem oficial para desenvolver aplicativos para *Android* é o *Java* que é uma linguagem de programação orientada a objetos desenvolvida por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Atualmente o *Java* é mantido e atualizado pela Oracle.

As aplicações em Java normalmente podem ser executadas em qualquer plataforma que possua a *Java Virtual Machine* (JVM) instalada, independente da arquitetura do computador, esta particularidade acontece porque o código desenvolvido em *Java* pode ser compilado em uma forma intermediária, que é conhecida como *bytecode,* com isso, esta forma intermediária pode ser interpretada pela *Java Virtual Machine.* O código java tambem pode ser compilado em código nativo dependendo da estratégia de configuração da *JVM*. (PEREIRA, 2009).

2.4 AMBIENTE DE DESENVOLVIMENTO

Umas das melhores ferramentas para desenvolvimento de aplicativos para Android é o Android Studio que é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android e é baseado no *IntelliJ IDEA*. Além do editor de código e das ferramentas de desenvolvedor avançados do IntelliJ, o Android Studio oferece ainda mais recursos para aumentar sua produtividade na criação de aplicativos Android, como:

- Um sistema de compilação flexível baseado no Gradle.
- Um emulador rápido com inúmeros recursos.
- Um ambiente unificado para você poder desenvolver para todos os dispositivos Android.
- *Instant Run* para aplicar alterações a aplicativos em execução sem precisar compilar um novo APK.
- Modelos de códigos e integração com *GitHub* para ajudar a criar recursos comuns dos aplicativos e importar exemplos de código.

- Ferramentas de verificação de código suspeito para detectar problemas de desempenho, usabilidade, compatibilidade com versões e outros.
- Compatibilidade com C++ e NDK.
- Compatibilidade embutida com o *Google Cloud Platform*, facilitando a integração do *Google Cloud Messaging* e do *App Engine*. (ANDROID DEVELOPERS, 2017)

2.5 WEB SERVICE

Segundo ERL (2009), o conceito de serviços em uma aplicação já existe há algum tempo. Serviços, assim como componentes de software, são considerados blocos de construção independentes, os quais em conjunto representam um ambiente de aplicação. No entanto, diferente de componentes tradicionais, serviços têm algumas características únicas que lhes permitem participar como parte de uma arquitetura orientada a serviços.

Uma destas características é a completa autonomia em relação a outros serviços. Isto significa que cada serviço é responsável por seu próprio domínio, o que tipicamente significa limitar seu alcance para uma função de negócio específica (ou um grupo de funções relacionadas).

Webservice é uma solução normalmente utilizada na integração de sistemas e na comunicação entre aplicações distintas. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *WebServices* são componentes que permitem às aplicações enviar e receber dados em um formato comum como por exemplo XML. Cada aplicação pode ter a sua própria linguagem, que é traduzida para uma linguagem universal, normalmente no formato XML. (SOAWEBSERVICES, 2017).

2.5.1.1 Provedor de serviços

Agindo como um provedor de serviços, *um web service* expõe uma interface pública através da qual pode ser chamado por consumidores do serviço. Um provedor de serviços disponibiliza esta interface publicando uma descrição do serviço. Pensando em um modelo cliente-servidor, o provedor de serviço pode ser comparado ao servidor.

O termo “provedor de serviço” pode também ser usado para descrever a organização ou ambiente que hospeda (provê) *o web service*.

Um provedor de serviço pode também agir como um consumidor de serviço. Por exemplo, *um web service* pode atuar como um provedor de serviço quando um solicitante de serviço lhe pede para executar uma função. Pode então atuar como um consumidor de serviço quando mais tarde contata o consumidor de serviço original (agora agindo como um provedor de serviço) para solicitar informação de status.

2.5.1.2 Solicitante de serviço

Basicamente um solicitante ou consumidor de serviço é o remetente de uma mensagem *web service* ou o programa de software solicitando um *web service* específico. O solicitante de serviço pode ser comparável ao cliente dentro de um modelo cliente-servidor padrão.

Um solicitante de serviço pode também ser um provedor de serviço. Por exemplo, num modelo de solicitação e resposta, o *web service* iniciador primeiro age como um solicitante de serviço ao requerer informações do provedor de serviço. O mesmo *web service* então, faz o papel de um provedor de serviço ao responder à solicitação original. (THOMAS ERL, 2009)

2.6 REST

A *Representational State Transfer (REST),* em português transferência de estado representacional, é um conceito abstrato da arquitetura da Web. Essa abordagem teve origem na tese de *Ph.D.* de Roy T. Fielding que é um dos principais autores da especificação *HTTP* que é o protocolo utilizado por sites da *interne*t, *Rest* é um conceito bastante abstrato, já que não se trata de um protocolo, e sim uma metodologia, um conjunto de regras e práticas que estabelece como ocorre a troca de informações.

Rest utiliza o próprio protocolo *HTTP*, com o princípio que todo recurso de aplicações deveria responder aos mesmos métodos. O conceito de *Rest* prega que as interações partem do conceito de requisições devem ser feitas por meio dos identificadores uniformes, que são conhecidos como URI. Deste modo seriam usadas as requisições e os métodos *HTTP* para realizar a comunicação entre as aplicações. (CONFERENCIA WEB, 2015).

Segundo Parves (2017) *Rest* é a tecnologia mais fácil para desenvolver serviços web. Hoje em dia, é muito popular na área dos serviços da web. Não há necessidade de usar o formato de troca de dados *XML* para solicitação e resposta.

Os serviços da Web *Rest* podem ser retornados em XML, *JSON* ou mesmo em formato HTML conforme Figura 6.

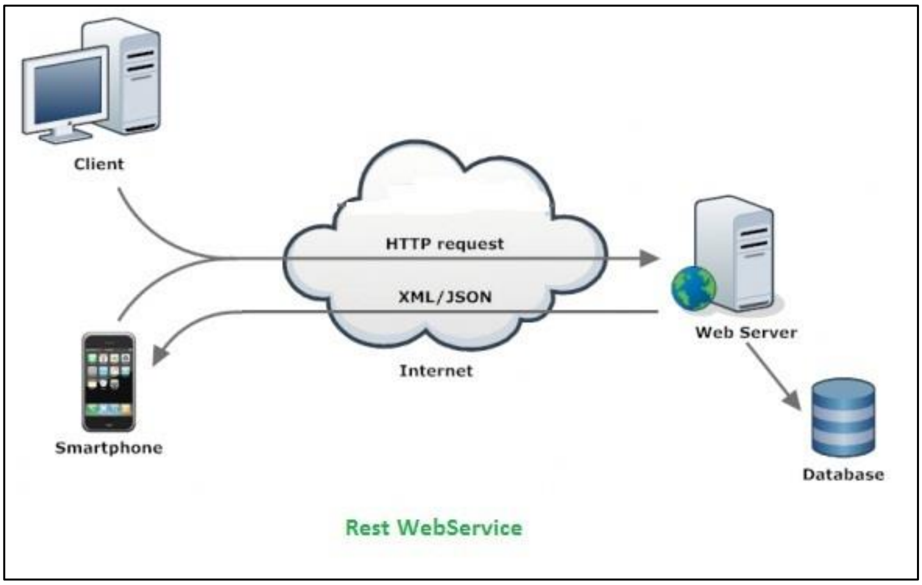

Figura 6 – Web Service Rest.
Fonte: PHP FLOW (2017).

2.7 JAX-RS E JERSEY

JAX-RS é uma especificação de referência para o desenvolvimento de serviços da Web *RESTful* na linguagem de programação Java projetada para facilitar o desenvolvimento de aplicativos que usam a arquitetura *REST*.

Jersey é uma API que implementa as especificações da *JAX-RS* usando as anotações de linguagem de programação Java para simplificar o desenvolvimento de serviços da Web *RESTful*. Os desenvolvedores inserem nos arquivos de classe de linguagem de programação Java anotações específicas de HTTP para definir recursos e as ações que podem ser executadas nesses recursos. As anotações de Jersey são anotações de tempo de execução, portanto, a reflexão de tempo de execução gerará as classes auxiliares e os artefatos para o recurso e, em seguida, a coleção de classes e artefatos será incorporada em um arquivo de aplicativo da Web (WAR). Os recursos podem ser expostos aos clientes, implantando o WAR em *Java EE* ou servidor web. Logo abaixo na tabela 1 está uma lista de algumas das anotações de programação Java que são definidas pelo JAX-RS, com uma breve descrição de como cada uma delas é usada. (ORACLE, 2017).

Anotação	Descrição
@Path	O valor da anotação *@Path* é um caminho *URI* relativo que indica onde a classe Java será hospedada, por exemplo, */helloworld*. Com isso é possível incorporar variáveis nas *URIs* para criar um modelo de caminho *URI*. Por exemplo, é possível solicitar o nome de um usuário e passá-lo para o aplicativo como uma variável na *URI*, assim, */helloworld/{nomeusuario}.*
@GET	A anotação *@GET* é um designador de método de solicitação que corresponde ao método *HTTP* chamado. O método Java anotado com este designador de método de solicitação processará pedidos *HTTP GET*. O comportamento de um recurso é determinado pelo método *HTTP* ao qual o recurso está respondendo.
@PathParam	A anotação *@PathParam* é um tipo de parâmetro que você pode extrair para uso em sua classe de recursos. Os parâmetros do caminho do URI são extraídos do *URI* da solicitação e os nomes dos parâmetros correspondem aos nomes das variáveis do modelo do caminho do *URI* especificado na anotação *@Path* no

Anotação	Descrição
	nível da classe.
@Consumes	A anotação @*Consumes* é usada para especificar os tipos de mídia MIME de representações que um recurso pode consumir que foram enviadas pelo cliente.
@*Produces*	A anotação @*Produces* é usada para especificar os tipos de representação MIME de representações que um recurso pode produzir e enviar de volta para o cliente, por exemplo, "*plain/text*".

Quadro 1 – Anotações *JAX-RS*.
Fonte: ORACLE, 2017.

2.8 JSON

JSON (JavaScript Object Notation - Notação de Objetos *JavaScript*) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. *JSON* é um conceito bem antigo que está baseado em um subconjunto da linguagem de programação *JavaScript*, *Standard* ECMA-262 3a Edição - Dezembro - 1999. *JSON* é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens *C* e familiares, incluindo *C++, C#, Java, JavaScript, Perl*, *Python* e muitas outras. Estas propriedades fazem com que *JSON* seja um formato ideal de troca de dados por isso ele vem sendo usando amplamente na Web. *JSON* pode ser utilizado em conjunto com o conceito de Rest para realizar troca de mensagens, com e evolução da web e o uso de chamadas assíncronas que são chamadas de *AJAX*, é comum o consumo de serviços *Rest* que retornam objetos *JSON*, porque é fácil de ler o conteúdo através do *JavaScript* e quando conveniente inseri-lo no HTML.

JSON está constituído em duas estruturas:

- Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, *record*, *struct*, dicionário, *hash table, keyed list*, ou *arrays* associativas.
- Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

Estas são estruturas de dados universais. Virtualmente todas as linguagens de programação modernas as suportam, de uma forma ou de outra. É aceitável que um formato de troca de dados que seja independente de linguagem de programação se baseie nestas estruturas.

