UNIVERSIDADE DO ESTADO DA BAHIA

DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RAFAEL OLIVEIRA FACTUM

GAMIFAPP: ARQUITETURA DE FRAMEWORK PARA GAMIFICAÇÃO DE APLICATIVOS MÓVEIS

SALVADOR - BAHIA

2016

RAFAEL OLIVEIRA FACTUM

GAMIFAPP: ARQUITETURA DE FRAMEWORK PARA GAMIFICAÇÃO DE APLICATIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientadora: Lynn Rosalina Gama Alves

Co-Orientador: Eduardo Manuel de Freitas Jorge

SALVADOR - BAHIA

2016

RAFAEL OLIVEIRA FACTUM

GAMIFAPP: ARQUITETURA DE FRAMEWORK PARA GAMIFICAÇÃO DE APLICATIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Trabalho aprovado. Salvador, BA, 27 de Outubro de 2016:

BANCA EXAMINADORA

Lynn Rosalina Gama Alves (Orientadora)
Departamento de Educação
Universidade do Estado da Bahia - UNEB

Eduardo Manuel de Freitas Jorge (Co-Orientador)
Departamento de Ciências exatas e da Terra
Universidade do Estado da Bahia - UNEB

Alexandre Rafael Lenz
Departamento de Ciências Exatas e da Terra
Universidade do Estado da Bahia - UNEB

RESUMO

Com o crescente número de aplicativos móveis nas lojas do *Android* (*Google*), *Windows Store* (*Microsoft*) e *iOS* (*Apple*) gera-se um aumento considerável na concorrência entre empresas, startups e desenvolvedores autônomos. A partir desse aumento de concorrência surge a necessidade de definir um fator diferencial nos aplicativos que possa manter os usuários motivados a continuarem usando-os além de aumentar a fidelização deles. Uma discussão emergente nos últimos anos é o de gamificação, uma técnica de aumento de motivação de usuários a partir do uso de técnicas de *design de games*. Esse novo domínio de conhecimento abre novos horizontes para formas de aumentar a fidelização de usuários de aplicativos móveis e gerar novos diferenciais importantes entre aplicativos. A partir de revisão da literatura percebeu-se a necessidade de criação de artefatos reutilizáveis para implementação de gamificação em aplicativos móveis. Desta forma propõe-se neste trabalho a modelagem de um *framework* para produção de aplicativos gamificados voltados para a plataforma *Android*, criando assim uma ferramenta de reuso de código capaz de dar suporte ao desenvolvimento de aplicativos gamificados de forma mais robusta. Somado a isso, se faz também um estudo de como aplicar a gamificação no design do aplicativo. Para validar esse modelo de *framework* apresenta-se a implementação de componentes centrais do modelo e a utilização destes na refatoração de duas aplicações, *Click me!* e *Movie List*.

Palavras-chave: Gamificação. Framework. Aplicativos móveis. Android

ABSTRACT

With the crescent number of mobile applications in the app stores of Android (Google), Windows 10 (Microsoft) and iOS (Apple) it is generated a considerable raising at the competition among companies, startups and autonomous developers. From this raising of competition appears the necessity of a differential factor in the apps to make people motivated to continue using the app besides increasing the loyalty of users. A domain of knowledge that is emerging in the last years is the gamification, a technique of using characteristics of game design to increase motivation in users. This new domain of knowledge opens new doors to create differential factors in mobile applications, therefore increasing users' loyalty. Doing a literature review it is possible to notice that it is still necessary the building of reusable artifacts to implement gamification in mobile applications. This work proposes to create a framework model to develop gamified mobile applications at the Android Platform, creating a tool of code reuse capable to provide support to the development of more robust gamified apps. Additionally, a study of how to apply gamification into the design of an app was done. To validate this model it is presented the implementation of core components and the use of them in the re-factoring of two applications, Click Me! and Movie List.

Keywords: Gamification. Framework. Mobile Application. Android

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
IDE	Integrated Development Environment
LED	Linguagem Específica de Domínio
ORMLite	Lightweight Object Relational Mapping
RPG	Role-Playing Game

SUMÁRIO

1 INTRODUÇÃO

O crescimento do uso de smartphones nos últimos cinco anos impulsionou o mercado de aplicativos móveis no mundo (UPADHYAYA, 2016; RIVERA e MEULEN, 2013), devido a facilidade para desenvolvedores autônomos, pequenas empresas e *startups* lançarem suas aplicações em lojas de aplicativos. Consequentemente criou-se a necessidade de produzir métodos inovadores destinados a aumentar o engajamento e a motivação dos usuários nos aplicativos móveis, gerando um diferencial em relação a concorrência.

Uma forma de retenção e motivação dos usuários de aplicativos móveis está baseada nos conceitos de gamificação. Esses conceitos de gamificação estão em constante crescimento na atualidade em vários setores como, por exemplo, educação, motivação de empregados e campanhas governamentais (HAMARI et al., 2014). Seguindo essa tendência, os desenvolvedores começaram a inserir características de gamificação nos seus aplicativos para tentar aumentar a fidelidade e o engajamento dos seus usuários (LAW et al., 2011). Entretanto, ainda existe uma carência de ferramentas relacionadas ao reuso de software que incorporem características de gamificação nesse cenário de aplicativos móveis.

No ramo de desenvolvimento de *software* é comum a utilização de reuso de ativos do ciclo de desenvolvimento, sejam eles componentes de *software*, documentação, estruturas ou metodologias (FILHO, 2013). Dentre as diversas formas de reuso utilizadas, uma de destaque entre os desenvolvedores é a do uso de *frameworks*. Os *frameworks* trazem ao desenvolvedor a facilidade de acrescentar características de determinado domínio em seu *software* sem a necessidade de implementar todo o código do início (MARKIEWICZ; LUCENA, 2001).

Apesar da importância de Framworks na estratégia de desenvolvimento das organizações, ainda é incipiente a sua aplicação no contexto de gameficação de aplicativos móveis. Com base nesse conceito emerge a questão de pesquisa a seguir:

"Como aplicar reuso de componentes de software para facilitar a utilização das estratégias de gamificação no contexto de desenvolvimento de aplicativos móveis"

Para responder essa questão objetiva-se nesta pesquisa especificar, desenvolver e avaliar um modelo de framework para implementação de aplicativos móveis com os principais elementos de gamificação. Para tanto, propõe-se os subsequentes objetivos específicos:

1. Identificar as principais características de gamificação que geram engajamento de usuários e que podem ser aplicadas no contexto de aplicativos móveis
2. Criar um modelo conceitual de *framework* para o desenvolvimento de aplicativos móveis

gamificados

3. Implementar o núcleo principal do modelo conceitual de framework, criando uma implementação da especificação proposta para a plataforma Android
4. Avaliar o modelo de *framework* através da construção de duas aplicações gamificadas na plataforma Android

Visa-se por meio dessa ferramenta de reuso de código (*Gamifapp*) contribuir com o engajamento e consequentemente a fidelização dos usuários. Esse trabalho é dividido em 6 capítulos, o primeiro deles irá explanar sobre Gamificação, falando sobre seu conceito e a aplicação dessa área de estudo nos aplicativos móveis. No capítulo seguinte será feita uma explicação sobre reuso de software focada em *frameworks*, seus tipos e as técnicas para desenvolvimento de *frameworks* comumente utilizadas. O terceiro capítulo trata de trabalhos relacionados, mostrando outros estudos sobre frameworks na área de gamificação. O quarto capítulo trata do desenvolvimento da arquitetura, sua metodologia, e como os módulos posteriormente utilizados para validação foram implementados. Em seguida, têm-se o capítulo de validação da proposta, onde os módulos implementados são utilizados na refatoração de dois aplicativos e em segudia avalia-se a sua utilização nesses aplicativos. Finalizando o trabalho são discorridas as considerações finais e os trabalhos futuros.

2 GAMIFICAÇÃO DE APLICATIVOS MÓVEIS

O número de aplicativos móveis gamificados está em crescimento. Isso acontece devido a difusão dos conceitos de gamificação e como citado anteriormente o aumento da competitividade no setor. Para desenvolver aplicativos gamificados antes de tudo é necessário entender o conceito de gamificação e como o setor de aplicativos móveis tem utilizado desse conceito.