Em *JSON*, os dados são apresentados da forma explicada a seguir, um objeto é um conjunto desordenado de pares nome/valor. Um objeto começa com “{“ (chave de abertura) e termina com “}” (chave de fechamento). Cada nome é seguido por “:” (dois pontos) e os pares nome/valor são seguidos por “,” (vírgula).

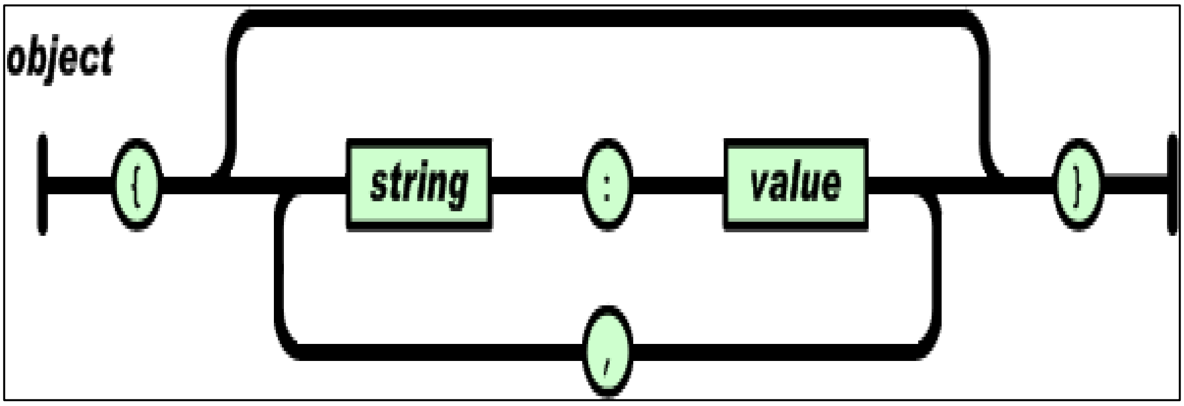

Figura 7 – Objeto JSON.
Fonte: JSON, 2017.

Uma *array* é uma coleção de valores ordenados. O *array* começa com [(colchete de abertura) e termina com] (colchete de fechamento). Os valores são separados por “, ” (vírgula). (JSON, 2017).

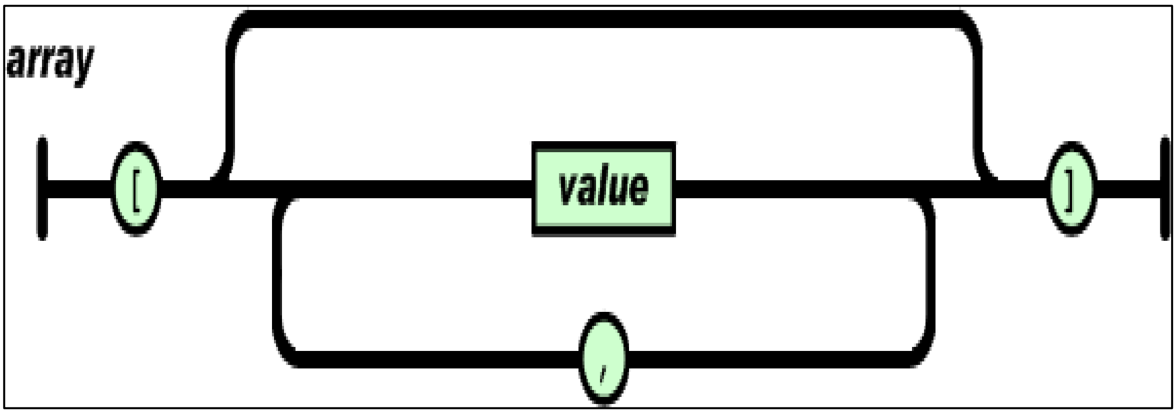

Figura 8 – Coleção de Valores JSON.
Fonte: JSON, 2017.

3 DESENVOLVIMENTO DO TRABALHO

Neste capitulo serão apresentadas as etapas necessárias para o desenvolvimento do protótipo de aplicativo móvel para gerenciar jogos e bolões da Mega-Sena.

3.1 ARQUITETURA DO SISTEMA

No desenvolvimento deste projeto será utilizada a linguagem de programação *Java* na versão 1.7.

O projeto está dividido em duas partes. A primeira é a aplicação em si que será um aplicativo nativo *Android* desenvolvido com uso da IDE de desenvolvimento Android Studio versão 3.0, a versão mínima do *Android* escolhida para executar este aplicativo será a 4.2 (J*elly Bean*), essa versão mínima foi escolhida devido a existir uma quantidade muito pequena de dispositivos executando verões anteriores a 4.2 (GLOBAL STATS, 2017). O aplicativo Android utilizará as redes móveis ou Wi-Fi para conectar no Web Service e buscar as informações atualizadas dos concursos da Mega Sena. Na figura 9, podemos observar como será a arquitetura do projeto.

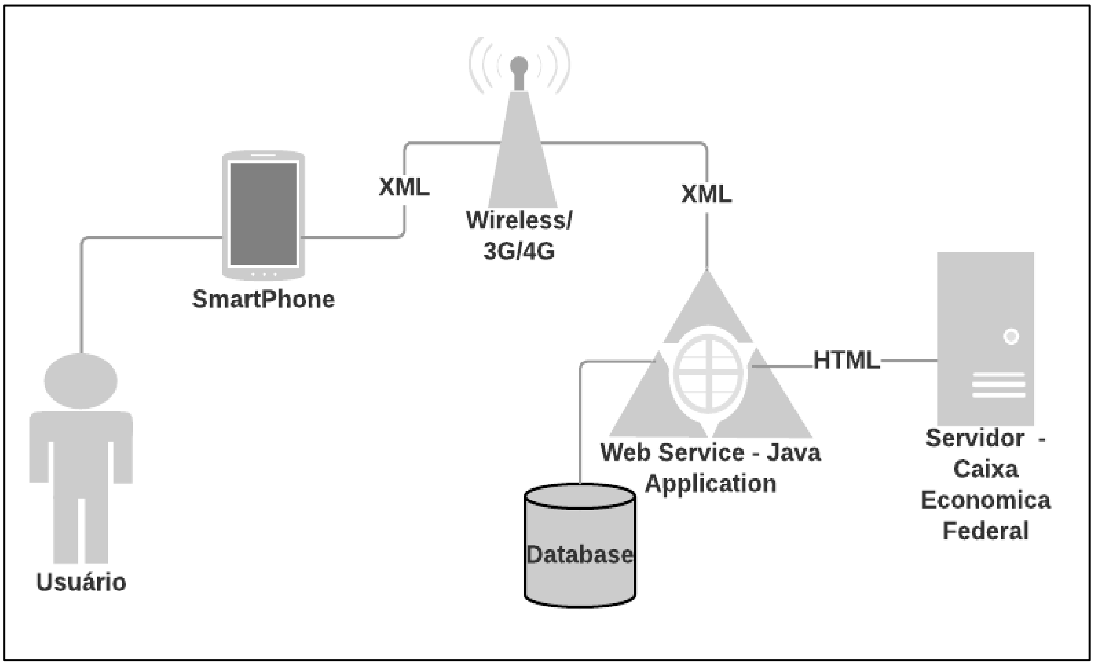

Figura 9 – Arquitetura geral do aplicativo.
Fonte: Autoria própria.

A segunda parte do projeto é uma aplicação web desenvolvida em *Java* responsável por prover os Web Services necessários para o aplicativo, a aplicação irá implementar a especificação *JAX-RS* com uso da API *Jersey* para expor os serviços em *Restfull*. Essa segunda aplicação será desenvolvida em Java utilizando a IDE Eclipse e o Servidor Web *Wildfly 11*. Essa aplicação *Java* será responsável por conectar ao site da Caixa Econômica Federal para buscar as informações atualizadas sobre os resultados dos sorteios. Essa comunicação entre a aplicação Java e o site da caixa econômica ser feita sobre o protocolo *HTTP* desta forma a aplicação Java ira interpretar a página *HTML* para extrair as informações relevantes.

3.2 ESPECIFICAÇÃO DOS REQUISITOS

Nesta seção são apresentados os requisitos funcionais e não funcionais, que foram obtidos após análise das avaliações e comentários feitos na Play Store em aplicativos semelhantes ao proposto nesta monografia, bem como na análise das funcionalidades dos aplicativos Ganhe Mega-Sena e Resultados Mega-Sena descritos no capitulo de referencial teórico.

3.2.1 Requisitos funcionais

Os requisitos funcionais do aplicativo abordados nesta monografia são:

- **Exibir o último resultado da Mega-Sena:** O aplicativo deverá apresentar uma tela com as informações atualizadas sobre o último resultado da Mega-Sena de acordo com resultado exibido no site da Caixa Econômica Federal.
- **Visualizar últimos concursos:** O aplicativo deverá permitir que o usuário possa visualizar os últimos concursos através de uma tela especifica com opções de pesquisa para filtrar o concurso desejado.
- **Gerar números aleatoriamente:** O aplicativo deve permitir a geração de números aleatoriamente semelhantes ao jogo da Mega-Sena surpresinha no qual os números são escolhidos de forma automática.
- **Exibir estatísticas sobre os sorteios:** O aplicativo deve exibir números e/ou gráficos estatísticos sobre os sorteios da Mega-Sena

- **Manter dados das apostas:** O sistema deve apresentar funcionalidade para o usuário salvar o bilhete apostado através da câmera ou teclado, com o intuído de manter o histórico de apostas do usuário.
- **Gerenciar Bolão:** O sistema deve permitir que o usuário gerencie um bolão da Mega-Sena através do cadastro de participantes que serão adicionados a partir dos contatos da agenda do usuário e do cadastro dos bilhetes apostados através da câmera ou teclado.

3.2.2 Requisitos não funcionais

Os requisitos não funcionais que serão implementados no protótipo de aplicativo proposto nesta monografia são:

- Realizar o acesso à internet para buscar informações atualizadas de forma eficiente a fim de não prejudicar o consumo de dados.
- A interface do usuário deverá ser intuitiva, a fim de permitir o usuário identificar com facilidade as funcionalidades.
- A interface do usuário deve seguir as melhores práticas conforme recomendação do Android Developers (*Best Practices for User Interface, 2017*).
- O aplicativo fará a comunicação com Web Services *Restfull* para obter informações atualizadas sobre a Mega-Sena.
- O aplicativo deverá armazenar informações relevantes em um servidor remoto.
- A versão mínima do Android que o aplicativo será compatível será a 4.2 que também é conhecida por J*elly Bean.*

3.2.3 Detalhamento dos Requisitos

Nesta seção são apresentados os requisitos no formato de história de usuário que é uma pratica utilizada nas metodologias ágeis para detalhar de forma curta e simples as necessidades do cliente a serem desenvolvidas. Para enriquecer o detalhamento dos requisitos será utilizado também alguns diagramas UML nas principais funcionalidades.