2.1 CONCEITO DE GAMIFICAÇÃO

O termo gamificação surgiu na indústria da mídia digital e foi cunhado no ano de 2008, popularizando-se durante o ano de 2010 com o crescimento da aplicação FourSquare (DETERDING et al., 2011). O significado do termo *gamificação* ainda é altamente debatido na indústria de games e na academia, mesmo que diversas vezes contestada (DETERDING et al., 2011) a definição comumente encontrada para esse termo é que a *gamificação* denomina o processo de usar conceitos e mecânicas de design de games para gerar engajamento em uma outra atividade ou contexto, seja essa atividade uma campanha publicitária, um aplicativo de celular ou uma dinâmica de grupo em uma sala de aula (MORA et al., 2015). Uma outra forma de definir *gamificação* de forma simplificada é que ela é a aplicação da perspectiva de um *designer de games* dentro de um determinado contexto ou problema para gerar soluções caso esse problema ou contexto fosse um game (FARDO, 2013).

Deterting et al. (2011) destaca a diferença entre brincadeira e game utilizando-se dos conceitos de *paida* (brincar) e *ludus* (jogar). A brincadeira, ou brincar, é livre de forma, expressiva, improvisada e "caótica". O game (jogo) é organizado por regras, competitivo e com um objetivo. Os autores utilizam essas definições para ilustrar num gráfico do "universo"dos games e brincadeiras em que posição se encontra a gamificação, como pode ser visto na Figura 1. Enquanto brinquedos e design lúdico estão mais voltados às características de brincadeiras os games e a gamificação se encontram no contexto de jogos. Dentro desse contexto de jogos os games são completos em termos de utilização das características de jogos enquanto a gamificação utiliza apenas parte dessas características que podem ser aplicadas fora do ambiente de um game. Dessa forma como representado na Figura 1 a gamificação está situada entre Jogo e Elementos na representação gráfica do universo dos jogos e das brincadeiras.

A gamificação gira em torno do objetivo maior de motivar os usuários, fazendo com

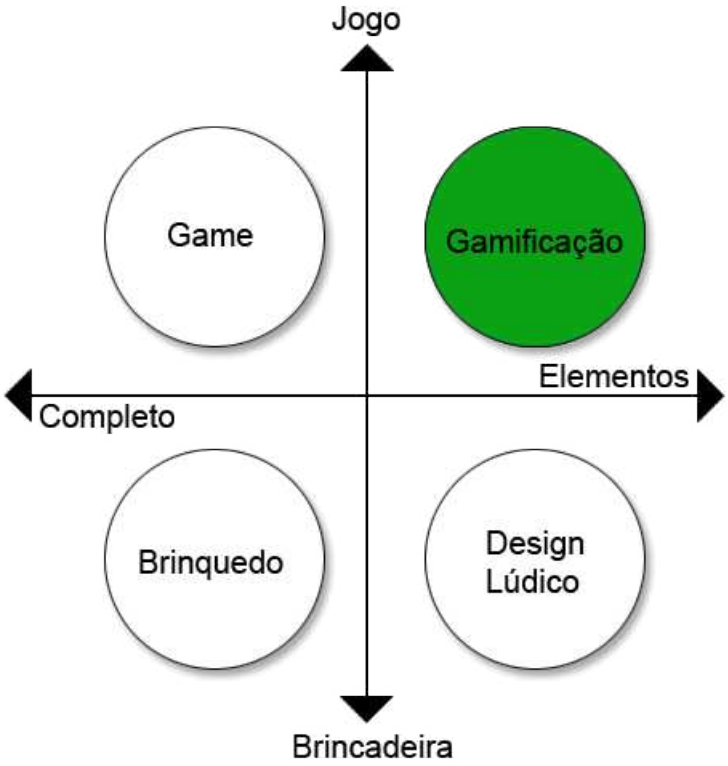

Figura 1 – Gamificação: completo x elementos e jogo x brincadeira

que eles se sintam inseridos no processo e animados a continuar participando dele. Segundo Hamari et al. (2014) a gamificação tem três partes principais: a efetividade da técnica de motivação implementada, os resultados psicológicos alcançados e o novo comportamento gerado. A gamificação busca o engajamento do usuário através de duas formas de motivação: a extrínseca e a intrínseca. A motivação intrínseca é uma motivação gerada em um indivíduo por parâmetros internos como por exemplo orgulho, força de vontade ou prazer. Dessa forma o indivíduo age ou muda seu comportamento para alcançar os objetivos motivado por esses sentimentos. A motivação extrínseca é um estímulo gerado nos indivíduos por fatores externos, como por exemplo alcançar alguma recompensa como uma premiação ou dinheiro (FARDO, 2013).

2.2 A TEORIA DO *FLOW* SOBRE A MOTIVAÇÃO

Como explicado anteriormente, o principal objetivo da gamificação é aumentar a motivação das pessoas ao realizar determinada ação. A teoria do *flow* (CSIKSZENTMIHALYI et al., 1990) estabelece as principais características psicológicas relacionadas a motivação e ao estado de bem estar ao realizar uma atividade. É importante entender como o ser humano funciona em relação a sua motivação para ser capaz de transpor essas motivações em atividades

interessantes dentro dos aplicativos móveis quando estiver criando um aplicativo gamificado.

O *flow* segundo Csikszentmihalyi et al. (1990) é um estado de consciência de total imersão e concentração em alguma atividade, e seria esse estado o responsável em tornar uma atividade de fato satisfatória. Partindo desse princípio é também estabelecido que é possível controlar esse estado de prazer ao estabelecer desafios a serem cumpridos. Esses desafios devem estar num nível razoável de dificuldade, nem tão complicado nem tão simples, dessa forma é possível se estabelecer um maior controle sobre o estado de *flow* que gera satisfação às pessoas (CSIKSZENTMIHALYI et al., 1990). Os auges de satisfação não ocorrem em situações relaxadas e sim em momentos onde a pessoa se esforça, desafiando a capacidade do corpo ou cérebro.

Também são estabelecidos oito elementos, que segundo pessoas analisadas durante o estudo realizado por Csikszentmihalyi et al. (1990), são importantes para o sentimento de prazer e satisfação que levam à motivação, são eles:

- Confrontar atividades que são possíveis de ser completadas;
- Possível de se concentrar no que está sendo feito;
- Objetivos claros;
- Feedback imediato;
- Remoção das frustações comuns do dia a dia;
- Senso de controle sobre as ações;
- Diminuição da preocupação com o *eu*;
- Senso de passagem de tempo é alterado/despercebido.

O conjunto desses elementos é responsável pelo estado de *flow*, gerando o aumento de satisfação explicado anteriormente(CSIKSZENTMIHALYI et al., 1990).

Um outro elemento importante descrito em (CSIKSZENTMIHALYI et al., 1990) é que junto ao estado de *flow* também se tem o sentimento de significado nas ações empregadas nesses desafios e controle sobre o que está ocorrendo. Esse sentimento de significado nas atividades realizadas também é um outro fator motivacional importante a ser levado em consideração.

2.3 APLICAÇÃO DOS CONCEITOS EM APLICATIVOS MÓVEIS

Quando se extrapola o conceito de gamificação para o setor de aplicativos móveis as nuances dessa área devem ser levadas em consideração para se alcançar o resultado esperado de gerar motivação, fidelização, diferencial e boa experiência de uso.

De acordo com Fong Li Law et al. (2011) as características de *game design* utilizadas no processo de gamificação de algum artefato se dividem em duas modalidades: mecânicas de games e dinâmicas de games. Nas mecânicas de games tem-se os pontos, os níveis e os desafios enquanto nas dinâmicas de games tem-se características como recompensa, conquista e status (LAW et al., 2011). Essas duas modalidades devem ser utilizadas de forma conjunta para que se possa alcançar o engajamento dos usuários.

No contexto de aplicações móveis a fidelização do usuário em relação a aplicação tem forte ligação com o seu engajamento e sua motivação em persistir utilizando aquele aplicativo no seu dia a dia. Esses fatores tornam o aplicativo mais atrativo ao usuário, fazendo com que a aplicação seja mais sustentável a longo prazo e por conseguinte uma aplicação bem sucedida para sua empresa ou desenvolvedor (LAW et al., 2011).