História de usuário 1	Consultar último resultado da Mega-Sena
Descrição	Eu como usuário do aplicativo, gostaria de consultar o ultimo resultado da Mega-Sena, para saber se houve ganhadores, a quantidade de ganhadores e os valores que serão pagos em cada modalidade.
Critério de aceite	Exibir informações sobre o ultimo resultado de forma correta e atualizada com o exibido no site da Caixa Econômica.

Quadro 2 – História de usuário 1.
Fonte: Autoria Própria.

História de usuário 2	Visualizar últimos concursos
Descrição	Eu como usuário do aplicativo, posso visualizar de forma semelhante a um relatório os últimos concursos realizados, para que eu saiba informações sobre os últimos concursos, como por exemplo, há quantos concursos o prêmio está acumulado ou quais foram os valores pagos nos últimos concursos.
Critério de aceite	Exibir funcionalidade de visualizar os últimos concursos, de forma semelhante a um relatório, com opções de filtro por datas.

Quadro 3 – História de usuário 2.
Fonte: Autoria Própria.

História de usuário 3	Realizar a geração de apostas aleatoriamente
Descrição	Eu como usuário do aplicativo, gostaria de realizar a geração de números de forma aleatória, para pode realizar a aposta de forma semelhante ao formato de jogo surpresinha.
Critério de aceite	O aplicativo deverá apresentar funcionalidade que realize a geração de números de forma aleatória com opção de quantidade de números.

Quadro 4 – História de usuário 3.
Fonte: Autoria Própria.

História de usuário 4	**Apresentar informações estatísticas sobre os números sorteados**
Descrição	Eu como usuário do aplicativo, gostaria de consultar informações estatísticas sobre os números sorteados, para que possa ter conhecimento se existe alguma tendência ou algum número que é sorteado mais ou menos vezes.
Critério de aceite	O aplicativo deverá apresentar a quantidade de vezes que o número foi sorteado. O aplicativo deverá apresentar também o último concurso que o número foi sorteado e a quantidade de concursos que o número não é sorteado.

Quadro 5 – História de usuário 4.
Fonte: Autoria Própria.

História de usuário 5	**Cadastrar aposta simples.**
Descrição	Eu como usuário do aplicativo, gostaria de realizar o cadastro das minhas apostas, de forma que eu consiga manter o número do sorteio, a data e os números apostados, para que seja possível manter um histórico das minhas apostas e acompanhar os números apostados e os números sorteados.
Critério de aceite	O aplicativo deverá permitir que o usuário cadastre o bilhete apostado através de um formulário com campos de data, número do sorteio, e com opção de tirar uma foto do bilhete com a câmera do celular.

Quadro 6 – História de usuário 5.
Fonte: Autoria Própria.

História de usuário 6	**Cadastrar bolão da Mega-Sena.**
Descrição	Eu como usuário do aplicativo, gostaria de gerenciar um bolão da Mega-Sena através do cadastro das informações relevantes ao bolão, como o nome dos participantes através do uso dos contatos da agenda, número de jogos e valor atual da aposta, e também com opção de informar os números apostados ou tirar foto do bilhete da aposta. Para que seja possível manter controle sobre os bolões realizados e os participantes envolvidos.
Critério de aceite	O aplicativo deverá exibir um formulário para inclusão de um novo bolão, com os campos de data, número do sorteio, data do sorteio, valor do bilhete, número de apostas por bilhete, números apostados ou fotos dos bilhetes apostados e também lista de participantes através do uso da agenda do celular.

Quadro 7 – História de usuário 6.
Fonte: Autoria Própria.

3.3 CASO DE USO

Nesta seção serão apresentados os casos de usos que estão associados às histórias de usuário. Na figura 7 é apresentando o diagrama de caso de uso do aplicativo proposto nesta monografia. Na figura 8 é apresentado o diagrama de caso de uso do Web Service que é a aplicação secundária desta monografia. Posteriormente, os casos de usos são descritos em um nível mais elevado de detalhes.

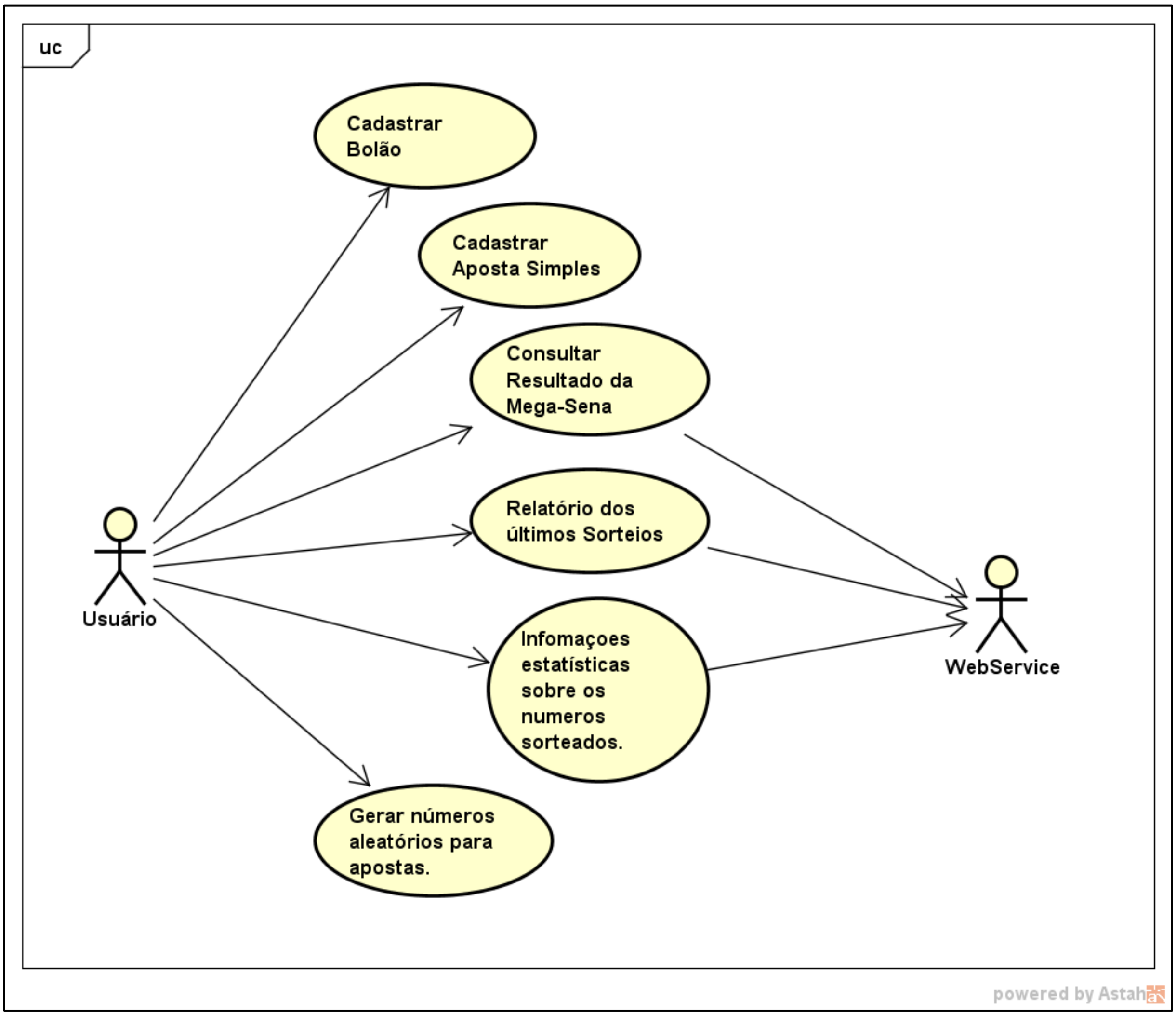

Figura 10 – Diagrama de Caso de Uso do Aplicativo.
Fonte: Autoria própria.

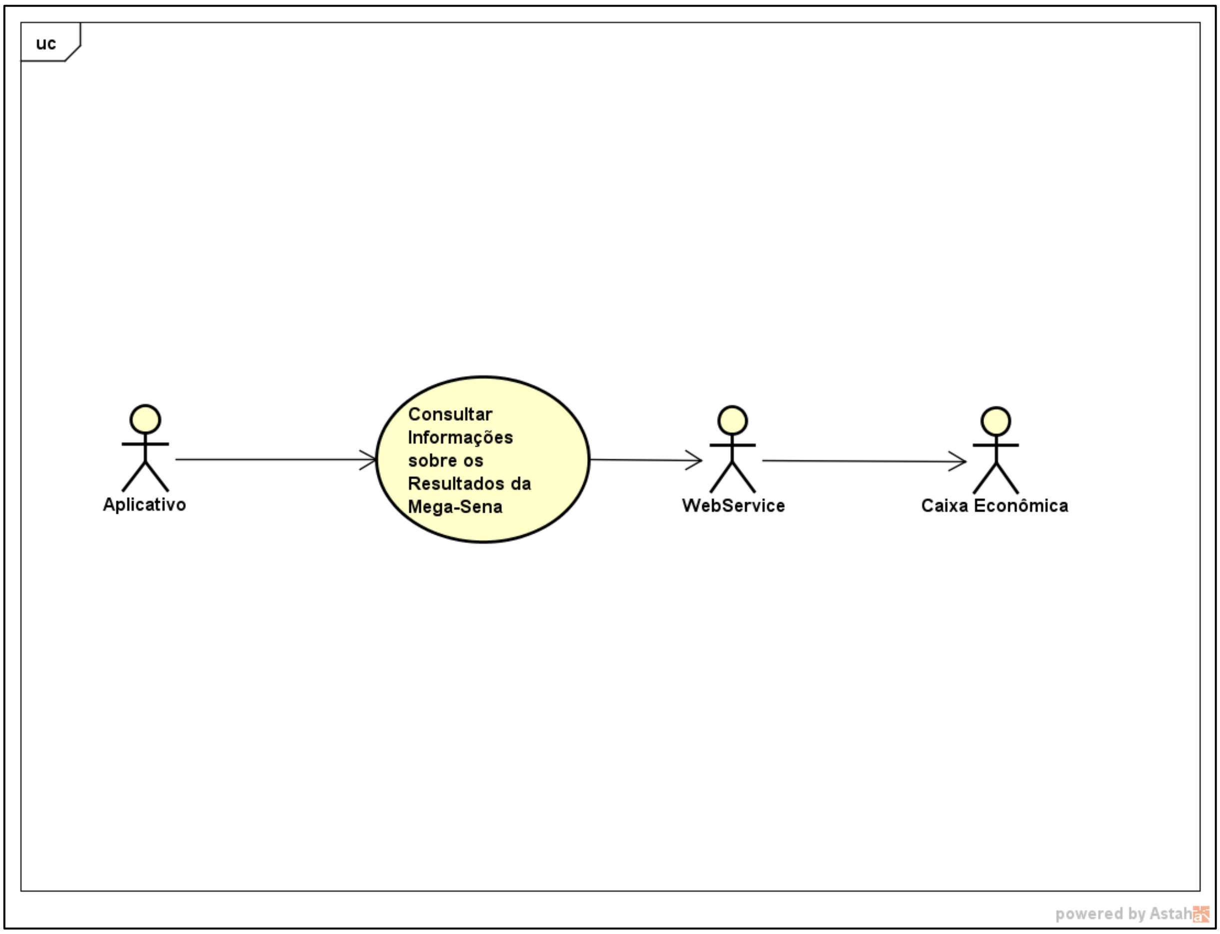

Figura 11 – Diagrama de caso de uso do Web Service.
Fonte: Autoria própria.

3.3.1 Caso de Uso – Cadastrar Aposta Simples

No quadro 8 podemos observar um caso de uso chamado "Cadastrar Aposta Simples" que referencia a história de usuário 5.

Nome	Cadastrar aposta Simples
Atores	1. Usuário
Fluxo Básico	1. O usuário clica no botão do menu "Minhas Apostas" 2. O aplicativo apresenta a tela de minhas apostas. 3. O usuário clica no botão nova aposta. 4. O aplicativo apresenta formulário de cadastro. 5. O usuário preenche os campos: Número do sorteio, quantidade de apostas, preço unitário da aposta, números apostados. 6. O usuário clica em "Salvar".

	7. Aplicativo verifica a regra N1 em caso positivo informa que dados foram “Salvos com sucesso”. 7.1 Caso a regra N1 não seja atendida sistema exibe mensagem de alerta e o caso de uso não é encerrado.
Regra N1	1. Os campos número do sorteio, quantidade de apostas, preço unitário da aposta e números apostados são campos obrigatórios. 2. Em caso negativo exibir mensagens de alerta com o texto “Campos obrigatórios não preenchidos”.
Pré-Condições	1.O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1. O aplicativo salva os dados da aposta.

Quadro 8 – Descrição de caso de uso – Manter apostas.
Fonte: Autoria própria.

3.3.2 Caso de uso – Cadastrar Bolão

O caso de uso “Cadastrar Bolão” demonstrado no quadro 9, faz referência a história de usuário 6.

Nome	Cadastrar bolão.
Atores	1. Usuário.
Fluxo Básico	1. O usuário clica no botão do menu “Meus Bolões”. 2. O aplicativo apresenta a tela com os bolões do usuário. 3. O usuário clica no botão “Novo bolão’. 4. O aplicativo apresenta formulário de cadastro. 5. O usuário preenche os campos: Número do sorteio, quantidade de apostas, preço unitário da aposta, números apostados. 7. O usuário clica no botão “Cadastrar participantes”. 8. O aplicativo apresenta a lista de contatos para seleção. 9. O usuário seleciona os contatos desejados e clica em “OK”. 10. O aplicativo apresenta novamente a tela de cadastro. 11. O usuário clica no botão “Salvar”.

	12. O aplicativo verifica a regra N1 em caso positivo informa que dados foram “Salvos com sucesso”. 12.1 Caso a regra N1 não seja atendida sistema exibe mensagem de alerta e o caso de uso não é encerrado.
Regra N1	1. Os campos número do sorteio, quantidade de apostas, preço unitário da aposta, números apostados e aos menos dois participantes cadastrados são itens obrigatórios. 2. Em caso negativo exibi mensagem de alerta com o texto “Campos obrigatórios não preenchidos”.
Pré-Condições	1. O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1. O aplicativo salva os dados do bolão.

Quadro 9 – Descrição de caso de uso – Gerenciar bolão.
Fonte: Autoria própria.

3.3.3 Caso de uso – Consultar resultado da Mega-Sena

O caso de uso “Consultar resultado da Mega-Sena", faz referência a história de usuário 1.

Nome	Consultar resultado da Mega-Sena
Atores	1. Usuário 2. Aplicação secundária.
Fluxo Básico	1. O usuário clica na opção do menu “Último resultado da Mega-Sena”. 2. O aplicativo consulta a aplicação secundaria para buscar dos dados do último sorteio. 3. O aplicativo apresenta a tela com o resultado da Mega-Sena.
Pré-Condições	1. O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1. As informações são exibidas de forma coerente ao disponibilizado no site da Caixa Econômica Federal.

Quadro 10 – Descrição de caso de uso – Consultar resultado da Mega-Sena.
Fonte: Autoria própria.

3.3.4 Caso de uso – Relatório dos últimos sorteios

O caso de uso “Relatório dos últimos sorteios", faz referência a história de usuário 2.

Nome	Relatório dos últimos sorteios
Atores	1. Usuário 2. Aplicação secundária.
Fluxo Básico	1. O usuário clica na opção do menu “Buscar Concurso”. 2. O aplicativo consulta a aplicação secundária para buscar dos dados dos últimos concursos. 3. O aplicativo apresenta a tela com opções de pesquisa dos concursos já realizados da Mega-Sena.
Pré-Condições	1. O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1. Informações são exibidas de forma coerente ao disponibilizado no site da Caixa Econômica Federal.

Quadro 11 – Descrição de caso de uso – Relatório dos últimos sorteios.
Fonte: Autoria própria.

3.3.5 Caso de uso – Informações estatísticas sobre os números sorteados

O caso de uso “Informações estatísticas sobre os números sorteados", faz referência a história de usuário 4.

Nome	Informações estatísticas sobre os números sorteados
Atores	1. Usuário 2. Web Service.
Fluxo Básico	1. O usuário clica na opção do menu “Estatísticas dos Números”. 2. O aplicativo consulta o Web Service para buscar os dados dos últimos concursos. 3. O aplicativo apresenta a tela com as informações estatísticas sobre os números.

Pré-Condições	1. O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1. Informações são exibidas de forma coerente ao disponibilizado no site da Caixa Econômica Federal.

Quadro 12 – Descrição de caso de uso – Informações estatísticas sobre os números sorteados.
Fonte: Autoria própria.

3.3.6 Caso de uso – Gerar números aleatórios para apostas.

O caso de uso “Gerar números aleatórios para apostas.", faz referência a história de usuário 3.

Nome	Gerar números aleatórios para apostas.
Atores	1. Usuário
Fluxo Básico	1. O usuário clica na opção do menu “Gerar Surpresinha”. 2. O aplicativo apresenta a tela para geração de números aleatórios. 3. O usuário escolhe a quantidade de números e clica no botão “Gerar”. 4. O aplicativo apresenta os números gerados aleatoriamente.
Pré-Condições	1. O Smartphone deve possuir acesso à internet.
Pós-Condições	1. Os números são gerados de forma aleatória.

Quadro 13 – Descrição de caso de uso – Gerar números aleatórios para apostas.
Fonte: Autoria própria.

3.3.7 Caso de uso do Web Service – Consultar informações sobre os resultados da Mega-Sena.

O caso de uso “Consultar informações sobre os resultados da Mega-Sena.", é o caso de uso do Web Service que é necessário para executar os casos de usos do aplicativo que consultam informações sobre a Mega-Sena. Neste caso de uso são realizadas as consultas ao site da Caixa Econômica Federal para que o aplicativo tenha informações atualizadas sobre os últimos concursos realizados.

Nome	Consultar informações sobre os resultados da Mega-Sena.
Atores	1. Aplicativo 2. Web Service 3. Caixa Econômica
Fluxo Básico	1. O aplicativo solicita informações sobre os resultados da Mega-Sena para o WebService. 1.1 Caso o Web Service não responda, o aplicativo exibe mensagem informando que o serviço está indisponível no momento. 2. O Web Service interpreta a solicitação e consulta o site da caixa Econômica Federal. 2.1 Caso o site da Caixa Econômica não responda o Web Service devolve mensagem de erro para o Aplicativo informando que a Caixa Econômica está indisponível no momento. 3. O site da Caixa Econômica Federal responde a solicitação. 4. O Web Service interpreta a resposta, transforma a informação e devolve para que o Aplicativo possa continuar a execução.
Pré-Condições	1. O Smartphone deve possuir acesso à internet através de alguma rede.
Pós-Condições	1.A informação solicitada é devolvida com sucesso.

Quadro 14 – Descrição de caso de uso – Consultar informações dos resultados da Mega-Sena.
Fonte: Autoria própria.

3.4 DIAGRAMA DE CLASSES

O diagrama de classes tem por objetivo demonstrar a estrutura e as relações das classes, que são a representação dos objetos que fazem parte do software a ser desenvolvido. Nesta monografia serão apresentadas a classes e os métodos principais do aplicativo a ser desenvolvido. Os métodos de acesso aos atributos privados, que são chamados de métodos *get* e *set* serão omitidos, a fim de evitar que o diagrama fique ilegível.

3.4.1 Diagrama de classe - Aplicativo

Na figura 12 é demonstrado o diagrama de classe das principais classes do aplicativo. Na classe *MainActivity* foi implementado o método *onNavigationItemSelected* da interface *OnNavigationItemSelectedListener* para que seja possível capturar os eventos de seleção de opções do menu lateral.

As classes do pacote *fragments* representam a camada de *controller*, ou seja, responsáveis pelas regras de negócio e a comunicação entre as telas e o modelo de dados, já o modelo de dados é representado pelas classes dos pacotes *DAO* e *entity*.

No momento em que alguma classe do pacote *fragments* é criada, ela irá criar uma instancia da DAO correspondente, essa classe DAO utiliza uma instancia da classe *DbGateway* para realizar a conexão com o banco de dados, e com isso, prover o acesso aos dados.

Umas das principais classes deste diagrama é a classe *BolaoEntity*, onde o relacionamento com as classes *ConcursoEntity, ParticipantesEntity* e *BilheteApostaEntity* são agregações do tipo composição, porque existe uma relação forte entre as classes, por exemplo, não faz sentido existir participantes se não existir um bolão.