Durante o desenvolvimento do *Gamifapp* foram estabelecidas, a partir dos estudos realizados anteriormente sobre gamificação, quais características seriam importantes para uma ferramenta de reuso de código voltada aos aplicativos móveis e quais delas eram comumente utilizadas por aplicações com alto nível de aceitação do público, que foi medido utilizando as ferramentas de avaliação das próprias lojas onde eles são baixados. Dessa forma foi realizada uma análise de três aplicativos gamificados com alto índice de aceitação dos usuários. Os aplicativos selecionados foram *Duolingo*, um aplicativo de aprendizado de idiomas, *Fitocracy*, um aplicativo que permite a criação de um calendário de atividades físicas e o *Habitica* um aplicativo que permite a criação de atividades e rotinas para ajudar na gerência das atividades diárias. Para definir esse índice foram medidas a avaliação nas lojas de aplicativos Google Play e App Store e o número de downloads. A Tabela 1 mostra quais os elementos de games esses aplicativos utilizaram para gerar motivação nos usuários.

Tabela 1 – Aplicativos estudados e o uso dos elementos de Games selecionados

App | Características	**Duolingo**	**Fitocracy**	**Habitica**
Troféus e Conquistas		x	
Missões	x	x	x
Pontuação	x	x	x
Níveis	x	x	x
Conexão social	x	x	x
Penalidade	x		x
Dinheiro Virtual	x		x

Mesmo tendo características semelhantes, cada aplicativo implementa as funcionalidades da sua própria maneira para que ela fique adaptada ao contexto ao qual o aplicativo propõe

inserir o usuário.

Troféus e Conquistas citados anteriormente são prêmios atribuídos ao usuário por fazer determinada ação no aplicativo. Dentre os aplicativos analisados, apenas o Fitocracy (Figura 2) utiliza-se desses elementos. No contexto do aplicativo os troféus e conquistas são usados como um método de premiação para os exercícios feitos pelo usuário e registrados no aplicativo.

Figura 2 – Troféus e Conquistas no *Fitocracy*

As **Missões** são metas a serem alcançadas para que o usuário avance, mude de nível e se motive. Elas estão presentes nos 3 aplicativos analisados (Figura 3). No *Duolingo* as missões são as atividades que devem ser feitas pelo usuário para aprimorar o idioma selecionado para aprendizado. Já no *Habitica*, as missões são rotinas criadas pelo usuário ou pelos grupos os quais esse usuário participa para assim ajudar a gerenciar as atividades dele. Além desse método, o *Habitica* ainda tem uma missão maior em grupo para reforçar os desafios da atividade. Essa missão contabiliza todas as atividades criadas por um grupo e quais pessoas a fizeram para completar o desafio de derrotar um chefe (inimigo virtual utilizado no aplicativo para simular um jogo de Role-Playing Game (RPG) junto com outras características que ele utiliza). No *Fitocracy* as missões são os conjuntos de exercícios físicos cadastrados pelo usuário no calendário de atividades que ele tem para fazer durante um determinado tempo.

Pontuação é uma forma de contabilizar o desempenho do usuário ao fazer as missões

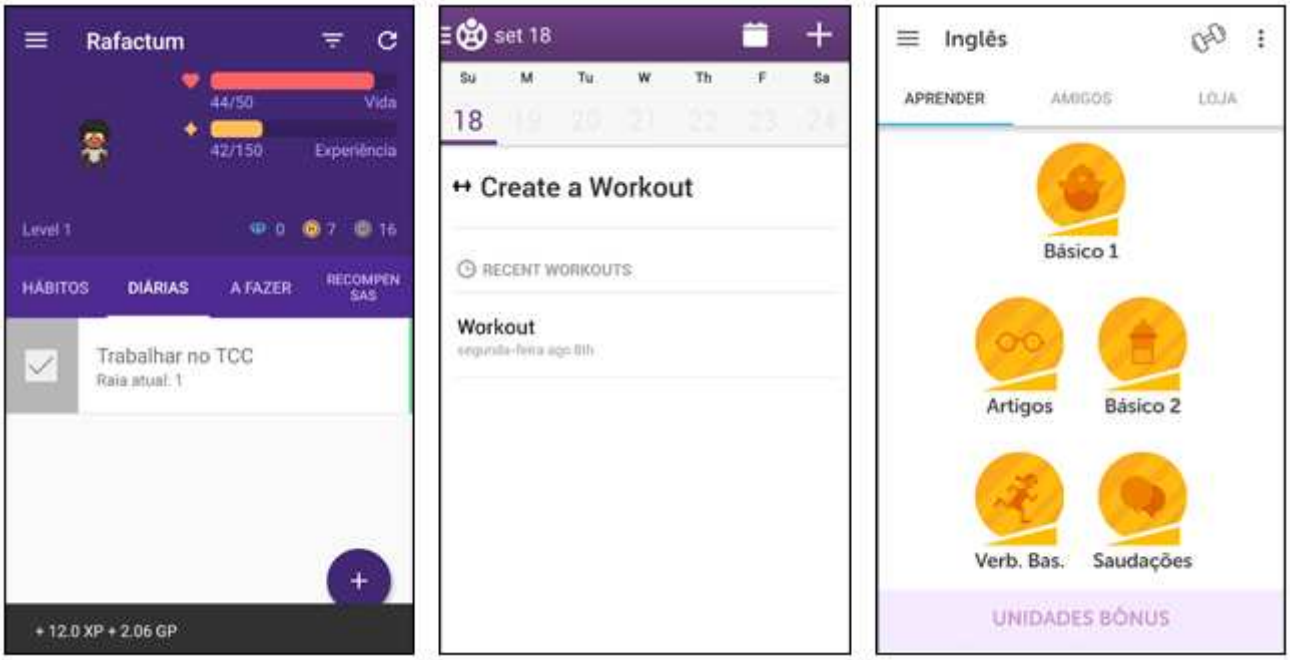

Figura 3 – Missões nos aplictivos *Habitica*, *Fitocracy* e *Duolingo* respectivamente

naquele aplicativo. Os três aplicativos estudados utilizam pontuação (Figura 4). As pontuações em todos os aplicativos são recebidas após realizar as missões estabelecidas e servem como um medidor de experiência adquirida tanto para passar de nível quanto para receber conquistas e dinheiro virtual.

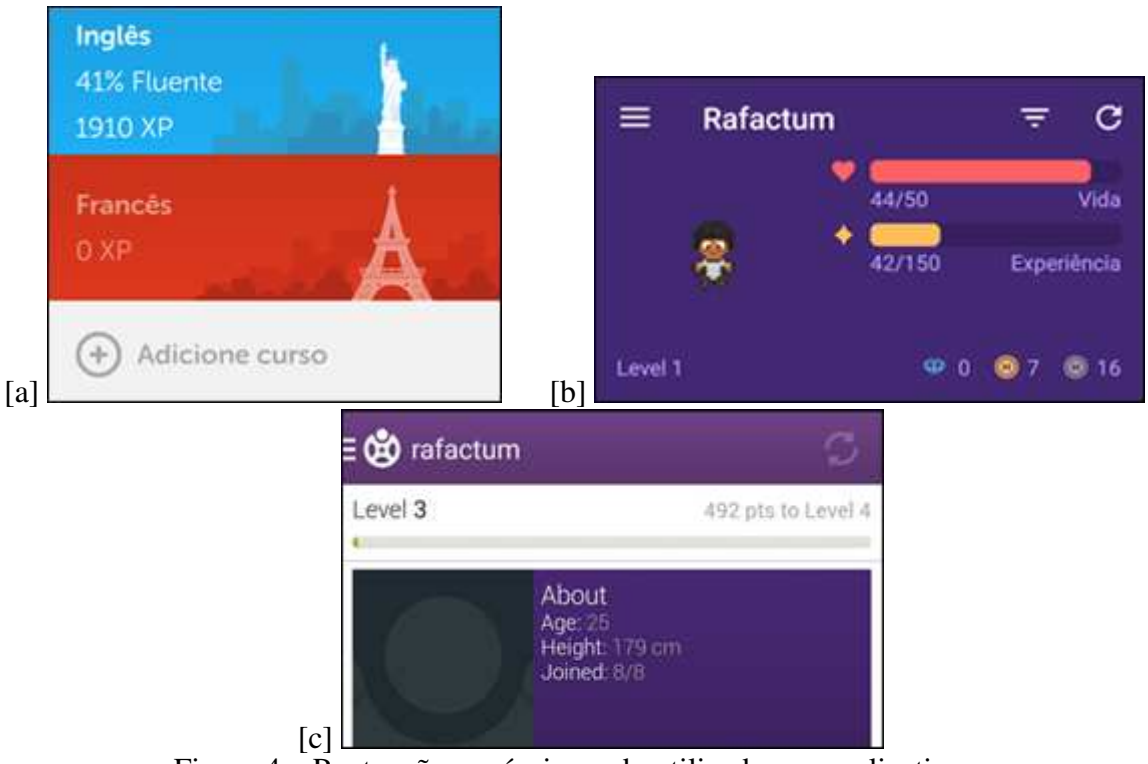

Figura 4 – Pontuações e níveis sendo utilizados nos aplicativos

Os **Níveis** são uma medida de controle de desenvolvimento, uma forma de dar *feed-back* aos usuários em relação ao desempenho deles nas suas atividades (Figura 4). Normalmente o nível vai aumentando à medida que o usuário vai realizando as missões e recebendo pontos. Nos três aplicativos analisados os níveis funcionam da forma descrita anteriormente.

Conexão Social é a forma com que o aplicativo faz com que as pessoas que o utilizam se sintam conectadas a outras pessoas que também fazem uso da aplicação. Essa conexão pode se dar por meio de amizades, rankings ou seguidores. Essa também é uma característica utilizada pelos três aplicativos analisados (Figura 5). O duolingo permite que o usuário siga outros a fim de ver como anda o desenvolvimento deles no aprendizado de algum idioma. Já o *Fitocracy* permite o usuário adicionar outros como amigos para ver quais atividades físicas esses amigos estão realizando atualmente. Também é possível postar *status* para que os amigos vejam no mural do usuário. Enquanto isso o *Habitica* permite a adição de amigos e a criação de grupos de atividades chamados no aplicativo de *guildas*.

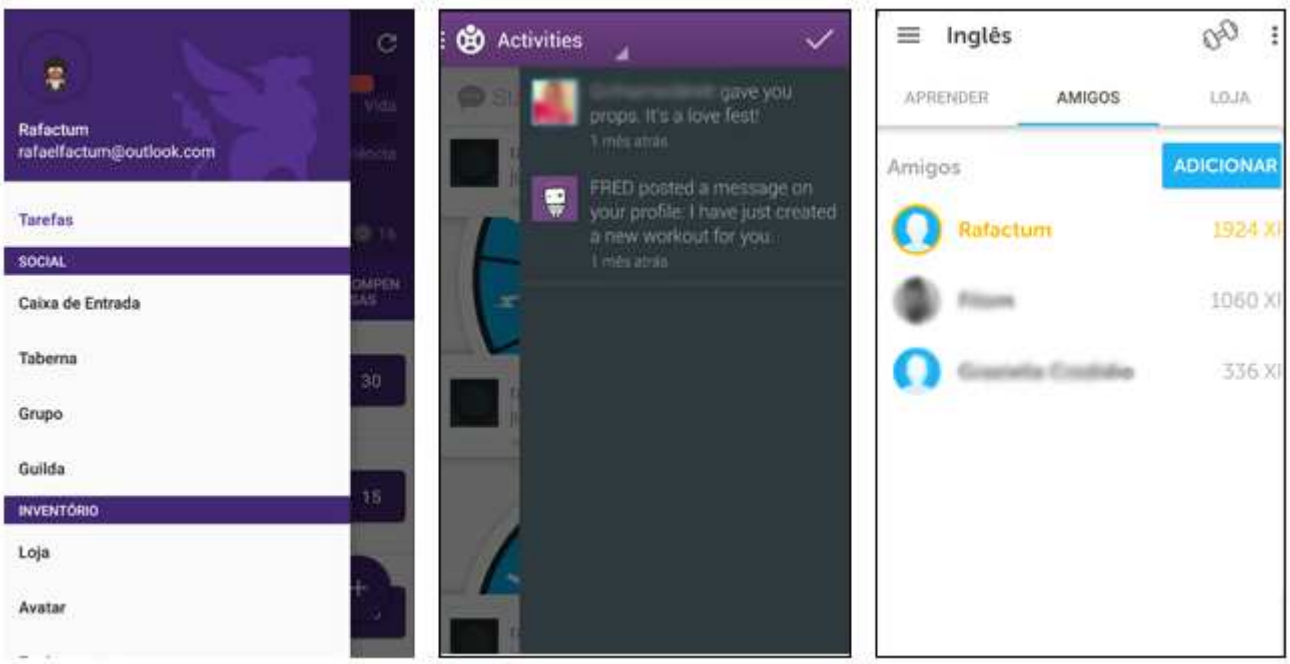

Figura 5 – Formas de interação social nos aplictivos Habitica, Fitocracy e Duolingo respectivamente

Penalidades são formas de punir os usuários por não realizarem ou realizarem de forma incorreta uma missão atribuída a eles. O *Duolingo* estabelece um número de *vidas* para realizar as missões. Ao perder em alguma atividade (missão) o usuário perde uma dessas vidas e ao zerar as vidas o usuário deve esperar um tempo para poder voltar a realizar atividades. O *Habitica* por outro lado, utiliza as penalidades como forma de lembrete ao usuário sobre alguma

tarefa à qual ele deveria ter realizado num determinado momento e não o fez. O *Habitica* no caso de atividades não realizadas retira *pontos de vida* do *avatar*(representação virtual) do usuário.

O **Dinheiro virtual** é uma forma de dinheiro recebida nos aplicativos após a realização de missões ou ao receber alguma premiação (troféu ou conquista). Normalmente o dinheiro virtual é utilizado para comprar novas oportunidades de realizar uma tarefa, para comprar atividades extras ou para adquirir itens para o *avatar* do usuário. O *Duolingo* utiliza as duas primeiras formas citadas anteriormente e o *Habitica* utiliza as duas últimas (Figura 6).

Figura 6 – Dinheiro virtual e seu uso nos aplictivos *Habitica* e *Duolingo* respectivamente

Os conceitos discutidos nesse capítulo serão utilizados posteriormente para junto as análises de frameworks de gamificação já existentes, traçar as características a serem utilizadas no Gamifapp.

3 FRAMEWORKS NO DESENVOLVIMENTO DE APLICATIVOS

O reuso de software é uma técnica de desenvolvimento de software que visa aumentar a produtividade, confiabilidade e diminuir os custos de desenvolvimento de um sistema. Segundo Johannes Sametinger citado em Filho (2013, p. 9) "O reuso de software é o processo de criar sistemas de software através de softwares existentes ao invés de criá-los do princípio". as *partes de softwares existentes* especificadas na definição de reuso podem ser qualquer tipo de artefato de software, desde documentação até funções e partes do código. Partindo do crescimento e importância do reuso de software no desenvolvimento de sistemas surgiram os *frameworks*, uma ferramenta de auxílio ao desenvolvimento de sistemas utilizando reuso. Neste capítulo os conceitos, classificações e técnicas de desenvolvimentos de *frameworks* serão explanados.

3.1 FRAMEWORKS E SUAS CLASSIFICAÇÕES

Uma forma de criar uma estrutura base para a implementação de aplicações gamificadas a fim de alcançar sustentabilidade e engajamento é por meio de um framework. Um framework é um gerador de aplicações de um domínio específico baseado em reuso de código e características similares (MARKIEWICZ; LUCENA, 2001). Outras definições para framework encontradas são: (1) "framework é um conjunto de classes incorporadas a partir de um design abstrato capazes de alcançar soluções de problemas de uma determinada família"e (2) "framework é um design reutilizável de todo ou parte de um sistema que é representado por um grupo de classes abstratas e como elas interagem"(STANOJEVIĆ et al., 2011).

A conexão entre o *software* em desenvolvimento e o *framework* acontece de duas formas, por *hotspot* (Figura 7) ou por *frozenspot*. Os *frozenspots* são partes do *framework* previamente desenvolvidas e fixas que por meio de chamadas (normalmente indiretas) aos *hotspots* realizam as suas funções (LIMA, 2015). Segundo Markiewicz e Lucena (2001), frameworks têm módulos que não podem ser modificados pelos desenvolvedores que os estão utilizando, esses módulos imutáveis são os *frozenspots*. Eles são o *kernel* do framework e são classes ou métodos que já vêm implementados com o objetivo de serem utilizados pelo desenvolvedor. Enquanto os *frozenspots* são fixos, os *hotspots* são as partes flexíveis do *framework*, eles são os componentes do framework que precisam de um complemento implementado pelos desenvolvedores *software* para realizar sua função (LARMAN, 2007). De forma simplificada, os *hotspots* são um conjunto de classes abstratas ou métodos aos quais os desenvolvedores têm acesso para implementar o

código específico do seu software (MARKIEWICZ; LUCENA, 2001).