3.5 DIAGRAMA DE SEQUENCIA

Os diagramas de sequência têm o objetivo de demonstrar o fluxo sistêmico de uma funcionalidade, exibindo como as mensagens entre os objetos são trocadas no decorrer do tempo para a realização de uma determinada operação do sistema. Nesta monografia serão apresentados os diagramas de sequência das principais funcionalidades do aplicativo, utilizando a ferramenta de modelagem *Astah.*

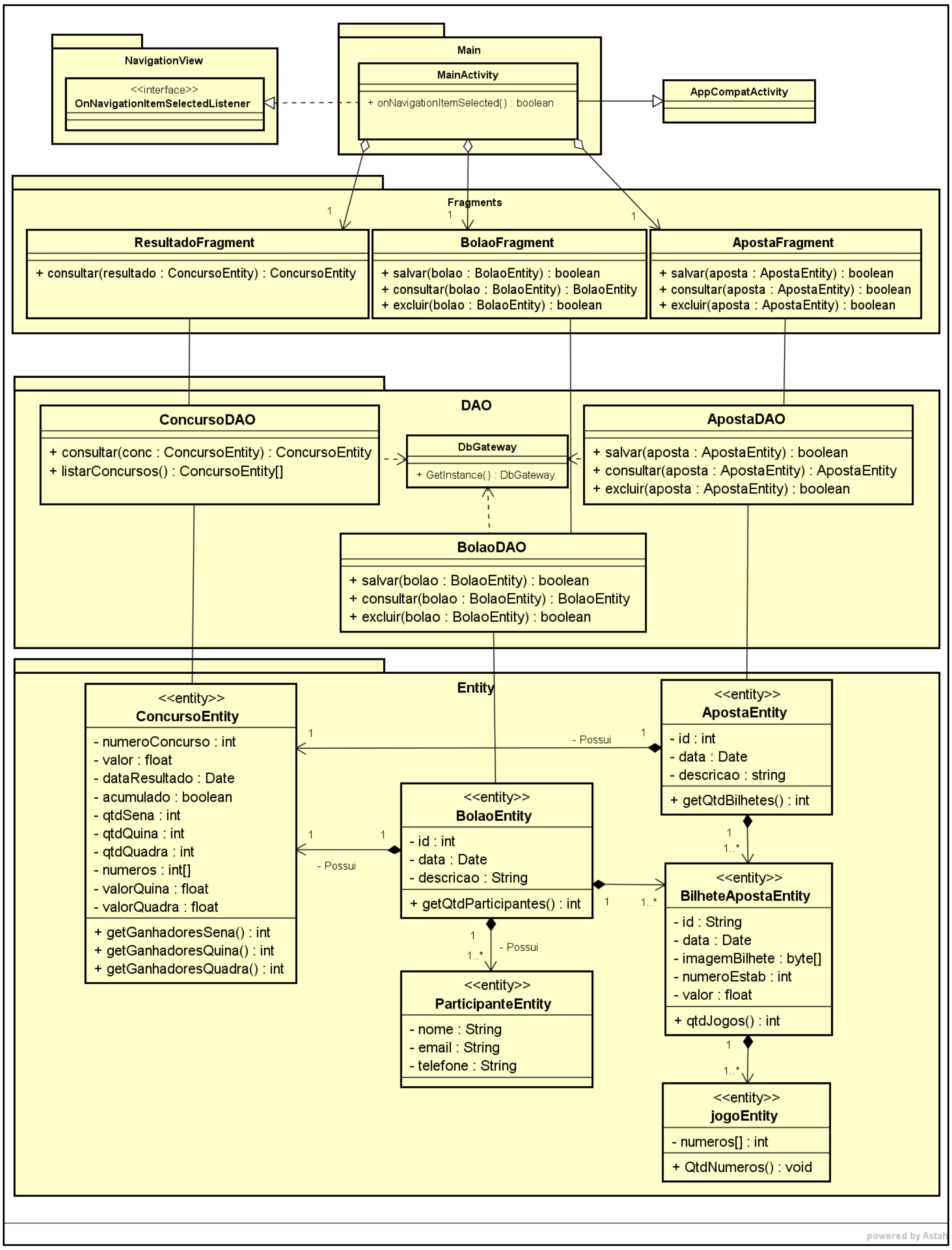

Figura 12 – Diagrama de classes do aplicativo.
Fonte: Autoria própria.

3.5.1 Diagrama de sequência – Consultar Resultado Mega-Sena

Na figura 13 são apresentadas as trocas de mensagens necessárias para realizar o caso de uso de “Consulta do último resultado da Mega-Sena”.

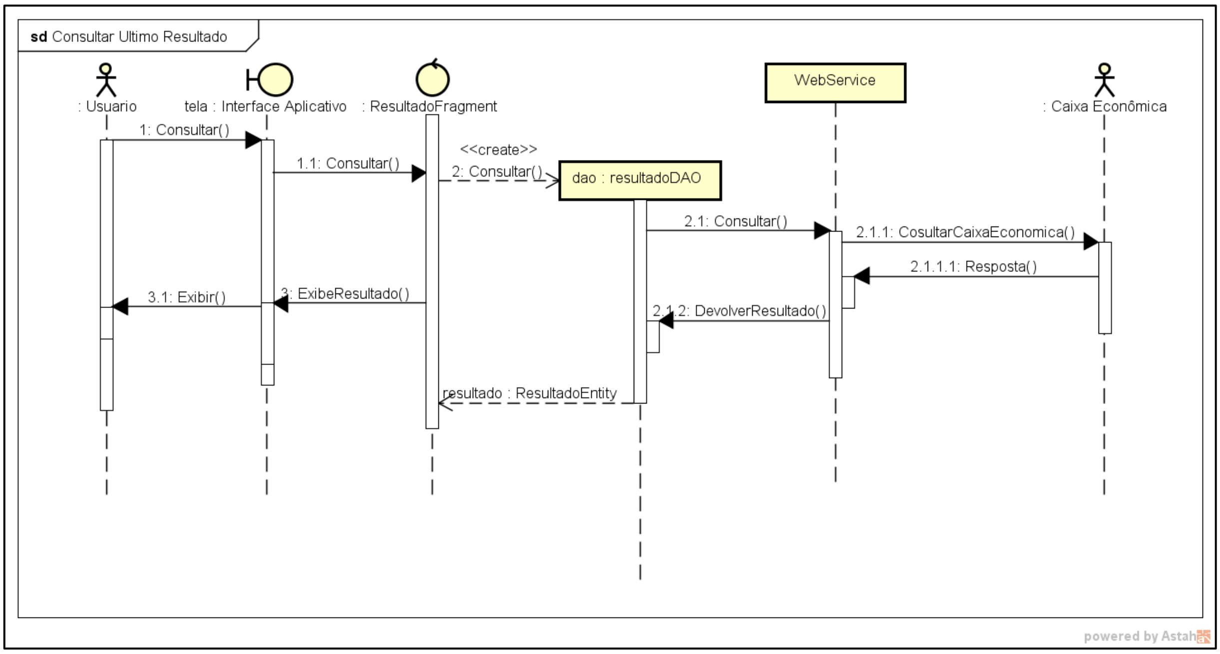

Figura 13 – Diagrama de sequência – Consultar último resultado.
Fonte: Autoria própria

3.5.2 Diagrama de sequência – Cadastrar Bolão

Na figura 14 são apresentadas as trocas de mensagens necessárias para realizar o caso de uso “Cadastrar bolão”.

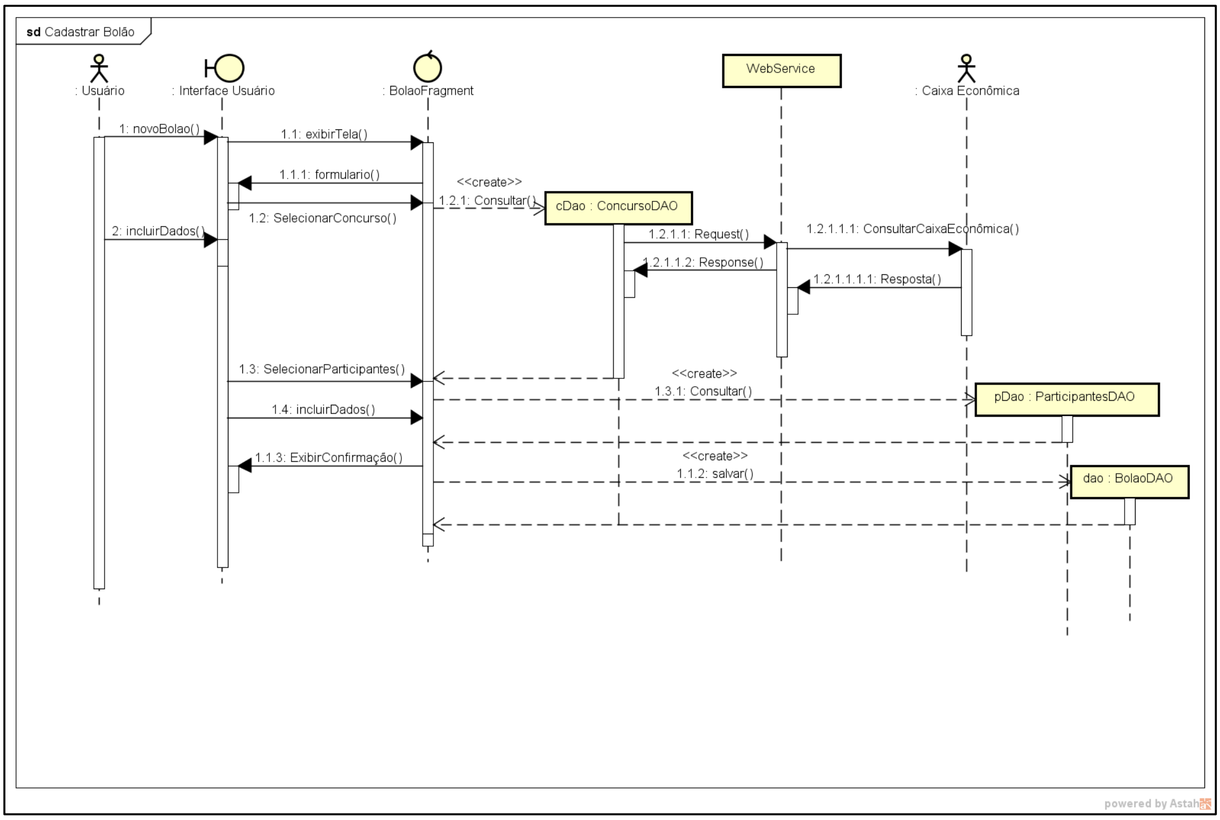

Figura 14 – Diagrama de sequência - Cadastrar Bolão.
Fonte: Autoria própria.

3.5.3 Diagrama de sequência – Cadastrar Aposta

Na figura 15 são apresentadas as trocas de mensagens necessárias para realizar o caso de uso “Cadastrar aposta”.