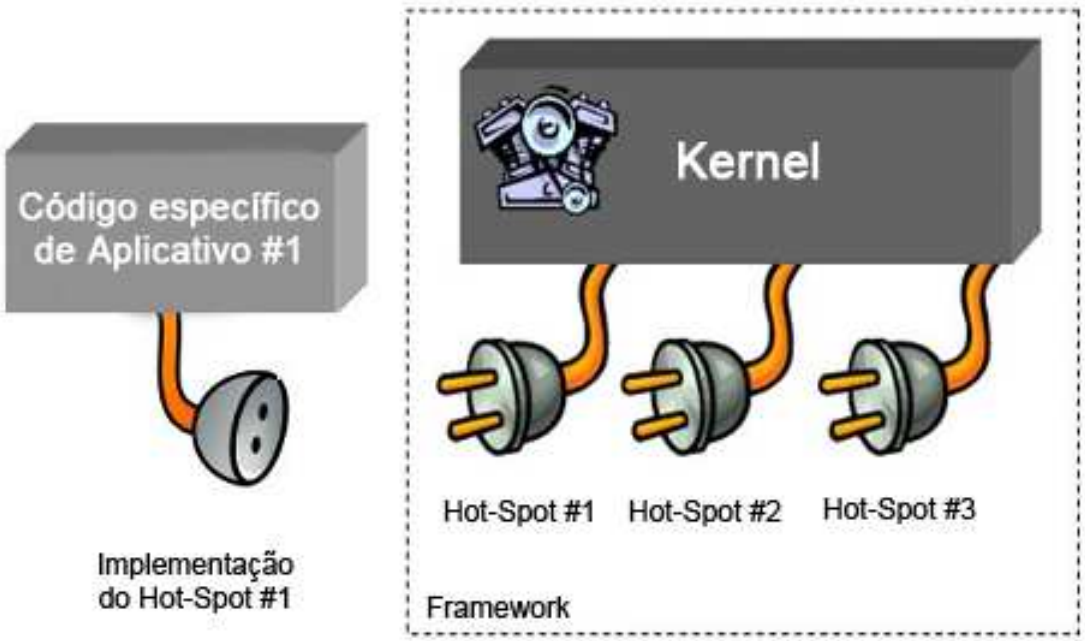

Figura 7 – Ilustração do funcionamento dos *Hot-Spots* de um *framework*

Leandro, citado em Lima (2015), especifica que existem dois tipos de grupo de classificação para *frameworks*. O primeiro grupo define a forma a qual o *framework* é utilizado. As classificações que compõem esse grupo são *framework* caixa-preta, *framework* caixa-branca e *framework* caixa-cinza. O *framework* caixa-branca é flexível que permite personalização por meio de classes abstratas e herança, é um tipo de *framework* normalmente complicado para iniciantes pela necessidade de entender como o código do *framework* funciona internamente. O tipo caixa-preta, diferente do caixa-branca, é mais simples de ser utilizado por iniciantes pois só é necessário entender como funciona a interface de conexão do *framework* para utilização de suas funções, além disso, o *framework* caixa-preta não permite personalização. O *framework* caixa-cinza tenta utilizar-se dos benefícios que ambos os tipos anteriormente explicados trazem. Ele faz uso tanto das classes abstratas e suas possibilidades de personalização por parte do programador quanto do acesso facilitado por meio de interfaces que permite o uso mais simples por parte de iniciantes (GONÇALVES, 2013; LIMA, 2015).

O segundo grupo de classificações especificado em Lima 2015 trata da área de aplicação ao qual o *framework* pode ser utilizado. A primeira classificação é a de *framework* horizontal, onde este pode ser aplicado em diversos tipos de aplicações, mas devido a essa flexibilidade de uso esse tipo de *framework* tende a resolver apenas pequenas partes do problema que são comuns a essas aplicações. A segunda classificação é o *framework* vertical, esse tipo de

framework é focado em resolver problemas de uma área específica de aplicações, dessa forma ele consegue resolver uma quantidade maior de problemas devido à grande semelhança entre os problemas das aplicações as quais o *framework* será utilizado.

3.2 TÉCNICAS DE DESENVOLVIMENTO PARA CONSTRUÇÃO DE FRAMEWORKS

Para garantir a flexibilidade de um *framework* é comum a utilização de técnicas de desenvolvimento específicas durante a sua elaboração. Além dos padrões de projeto já difundidos entre os desenvolvedores, como por exemplo *Data Access Object*, *Value Object* e *Template Method*, algumas específicas também são utilizadas. Algumas dessas técnicas são *Anotações*, *Reflexão* e *Linguagem específica de domínio*. Essas técnicas serão explicadas nas subseções a seguir.

3.2.1 Anotações

Em Araújo (2012) é especificado que uma anotação no contexto de orientação a objeto é um metadado. Um metadado é uma informação sobre um elemento do código. Metadados são comumente usados para documentação de código (LIMA, 2015), como por exemplo no *JavaDoc* - método mais comum de documentação de códigos em Java, ou para auxílio na codificação em tempo de compilação, como por exemplo na anotação @Override da linguagem Java usada para garantir sobrescrita de método (ARAÚJO, 2012). A anotação segundo Alves citado em Lima (2015) é um recurso de marcação de componentes de código (métodos, variáveis e classes) para tratamento por meio de compiladores ou um Integrated Development Environment (IDE) - Ambiente Integrado de Desenvolvimento. Guerra citado em Araújo (2012) explica que as anotações também podem ser acessadas por outras classes e métodos para que esses possam se adaptar a presença da anotação e reproduzir um comportamento diferente.

Existem três categorias de marcação, são elas (ARAÚJO, 2012):

1. Anotação marcadora - são as anotações que possuem apenas seu nome, sem nenhum tipo de conteúdo adicional. Um exemplo dessa marcação é o @Override;
2. Anotação de valor único - são anotações que possuem um campo chamado valor. Elas são representadas pelo nome da anotação e o nome valor entre parênteses como no exemplo @Anotacao("valor").
3. Anotações complexas - são anotações que têm mais de um valor e esses valores devem ser

representados por pares (como nome=valor) separados por vírgula. A sintaxe para esse tipo de anotação é @Anotacao(nome=valor, nome2=valor2).

Segundo Guerra citado em Araújo (2015) as Anotações são métodos de inserir metadados direto na aplicação de modo a manter o código e os metadados em um mesmo lugar, facilitando assim a administração.

3.2.2 Reflexão

A reflexão é um processo que permite à aplicação, em tempo de execução, observar e modificar a sua estrutura de forma a adaptar o seu comportamento (LIMA, 2015). A técnica de reflexão permite ao programa obter informações sobre ele mesmo e a partir dessas informações manipular seus métodos e atributos. O uso de reflexão tem como principal objetivo aumentar a flexibilidade do código permitindo maior reutilização. Apesar de a reflexão trazer diversos benefícios como diminuição dos erros, padronização do código e aumento de produtividade ela também traz consigo riscos como diminuição do desempenho e exposição da estrutura interna dos objetos (LIMA, 2015).

3.2.3 Linguagem específica de domínio

A LED - Linguagem específica de domínio é uma linguagem de programação utilizada para resolver problemas de um domínio específico. É um tipo de linguagem de expressividade limitada (GONÇALVES, 2013). Essas linguagens têm quatro elementos-chave, são eles:

• Linguagem de programação: a LED é utilizada pra dar instruções a um computador para que ele realize alguma ação. Ela é feita para que seja de fácil entendimento por humanos e ainda assim seja passível de execução pelo computador.

• Natureza da linguagem: a LED deve ter uma natureza tanto para expressões individuais quanto nas composições de expressão.

• Expressividade limitada: uma LED deve suportar o mínimo de técnicas para atender ao seu domínio, dessa forma ela só pode ser utilizada para desenvolver parte de um software e não um sistema completo. Isso ocorre devido a complexidade gerada por uma linguagem para atender à todas as necessidades de desenvolvimento de um sistema completo.

• Foco no domínio: a LED é uma linguagem de domínio limitado e foco específico, o que permite a ela uma maior utilidade quando aplicada dentro desse domínio específico. Isso

acontece devido à sua expressividade limitada.

Vantagens importantes do uso de LEDs são a fácil manutenção do código, a clareza de comunicação, entendimento e o aumento de produtividade (GONÇALVES, 2013).

As LEDs são divididas em dois tipos, as externas e as internas. As externas são feitas numa linguagem diferente da linguagem da aplicação, normalmente são scripts utilizados para leitura interna da aplicação, como um documento *XML*. As LEDs internas são desenvolvidas utilizando a própria linguagem da aplicação, o que traz uma vantagem ao facilitar o entendimento do funcionamento da LED. Um outro nome atribuído às LEDs internas é *interface fluente* (GONÇALVES, 2013).

As técnicas de desenvolvimento de framework descritas no capítulo são importantes para robustez, confiabilidade e fácil utilização do framework por parte dos desenvolvedores, as técnicas descritas nesse capítulo serão tratas novamente nos capítulos de desenvolvimento do Gamifapp demonstrando quais e como foram utilizadas.

4 FRAMEWORK PARA GAMIFICAÇÃO DE APLICATIVOS MÓVEIS

Notando-se o crescimento da gamificação no âmbito de tecnologia, principalmente em aplicativos móveis, emerge a necessidade de ferramentas de reuso de software capazes de acelerar o desenvolvimento e a qualidade desses aplicativos que utilizam gamificação como motivador e diferencial para os seus usuários. Como descrito anteriormente o *framework* é uma técnica de reuso bem propagada e estabelecida entre desenvolvedores de software. Dessa forma um *framework* para desenvolvimento de aplicativos móveis gamificados pode ser uma ferramenta muito construtiva e importante para a indústria. Existem diversos *frameworks* de gamificação em áreas como educação, negócios e internet além da diversidade de frameworks genéricos existentes (MORA et al., 2015), entretanto ainda existe incipiência quando o assunto é frameworks de gamificação focados na área de aplicativos móveis.

4.1 FRAMEWORKS DE GAMIFICAÇÃO

MORA et al. (2015) estabelece por meio de uma pesquisa direcionada aos *frameworks* de gamificação que entre os anos de 2011 e 2015 existiam por volta de vinte e dois *frameworks* de gamificação, sendo a maior parte deles genéricos. O número de *frameworks* por ano pode ser encontrado na Figura 8.

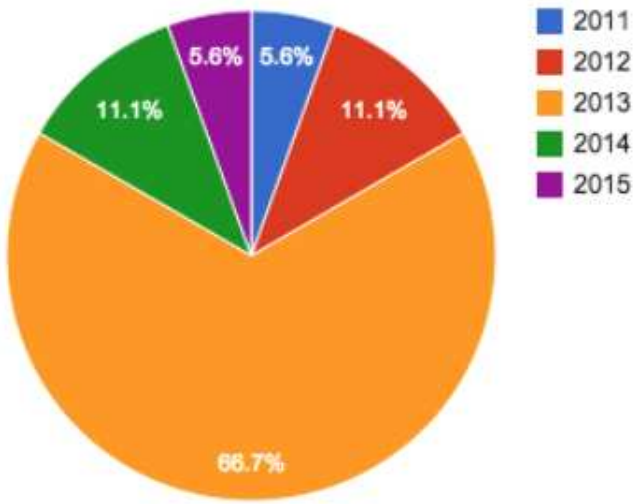

Figura 8 – Número de frameworks lançados entre 2011 e 2015 de acordo com Mora et al. (2015)

Dentre os vinte e dois encontrados na pesquisa feita em Mora et al. (2015) foram selecionados dezoito os quais foram separados em diferentes categorias. Essas categorias foram

background, escopo e abordagem. *Background* refere-se a se o framework seria acadêmico ou não, *escopo* trata da divisão entre gamificação completa do processo ou apenas focada numa parte e no que tange *abordagem* tem-se a divisão entre os *frameworks* genéricos e os específicos. O número de *frameworks* em relação às classificações pode ser encontrado na Figura 9.

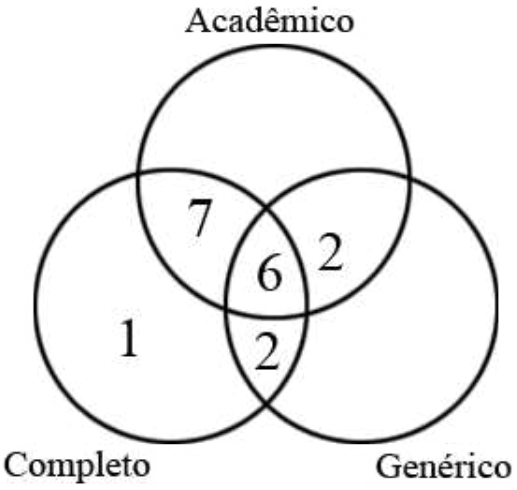

Figura 9 – Classificação dos dezoito *frameworks* descritas em Mora et al. (2015)

Um importante dado notado por Mora et al. (2015) é que a maioria dos *frameworks* descritos no estudo eram baseados em design focado em humano ao invés de design focado em funções. O design centrado em humano é um princípio onde o objetivo é a motivação da pessoa e não a eficiência por si só. Levando isso em conta nota-se que as necessidades sociais para a motivação das pessoas é o maior foco em grande parte dos *frameworks* de gamificação existentes até o momento. Zichersmann citado em Mora et al.(2015) teoriza que gamificação é 75% psicologia e 25% tecnologia. Mora et al. (2015) estabelecem uma lista de dezenove características de design de jogos que são importantes na gamificação e as divide em 5 diferentes categorias, são elas:

1. Economia
   - Objetivos: os objetivos de performance a serem alcançados.
   - Viabilidade: análise de eficácia da gamificação no projeto proposto.
   - Risco: a chance de ocorrer algum tipo de prejuízo.
   - Retorno de investimento: retorno positivo gerado pelo uso da gamificação no projeto.
   - Partes interessadas: pessoas que interagem com o processo de desenvolvimento.

2. Lógica
   - Laço: as mecânicas de jogo e a forma de *feedback* para gerar uma experiência motivadora no usuário.
   - Fim de jogo ou Vitória épica: uma experiência pré-estabelecida que requer extensa habilidade do usuário, tem como finalidade aumentar o nível de motivação por meio de uma superação marcante.
   - Integração: o método utilizado para atrair novos usuários
   - Regras: as normas estabelecidas para o contexto o qual o projeto gamificado estará inserido.

3. Medição
   - Métricas: as medidas padrão para mensurar performance, eficiência, qualidade, etc.
   - Analítico: algoritmos para analisar indicadores-chave de performance.

4. Psicologia
   - Diversão: o aproveitamento e apreciação dos usuários
   - Motivação: o que gera a vontade da pessoa de continuar participando de algo.
   - Social: a interação entre os usuários.
   - Comportamentos desejados: o comportamento esperado que os usuários tenham ao interagirem com o projeto.
   - Ética: a separação dos conceitos de certo e errado de acordo com a psicologia.

5. Interação
   - Narrativa: o contexto criado pelos *designers* no qual os usuários estarão inseridos.
   - Interface de usuário e experiência de usuário: a forma de interação entre usuários e projeto, além de muitas vezes representar graficamente características do usuário, como por exemplo um avatar.
   - Tecnologia: o uso de *software* ou componente tecnológico no projeto gamificado.

Dentre os dezenove itens descritos anteriormente Mora et al. (2015) selecionam dez deles para uma análise dos *frameworks* estudados. Esses dez são viabilidade, partes interessadas, laço, fim de jogo, integração, regras, métricas, ética, design de interface e tecnologia. Nessa

avaliação percebeu-se que a característica menos utilizada nos *frameworks* foi ética e as mais utilizadas foram regras, métricas e laços, respectivamente.