3.6 PROJETO DE INTERFACE

Nesta seção será apresentado o projeto de interface das principais telas do aplicativo proposto nesta monografia. Nas seções seguintes será demonstrado um protótipo de design de interface de alta fidelidade que foi construído utilizando a ferramenta *JustiMind.*

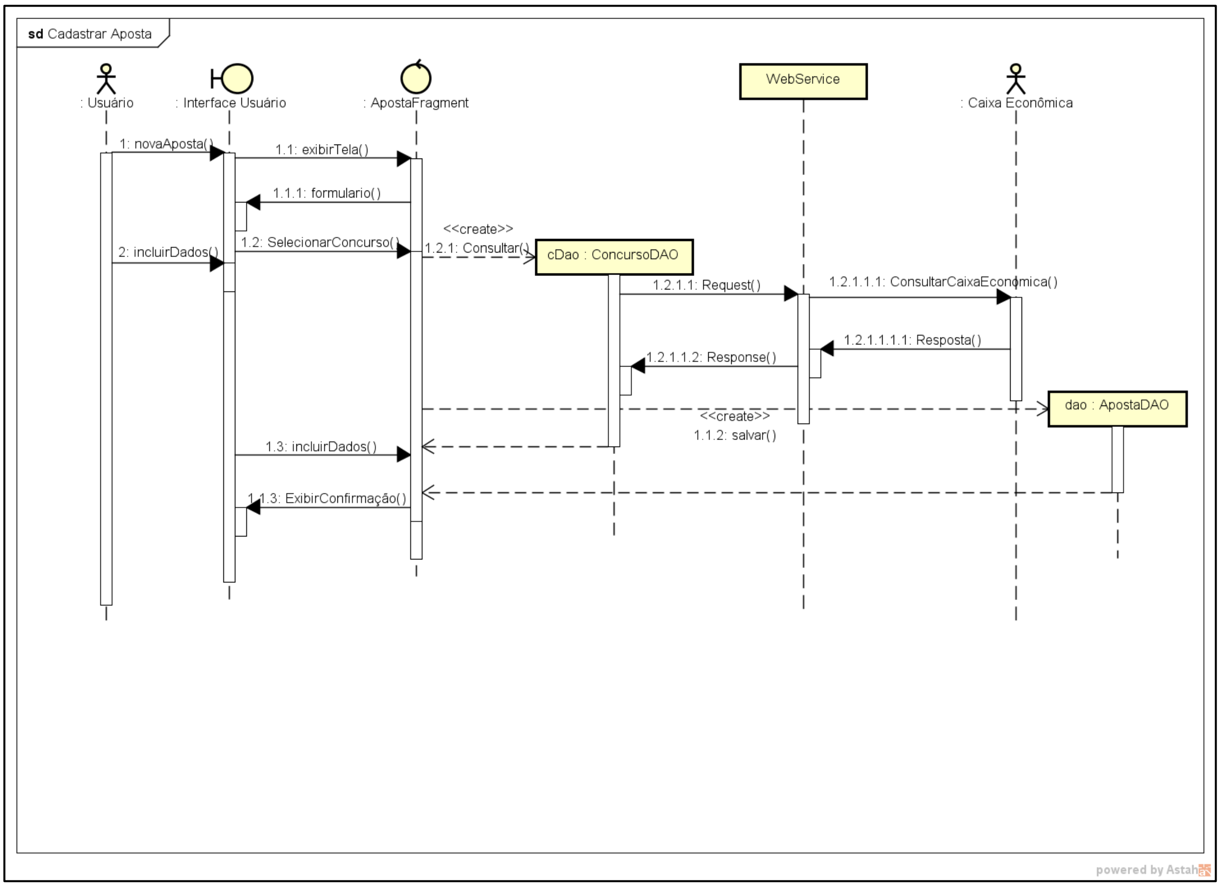

Figura 15 – Diagrama de sequência – Cadastrar aposta.
Fonte: Autoria própria.

3.6.1 Menu lateral

Seguindo as boas práticas para implementar uma navegação efetiva, foi criado um menu lateral que é chamado de *Navigation Drawer* – navegação de gaveta – este menu exibe as principais opções do painel de navegação do aplicativo na borda esquerda da tela. O menu fica oculto a maior parte do tempo, mas é apresentado quando o usuário desliza o dedo a partir da borda esquerda da tela, ou, quando o usuário clica no ícone no nível superior do aplicativo. (ANDROID DEVELOPERS, 2017).

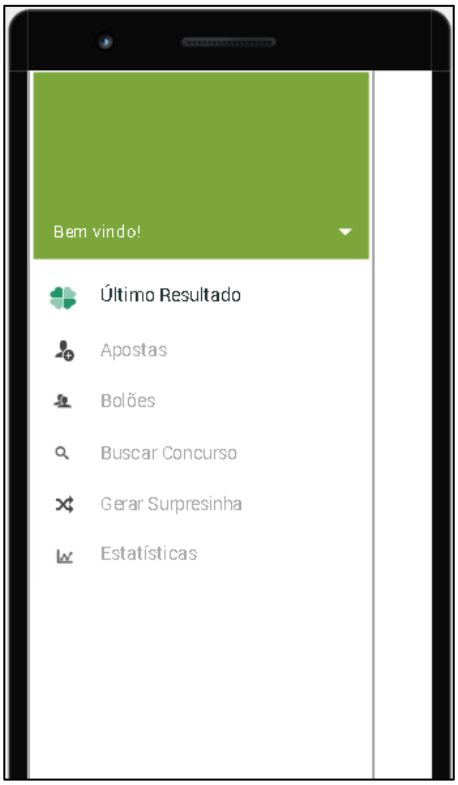

Figura 16 – Protótipo de alta fidelidade – Menu lateral.
Fonte: Autoria própria.

3.6.2 Tela de Lista de bolões

Na figura 17 é apresentada a tela de lista de bolões, onde o usuário poderá consultar os últimos bolões cadastrados, ou, iniciar o cadastro de um novo bolão através do botão de adicionar que está posicionado na parte superior da tela.

Figura 17 – Protótipo de alta fidelidade – Lista de bolões.
Fonte: Autoria própria.

3.6.3 Tela de Cadastro de bolões

Na figura 18 é apresentado o protótipo da tela de cadastro de bolão, onde o usuário poderá iniciar o processo de cadastro de um novo bolão.

Figura 18 – Protótipo de alta fidelidade – Cadastro de bolão.
Fonte: Autoria própria.

3.7 IMPLEMENTAÇÃO

Nesta seção será apresentado o resultado final da implementação do protótipo de aplicativo para smartphones *Android.*

3.7.1 Descrição do projeto do aplicativo

O aplicativo foi desenvolvido de forma nativa para *Android* utilizando a IDE *Android Studio,* na figura 19 podemos observar a estrutura dos pacotes do projeto do aplicativo. O menu da aplicação foi implementado utilizando o padrão *Navigation Drawer* – navegação de gaveta – com isso a estrutura de navegação de telas foi implementada utilizando *Fragments* que se comportam como seções modulares da atividade *e* possuem ciclo de vida próprio, desta forma a transição de telas pode ser feita de forma mais simples, na figura 20 é apresentado o código que substitui o fragmento atual pelo fragmento da tela desejada.

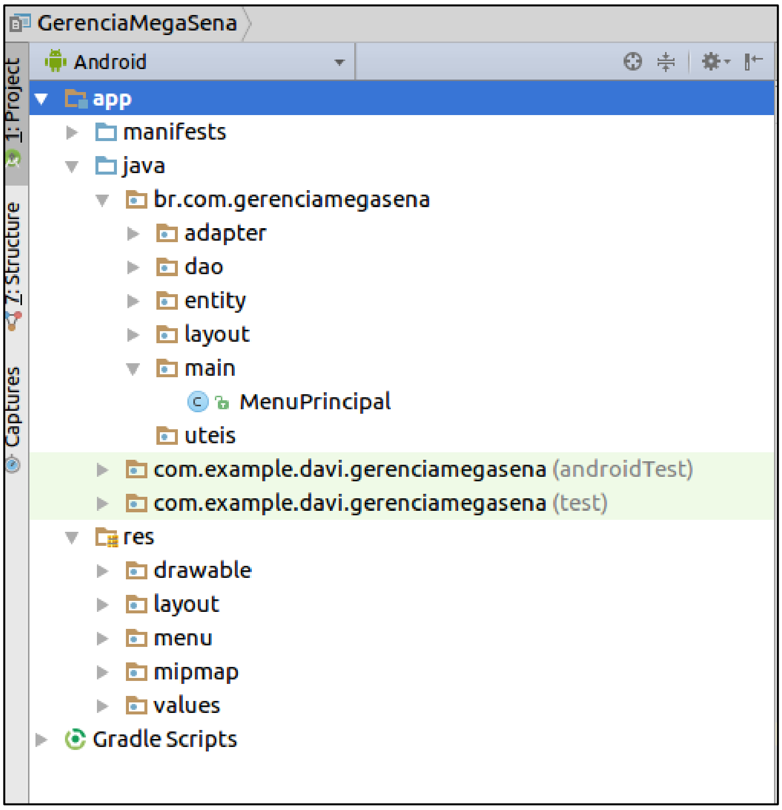

Figura 19 – Estrutura de pacotes do projeto *Android*
Fonte: Autoria própria.

```
    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();
        FragmentManager fragmentManager = getFragmentManager();

        if (id == R.id.resultMegaSena) {
            fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new MegaSenaFragment()).commit();

        } else if (id == R.id.gerarApostas) {
            fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new FragmentGerarNumeros()).commit();
        } else if (id == R.id.minhasApostas) {
            fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new FragmentApostas()).commit();
        } else if (id == R.id.meusBoloes) {
            fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new FragmentBoloes()).commit();

        }else if(id == R.id.estatisticas){
            fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new FragmentEstatisticas()).commit();
        }else if(id == R.id.buscarConcursos){
        fragmentManager.beginTransaction().replace(R.id.conteudo_fragmento, new FragmentConcursos()).commit();
        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Figura 20 – Código de transição de fragmentos.
Fonte: Autoria própria.

3.7.2 Implementação do acesso ao Web Service

Para realizar a comunicação com o servidor de serviços *Rest,* foram utilizadas as classes *java.net.HttpURLConnection* e *java.net.URL* que são classes nativas da linguagem *Java* desde a *JDK1.1.* Essas classes em conjunto conseguem realizar uma requisição utilizando o protocolo *HTTP.* Utilizando o método *getInputStream* da classe *HttpURLConnection* é possível recuperar a resposta do servidor e utilizá-la da maneira que for conveniente.

Neste projeto, como a resposta do Web Service é enviada em um formato de *JSON*, foi necessário utilizar uma biblioteca para realizar a transformação da resposta em um objeto do tipo *org.json.JSONArray,* esta biblioteca está na API do Android desde a versão 1. A escolha desta biblioteca de Objetos *JSON* foi escolhida principalmente pela simplicidade e por ela já estar na SDK de desenvolvimento do Android, não sendo necessário utilizar uma API de terceiros. Na figura 21 é apresentado um trecho do código responsável pelo acesso ao web service, onde a instância da classe *JSONArray* já é criada com uma String(Caracteres) contendo a resposta do web service, com isso, simplifica o acesso aos objetos do tipo de *JSON* que são recuperados após iterar o objeto da classe JSONArray.