4.2 O FRAMEWORK OCTALYSIS

O *framework Octalysis* é um *framework* genérico criado por Yu-Kai Cho depois de dez anos de estudo sobre gamificação (CHOU, 2015). O *framework Octalysis* foi criado baseado em oito características principais, são elas: significado, realização e conquista, propriedade, influência social, escassez, imprevisibilidade e anulação. Essas oito características são utilizadas pelo *framework* para gerar um octógono (Figura 10) que é usado para medir o grau de desenvolvimento de algo gamificado.

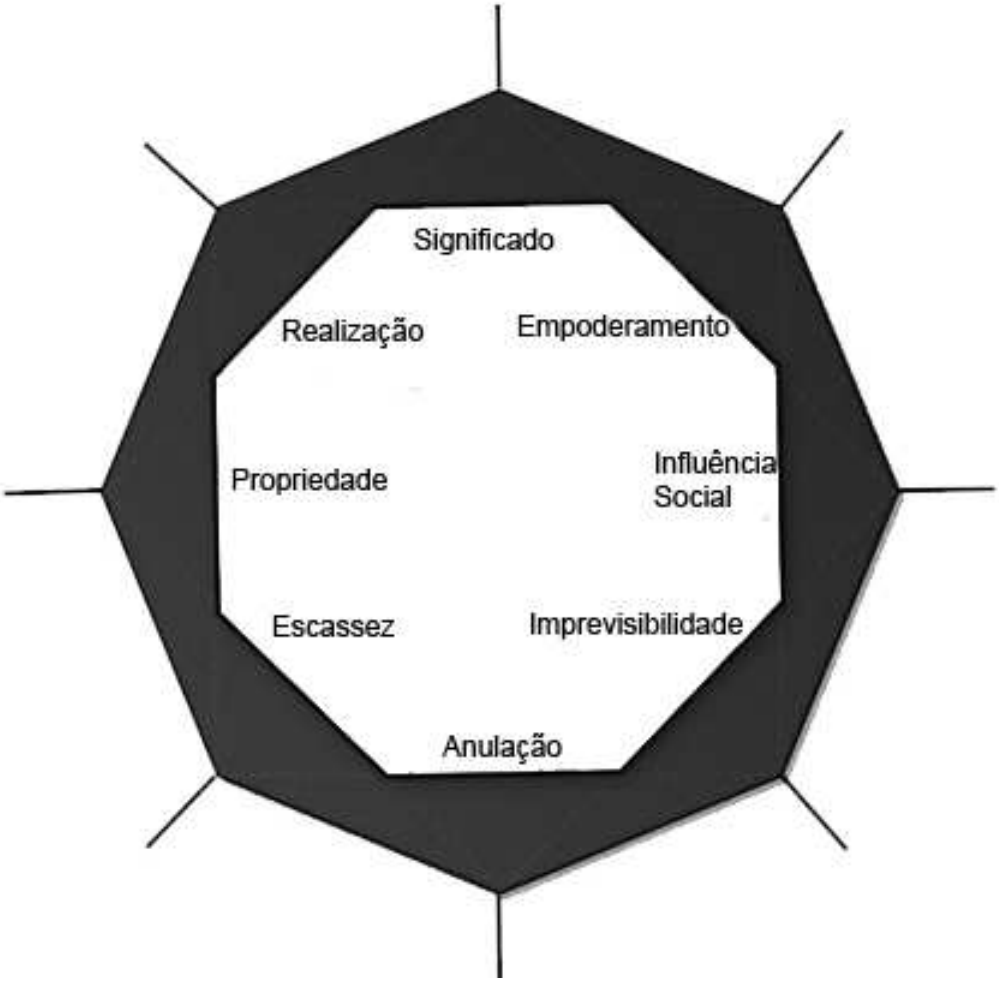

Figura 10 – Octógono criado com as 8 características do Octalysis (CHOU, 2015)

Significado segundo Cho (2015) é o sentimento de que a pessoa está participando de algo que vai além daquela simples atividade e que é maior do que ela, como por exemplo

quando alguém se dedica a ajudar no desenvolvimento de algum fórum ou na criação de conteúdos públicos como no *Wikipédia*. Isso também acontece quando a pessoa se sente única em determinado contexto, como por exemplo quando alguém consegue algum item raro no começo de um jogo, fazendo-o se sentir como alguém ¨escolhido¨ e diferente dos demais.

Realização no Octalysis se refere à atividade de realizar desafios e progredir em determinado contexto, melhorando as habilidades e alcançando objetivos. É normalmente onde a maioria das abordagens de gamificação se foca.

Empoderamento é a participação ativa e criativa do usuário dentro do contexto gamificado, onde ele deve pensar e repensar diversas vezes a forma com a qual ele vai interagir com aquele contexto para alcançar os seus objetivos. Isso faz com que o usuário tenha formas de expressar sua criatividade, uma necessidade comum humana.

Propriedade é descrito como uma forma de fazer o usuário se sentir como dono de algo. Isso aumenta a motivação do usuário, pois quando ele se sente dono de algo isso faz com que ele queira o melhor para aquilo e queira também desenvolvê-lo. Normalmente essa técnica é aplicada por meio de objetivos ou avatares virtuais.

Influência Social é uma característica do *Octalysis* que incorpora todas as relações sociais possíveis de serem aplicadas em gamificação, como por exemplo companheirismo, responsabilidades sociais, mentoria, competição e aceitação. É utilizada, por exemplo, para que dentro de um grupo de pessoas o usuário se sinta motivado a melhorar suas habilidades para se equiparar aos seus amigos.

Escassez é fazer com que o usuário queira obter algo que ele não pode naquele exato momento. É uma característica muito utilizada em games de celular que pedem para que o jogador volte em algumas horas ou minutos para obter algo. Também foi utilizada por grandes empresas como o *Google* com o *Gmail* (provedor de e-mail) ou o Facebook com sua rede social, onde para participar as pessoas precisavam ser convidadas. Então quando essas empresas abriram o serviço para todos um grande número de usuários passou a utilizá-los. É uma característica que deve ser utilizada cuidadosamente para não diminuir a motivação do usuário, tendo um efeito contrário ao desejado.

Imprevisibilidade é o ato de gerar curiosidade nos usuários, fazendo-os esperar para descobrir o que vai acontecer em seguida. Dessa forma o usuário sempre fica pensando sobre o projeto gamificado. Essa característica é uma das maiores responsáveis pela criação dos vícios.

Anulação/Perda é a utilização da perda como motivador. Por meio da anulação a gamificação faz com que o usuário continue a participar daquele projeto (aplicativo, jogo, trabalho ou campanha, por exemplo) para que ele não perca algo que se importa. O processo de evitar perda também inclui tudo o que o usuário adquiriu até aquele momento participando do projeto. Nesse caso, ao deixar de participar, tudo o que foi obtido anteriormente deixaria de ter valor.

5 GAMIFAPP

5.1 METODOLOGIA DE DESENVOLVIMENTO

O desenvolvimento do modelo proposto foi dividido em quatro etapas: especificação dos elementos de gamificação, definição do tipo de *framework*, design do modelo conceitual e por fim a implementação de componentes do modelo conceitual do *framework* para avaliação na plataforma *Android*.

Na primeira etapa foi realizado um estudo das principais características de gamificação utilizadas por empresas de aplicativos móveis no mercado. A avaliação da relevância desses aplicativos perante o mercado é feita por meio das qualificações recebidas e o número de *downloads* nas lojas de aplicativos móveis das plataformas *iOS* e *Android*. A partir da análise dessas aplicações, suas características de gamificação e sua relação com o aumento de engajamento, são selecionadas as principais mecânicas e dinâmicas de games a serem utilizadas no modelo.

Na segunda etapa do desenvolvimento foi feito um estudo da utilização de *frameworks* em aplicações móveis para identificar qual o tipo de *framework* tem sido mais utilizado nesse ambiente de desenvolvimento. Dessa forma foi possível definir qual método de desenvolvimento seria utilizado no projeto.

Na terceira etapa foi desenvolvido o modelo conceitual do *framework* utilizando-se dos conhecimentos adquiridos nas duas etapas anteriores. Para dar apoio ao desenvolvimento desse modelo foram feitos estudos de similares e outros tipos de *framework* de gamificação já existentes.

Na quarta e última etapa de desenvolvimento foram implementados componentes do modelo conceitual do *framework* na plataforma *Android* para a avaliação de viabilidade dele. Esses módulos foram utilizados posteriormente para refatorar dois aplicativos gamificados simples de forma a provar a funcionalidade do framework e demonstrar a simplificação do desenvolvimento com a sua utilização.