```
protected Void doInBackground(Void... arg0) {
    HttpHandler sh = new HttpHandler();

    concursoDAO = new ConcursoDAO(getApplicationContext());
    ConcursoEntity ultimoConcurso = concursoDAO.buscarUltimoConcurso();
    int intUltimoConcurso = 0;

    if(ultimoConcurso != null && ultimoConcurso.getNumeroConcurso() > 0)
        intUltimoConcurso = ultimoConcurso.getNumeroConcurso();


    // Making a request to url and getting response
    String url = "http://192.168.43.43:8080/RestFulMegaSena/rest/concursos/maior/" + intUltimoConcurso;
    String jsonStr = sh.makeServiceCall(url);

    Log.e(TAG, "Response from url: " + jsonStr);
    if (jsonStr != null) {
        try {
            JSONArray jsonArray = new JSONArray(jsonStr);

            for (int i= 0; i < jsonArray.length(); i++){
                JSONObject jsonObj = jsonArray.getJSONObject(i);

                concurso = new ConcursoEntity(jsonObj.getInt("numeroConcurso"));
                concurso.setAcumulado(jsonObj.getBoolean("acumulado"));
                concurso.setQtdSena(jsonObj.getInt("ganhadoresSena"));
                concurso.setQtdQuina(jsonObj.getInt("ganhadoresQuina"));
                concurso.setQtdQuadra(jsonObj.getInt("ganhadoresQuadra"));
                concurso.setValor(jsonObj.getString("valor"));
                concurso.setValorQuina(jsonObj.getString("valorQuina"));
                concurso.setValorQuadra(jsonObj.getString("valorQuadra"));
                concurso.setValorAcumulado(jsonObj.getString("valorAcumulado"));
                concurso.setValorAcumuladoMegaVirada(jsonObj.getString("valorAcumuladoMegaVirada"));
                concurso.setValorEstimativa(jsonObj.getString("valorEstimativa"));
                concurso.setDataResultado(jsonObj.getString("dataResultado"));
```

Figura 21 – Código de acesso ao Web Service Rest.
Fonte: Autoria própria.

3.7.3 Funcionalidades e imagens do aplicativo

Nesta seção serão apresentadas as principais funcionalidades desenvolvidas bem como as telas do aplicativo proposto nesta monografia.

3.7.3.1 Menu

A figura 22 apresenta o menu do aplicativo, onde usuário poderá escolher em qual tela deseja navegar. A tela inicial do aplicativo será a tela com o último resultado da Mega-Sena conforme demonstrado na figura 23, essa tela é também a primeira opção do menu. Com isso o usuário já poderá ter as informações sobre o ultimo sorteio da Mega-Sena assim que o aplicativo for iniciado.

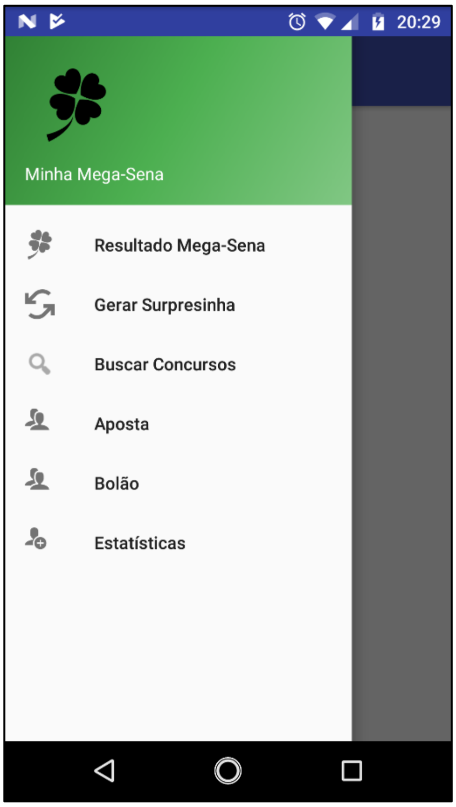

Figura 22 – Menu do Aplicativo.
Fonte: Autoria própria.

Figura 23 – Tela do Último resultado da Mega-Sena.
Fonte: Autoria própria

3.7.3.2 Bolão

Selecionada a opção “Bolão” no menu do aplicativo, será apresentada uma tela com a lista dos últimos bolões cadastrados (Figura 24), para que o usuário possa consultar seus bolões. Nesta tela é possível que o usuário inicie o cadastro de um novo bolão através do botão “Adicionar” representado pela cruz verde no lado superior direito da tela conforme figura 25.

Figura 24– Tela de lista de bolões.
Fonte: Autoria própria.

Após clicar no botão “Adicionar”, é apresentada a tela de cadastro do bolão, onde é possível preencher os campos de descrição e concurso, o campo concurso é uma lista de seleção que já vem carregada com os últimos concursos da Mega-Sena. O botão “Incluir Participantes” irá apresentar a lista de contatos do aparelho para a seleção dos participantes, após a seleção o contato irá aparecer na lista de participantes que está posicionada logo abaixo do botão, após isso o usuário poderá clicar no botão “Incluir Bilhetes” para capturar imagens dos bilhetes apostados.

Figura 25 – Tela de cadastro de bolão.
Fonte: Autoria própria.

3.7.3.3 Aposta

Selecionada a opção “Aposta” no menu do aplicativo, será apresentada uma tela (Figura 26) com a lista das últimas apostas realizadas pelo usuário, para que usuário possa consultar suas apostas e verificar os detalhes de cada uma. Nesta tela é possível que o usuário inicie o cadastro de uma nova aposta através do botão “Adicionar” representado pela cruz verde conforme figura 25.

Figura 26 – Tela de lista de apostas.
Fonte: Autoria própria.

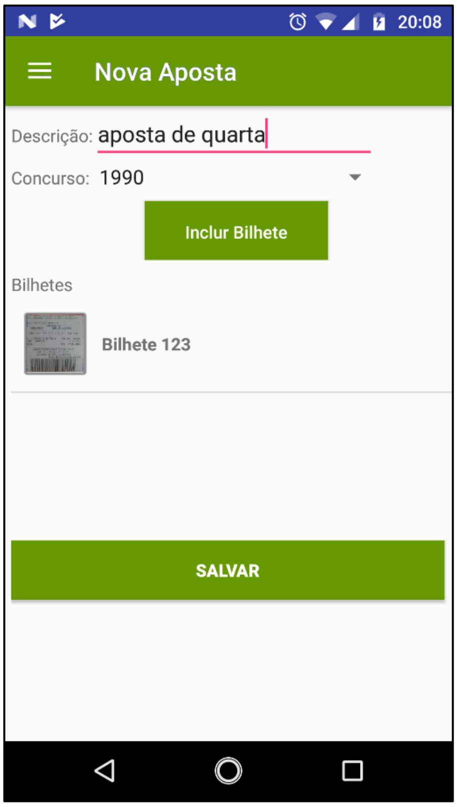

Figura 27 – Tela de cadastro de aposta.
Fonte: Autoria própria.

3.7.3.4 Gerar Surpresinha

Selecionada a opção “Gerar Surpresinha” no menu, é apresentada a tela de geração de números aleatórios – Surpresinha – onde o usuário pode escolher a quantidade de números que deseja que sejam gerados aleatoriamente. Para essa tela foi escolhido um botão seletor parecido com um botão de zoom como podemos observar na figura 28. No momento em que o usuário alterar o valor deste controle, automaticamente os números serão gerados, de acordo com a quantidade escolhida.

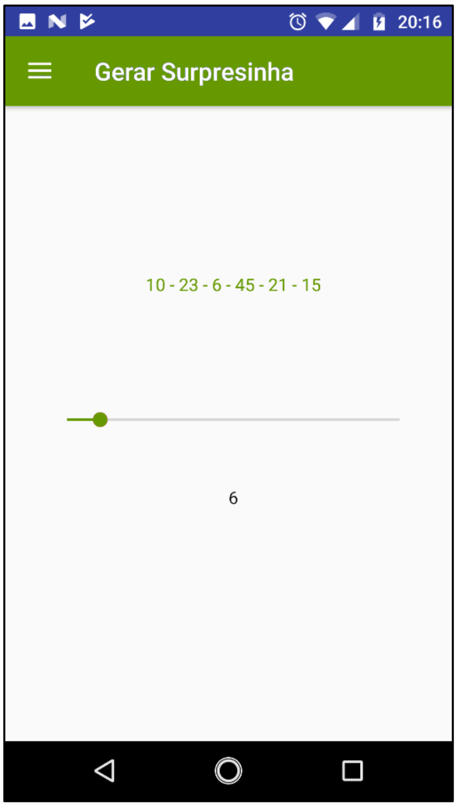

Figura 28 – Tela de gerar surpresinha.
Fonte: Autoria própria.

3.7.3.5 Buscar concursos

Selecionada a opção “Buscar Concursos” é apresentada uma tela com a lista dos últimos concursos conforme apresentado na figura 29. Nesta tela o usuário pode realizar uma busca pelo número do concurso para localizar o concurso desejado.

Figura 29 – Tela de buscar de concursos.
Fonte: Autoria própria.

3.7.3.6 Tela de estatísticas

Quando o usuário seleciona a opção “Estatísticas” no menu do aplicativo, é apresentada a tela com a relação de todos os números possíveis de serem apostados, o usuário poderá clicar em cima do número desejado e com isso será apresentada as informações estatísticas daquele número, conforme apresentado na figura 30.

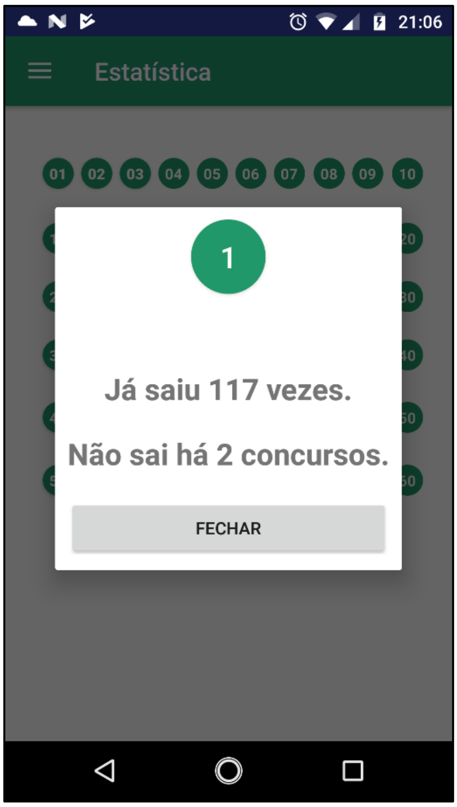

Figura 30 – Tela de estatísticas sobre os números sorteados.
Fonte: Autoria própria.

4 TESTES

Neste capitulo serão apresentados os cenários de testes necessários para garantir a qualidade das funcionalidades desenvolvidas.

4.1 CASOS DE TESTE

Após a conclusão das principais telas do aplicativo será utilizado o método de teste de caixa-preta, que são testes que se concentram nos requisitos funcionais do software, sem focar a estrutura lógica interna da aplicação, com isso, conseguimos assegurar que todos os comportamentos e requerimentos do aplicativo ou de um componente do aplicativo estejam corretos.

Nº do Caso de Teste:	**001**
Localização:	Página inicial / Menu >> Resultado Mega-Sena
Objeto de Teste:	As informações sobre o último resultado da Mega-Sena devem estar coerentes com o apresentado no site da Caixa Econômica Federal.
Tipo de teste:	Caixa-preta
Caso de Teste:	Consultar último resultado da Mega-Sena
Inputs de Teste:	1. N/A.
Procedimento:	1. Abrir a o aplicativo 2. Clicar na opção de Menu “Resultado Mega-Sena”.
Resultado esperado:	1. O aplicativo deverá apresentar o resultado da Mega-Sena de forma coerente.

Quadro 15 – Caso de teste 1.
Fonte: Autoria própria.

Nº do Caso de Teste:	**002**
Localização:	Menu >> Apostas
Objeto de Teste:	Operações referentes ao cadastro de uma nova aposta.
Tipo de teste:	Caixa-preta
Caso de Teste:	Testar funcionalidade de cadastro de apostas.
Inputs de Teste:	1. Informações referentes ao cadastro da aposta.
Procedimento:	1. Clicar no menu “Aposta” 2. Clicar no botão “Incluir”. 3. Preencher todos os campos. 4. Clicar no botão incluir bilhetes. 5. Capturar bilhetes. 6. Clicar no botão “Salvar”.
Resultado esperado:	1. O aplicativo devera salvar corretamente todas as

	informações na base de dados e exibir uma mensagem informado que o cadastro foi realizado com sucesso.

Quadro 16 – Caso de teste 2.
Fonte: Autoria própria.

Nº do Caso de Teste:	**003**
Localização:	Menu >> Bolões
Objeto de Teste:	Operações referentes ao cadastro de um novo bolão.
Tipo de teste:	Caixa-preta
Caso de Teste:	Testar funcionalidade de cadastro de bolões.
Inputs de Teste:	1. Informações referentes ao cadastro do bolão.
Procedimento:	1. Clicar no menu “Bolão” 2. Clicar no botão “Incluir”. 3. Preencher todos os campos. 4. Clicar no botão “Incluir bilhetes”. 5. Capturar bilhetes. 6. Clicar no botão “Incluir participantes” 7. Selecionar participantes. 6. Clicar no botão “Salvar”.
Resultado esperado:	1. O aplicativo devera salvar corretamente todas as informações na base de dados e exibir uma mensagem informado que o cadastro foi realizado com sucesso.

Quadro 17 – Caso de teste 3.
Fonte: Autoria própria.

Nº do Caso de Teste:	**004**
Localização:	Menu >> Gerar Surpresinha
Objeto de Teste:	Operações referentes a geração de números aleatórios
Tipo de teste:	Caixa-preta
Caso de Teste:	Testar geração de números aleatórios.
Inputs de Teste:	1. Quantidade de números a gerar.
Procedimento:	1. Clicar no menu “Gerar Surpresinha”. 2. Selecionar a quantidade de números através do botão deslizante.
Resultado esperado:	1. O aplicativo deverá apresentar os números gerados de acordo com a quantidade solicitada.

Quadro 18 – Caso de teste 4.
Fonte: Autoria própria.

Nº do Caso de Teste:	**005**
Localização:	Menu >> Buscar Concursos
Objeto de Teste:	Operações referentes a busca de informações sobre os últimos resultados da Mega-Sena.
Tipo de teste:	Caixa-preta

Caso de Teste:	Verificação da busca dos últimos resultados da Mega-Sena.
Inputs de Teste:	1. Número do concurso.
Procedimento:	1. Clicar no menu “Buscar Concursos”. 2. Digitar o número do concurso. 3. Clicar no botão “Buscar”.
Resultado esperado:	1. O aplicativo deverá apresentar o resultado possível de acordo com o que foi pesquisado.

Quadro 19 – Caso de teste 5.
Fonte: Autoria própria.

Nº do Caso de Teste:	**006**
Localização:	Menu >> Estatísticas
Objeto de Teste:	Operações referentes as informações estatísticas.
Tipo de teste:	Caixa-preta
Caso de Teste:	Verificar informações estatísticas sobre os números sorteados.
Inputs de Teste:	1. Um número de 1 a 60.
Procedimento:	1. Clicar no menu “Estatísticas”. 2. Informar o número desejado. 3. Clicar no botão “Analisar”
Resultado esperado:	1. O aplicativo deverá apresentar informações estatísticas do número selecionado.

Quadro 20 – Caso de teste 6.
Fonte: Autoria própria.

4.1.1 Conclusão dos testes

Os testes de caixa preta se mostraram essenciais para identificar como usuário irá interagir com o aplicativo, quais botões ele irá clicar, que informações digitará nos campos que precisam ser preenchidos. A interface também foi avaliada com os testes de caixa preta com a intenção de encontrar links que não funcionam, tamanhos de caixa de texto que podem dificultar a interação do usuário e também botões posicionados de forma a dificultar a interação com o usuário. A conclusão é que os resultados dos testes de caixa preta estão de acordo com as especificações e com o que era esperado das funcionalidades.

5 CONCLUSÃO

5.1 RESUMO DO TRABALHO

O protótipo desenvolvido foi uma aplicação Android em linguagem nativa, que se mostrou útil para apostadores da Mega-Sena, tendo como ponto forte a manutenção de um histórico dos jogos do apostador, que podem ser acessados a qualquer momento. O aplicativo dispõe de funcionalidades que facilitam a consulta de informações sobre os resultados da Mega-Sena, sem a necessidade de o usuário ter que abrir o site da Caixa Econômica Federal.

No decorrer da construção da monografia e no desenvolvimento do protótipo uma das maiores dificuldades foi a funcionalidade de realizar a leitura dos jogos por meio de uma API OCR, que não pode ser implementada devido à dificuldade de implementação e o tempo necessário para realizar os ajustes nas API existentes para que possam realizar a leitura dos bilhetes.

Um dos pontos interessantes deste trabalho foi a vasta pesquisa na documentação oficial, tutoriais e fóruns a fim de encontrar as melhores práticas para desenvolver as funcionalidades deste protótipo, como por exemplo, o menu lateral – *Navigation Drawer* – que é uma boa pratica recomendada na documentação oficial do *ANDROID DEVELOPER*, outro exemplo interessante é o uso de *Fragments* que deixam o desenvolvimento da interface muito mais produtivo. Posso afirmar que o autor desta monografia adquiriu novos conhecimentos e aprofundou bastante os conceitos que já possuía certo conhecimento com a construção do protótipo proposto nesta monografia.

5.2 TRABALHOS FUTUROS

Após a conclusão do desenvolvimento do protótipo apresentado nesta monografia, perceberam-se algumas funcionalidades que seriam interessantes para obter um aplicativo mais completo para os apostadores. Como por exemplo, a funcionalidade de leitura dos bilhetes por OCR, que não foi desenvolvida por falta de tempo, e que poderia ser interessante em conjunto com uma funcionalidade que confere os jogos automaticamente. A seguir são apresentadas algumas outras funcionalidades que seriam interessantes:

- Notificação de concursos da Mega-Sena a partir de um determinado valor de prêmio, com isso, o usuário poderá ser avisado que o concurso está acumulado com um valor pré-definido pelo próprio usuário.
- Exibir probabilidade de ganhar conforme a quantidade das apostas.
- Compartilhar informações com outros aplicativos como SMS e WhatsApp.
- Gerenciar o pagamento dos participantes do bolão.
- Permitir que o usuário digite os números da aposta com o teclado.

6 REFERÊNCIAS

ANDROID DEVELOPER, **Android Studio.** Disponível em: <https://developer.android.com/studio/intro/index.html>. Acessado em: 20 set. 2017.

ANDROID DEVELOPER. **Best Practices for User Interface**. Disponível em:<https://developer.android.com/training/best-ui.html>. Acessado em: 28 out. 2017.

ANDROID DEVELOPER, **Fundamentos de aplicativos**. Disponível em: <https://developer.android.com/guide/components/fundamentals.html>. Acessado em: 27 ago. 2017.

CAIXA**.** **Sorte em Números.** Disponível em: <http://www.caixa.gov.br/Downloads/Sorte_em_numeros_2016_PT.pdf>. Acessado em: 25 ago. 2015.

CONFERENCIA WEB. **Arquitetura rest pode garantir um Web ainda mais eficiente**. Disponível em: <http://conferenciaweb.w3c.br/2015/arquitetura-rest-pode-garantir-uma-web-ainda-mais-eficiente/>. Acessado em: 10 out. 2017.

ERL, Thomas. **Introdução às tecnologias Web Services: SOA, SOAP, WSDL e UDDI**. Disponível em: <http://www.devmedia.com.br/introducao-as-tecnologias-web-services-soa-soap-wsdl-e-uddi-parte1/2873>. Acessado em: 20 set. 2017.

FELIPE, Alex. **Criando uma lista com ListView.** Disponível em: <http://blog.alura.com.br/criando-uma-lista-com-listview-no-android/>. Acessado em: 20 nov. 2017.

FELIPE, Alex. **Personalizando uma ListView.** Disponível em: <http://blog.alura.com.br/personalizando-uma-listview-no-android/>. Acessado em: 22 nov. 2017.

JSON, **Introdução ao JSON**. Disponível em: <http://www.json.org/json-pt.html>. Acessado em: 28 set. 2017.

KANTAR WORLD PANEL, **Compartilhamento de mercado de vendas de SO de smartphones.** Disponível em: <https://www.kantarworldpanel.com/br>. Acessado em: 1 set. 2017.

METODOLOGIA ÁGIL, **Metodologia Scrum**. Disponível em: <http://metodologiaagil.com/scrum/>. Acessado em: 26 ago. 2015.

ORACLE. **Criando Serviços Web RESTful com JAX-RS e Jersey**. Disponível em: <https://docs.oracle.com/cd/E19226-01/820-7627/giepu/index.html>. Acessado em 20 set. 2017.

PARVES. **Tipos de serviços da Web SOAP, XML-RPC e Restful**. Disponível em: https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful/>. Acessado em 20 set. 2017.

PEREIRA, Ana Paula. **O que é Java?** Disponível em: < https://www.tecmundo.com.br/programacao/2710-o-que-e-java-.htm>. Acessado em: 04 dez. 2017.

PHP FLOW. **Tipos de serviços da Web SOAP, XML-RPC e Restful**. Disponível em: https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful/>. Acessado em 20 set. 2017.

REIS, Bernardo. **Conceitos Básicos Do Content Provider.** Disponível em: <http://www.decom.ufop.br/imobilis/tutorial-android-content-provider-principios/>. Acessado em 10 out. 2017.

SILVEIRA, Felipe. **Criando um Service em Android.** Disponível em: <http://www.felipesilveira.com.br/2015/03/service-em-android/>. Acessado em 10 out. 2017.

SOURCE ANDROID, **O código fonte Android**. Disponível em: <https://source.android.com/source/>. Acesso em 26 ago. 2017.

WIKIPÉDIA, **Rest.** Disponível em: <https://pt.wikipedia.org/wiki/REST>. Acessado em: 21 set. 2017.

PRIMO, Glauco. **User Stories – O que são? Como usar.** Disponível em: <http://blog.myscrumhalf.com/2011/10/user-stories-o-que-sao-como-usar/>. Acessado em 20 jan. 2018.

HIGHSMITH, Jim. **History: The Agile Manifesto.** Disponível em: <http://agilemanifesto.org/history.html>. Acessado em 31 jan. 2018.

TELES, Vinicius. **Manifesto ágil.** Disponível em: <http://www.desenvolvimentoagil.com.br/xp/manifesto_agil>. Acessado em 31 jan. 2018.

GLOBAL STATS. **Android Version Market Share.** Disponível em: <http://gs.statcounter.com/android-version-market-share/mobile-tablet/brazil>. Acessado em: 03 fev. 2018.

DEVELOPERS ANDROID. **JSONArray.** Disponível em: <https://developer.android.com/reference/org/json/JSONArray.html>. Acessado em: 05 fev. 2018.