A metodologia utilizada no projeto tem como base a metodologia de desenvolvimento de *framework* estabelecida em Joon (1998). Essa metodologia se divide em quatro etapas, são elas:

• Análise: Consiste na especificação do domínio o qual o *framework* será aplicado. A especificação é feita a partir da análise de aplicações similares para que seja feita a extração de características comuns entre elas.

• Design: Nessa fase é feita a análise do tipo de *framework* que será desenvolvido (caixa-preta, caixa-branca ou caixa-cinza) e quais são os *hotspots* desse *framework*.

• Implementação: A implementação do *framework* consiste em iterações contínuas de desenvolvimento e aprimoramento.

• Testes: Nessa etapa o que foi desenvolvido na etapa anterior é testado para que o ciclo de desenvolvimento do *framework* possa continuar.

Para esse projeto algumas etapas da metodologia de Joon (1998) foram adaptadas para atenderem mais adequadamente às necessidades, deixando-as como o esquema ilustrado na Figura 11. Dessa forma, a etapa de *Design* foi dividida em duas partes: Especificação de *Framework*, onde é feito o estudo da melhor abordagem de desenvolvimento para o *framework* proposto, e *Design do Modelo*, onde é feito o desenvolvimento do modelo de *framework* proposto no trabalho. Além disso, foi inserida uma nova etapa no ciclo de construção do *framework*. Essa etapa ocorre após alguns ciclos de implementação e testes onde o modelo de *framework* proposto é reavaliado e modificado caso seja necessário.

Figura 11 – Diagrama da metodologia de desenvolvimento do projeto. Fonte: Elaborada pelo autor.

A metodologia foi utilizada no projeto da seguinte forma:

• Análise: Nessa etapa foram desenvolvidos dois aplicativos, o *Click me!* e o *Movie List* que serão melhor explicados nas seções seguintes. A partir dessas duas aplicações foi possível extrair como funciona o uso de pontuações e troféus em diferentes contextos de aplicativos móveis, facilitando assim a extração das semelhanças para a implementação desses componentes do *framework*, como será demonstrado na seção de desenvolvimento dos módulos para avaliação do *framework*;

• Especificação do *Framework*: Na etapa de especificação foram analisados os dados extraídos dos estudos sobre os aplicativos descritos anteriormente (*Duolingo, Habitica e Fitocracy*) e o estudo sobre o *framework Octalysis* (CHOU, 2015) e suas especificações das características mais importantes da gamificação. Após essa análise foram selecionados quais os módulos seriam utilizados para a construção do *framework* proposto nesse trabalho;

• Design do Modelo: Nessa etapa foram feitas duas versões do *framework*. A primeira versão foi do tipo caixa-branca onde os desenvolvedores teriam acesso mais direto a sobrescrever os métodos dos módulos do *framework*. Entretanto, após a primeira avaliação foi decidido refazer a forma de acesso que os desenvolvedores teriam ao *framework*, gerando assim uma segunda versão desse. Nessa segunda versão, o *framework* se tornou caixa-cinza, tornando o acesso do programador a ele mais direto por meio de uma classe chamada *Gamifapp*, utilizando o padrão de projeto *facade*. Essa modificação será mais detalhada na seção de desenvolvimento dos módulos do *framework*;

• Implementação: Na etapa de implementação foram selecionados os padrões de projeto que seriam utilizados e em seguida foi feita a implementação dos módulos do *framework*;

• Teste: Os testes foram realizados utilizando a refatoração dos aplicativos citados anteriormente, *Click me!* e *Movie List*.

As seções a seguir detalham de forma mais clara as etapas da metodologia no desenvolvimento do *Gamifapp*.

5.2 ARQUITETURA CONCEITUAL DO *FRAMEWORK*

Seguindo a metologia detalhada anteriormente, foi feita uma arquitetura de alto nível para o *framework* como pode ser visto na Figura 12. Essa arquitetura é dividida em três camadas, são elas:

• App: esta camada contém a aplicação que fará uso das funcionalidades do *framework*.

• Framework<funcionalidades>: nesta camada ficam situadas duas partes do *framework* - Núcleo do *Framework* e Redes Sociais. O Núcleo do *Framework* contém as funcionalidades que podem ser inseridas no aplicativo como controle de conquistas, pontuação, troféus e etc. Já a parte de Redes Sociais controla as conexões sociais estabelecidas pelo *framework* para compartilhamento de pontos, conexão entre amigos e login por redes socais como *Facebook* e *Google+*.

• Framework<persistência>: na camada de persistência do *framework* ficam os controles de dados relacionados ao banco de dados. Essa parte do *framework* se conecta à camada de funcionalidades para salvar os dados gerados por essa camada ou retornar dados previamente salvos.

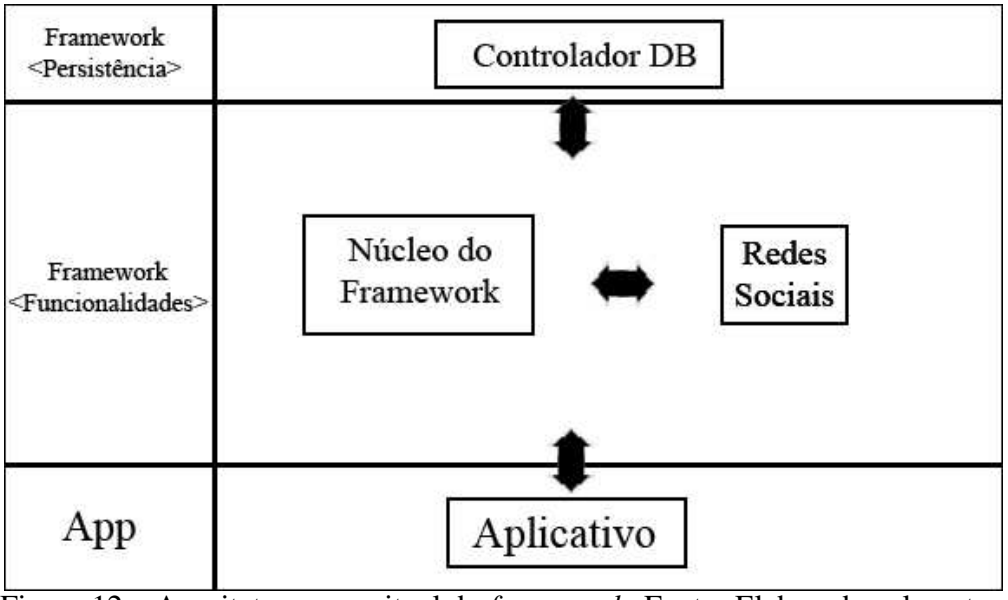

Figura 12 – Arquitetura conceitual do *framework*. Fonte: Elaborada pelo autor.

Essa divisão foi selecionada a partir da percepção de que *frameworks* de persistência de dados já existiam para as diversas plataformas móveis e que desenvolvedores já os utilizam, validando assim a sua funcionalidade. Dessa forma seria mais simples acoplar um *framework* de persistência à desenvolver um completamente. Além disso, foi notado que seria mais robusto para o *framework* utilizar de Application Programming Interface (API)s de redes sociais para realizar a tarefa de conexão social nos aplicativos gamificados, essa escolha foi baseada na já utilização dessas APIs em aplicativos variados, como por exemplos as APIs do *Facebook*, *Google+* e *Twitter*.

5.3 DESENVOLVIMENTO DO MODELO DO *FRAMEWORK*

Para o desenvolvimento do núcleo principal do *Gamifapp*, que comporta todas as características de gamificação do modelo conceitual exceto conexão social, foi feito um estudo do *Octalysis* e dos aplicativos descritos anteriormente para que as principais características fossem extraídas e aplicadas no modelo proposto para o framework de aplicativos móveis. Um framework é uma ferramenta de reuso de código, devido a isso, não é possível englobar todas as características de gamificação necessárias para se obter um aplicativo seguindo completamente as sugestões do *Octalysis* e as suas recomendações de design de projeto para utilização da gamificação. Levando isso em conta, o *Gamifapp* provê as principais características propostas pelo *Octalysis* (CHOU, 2015) que podem ser replicadas com facilidade em diversos contextos de aplicativos móveis gamificados. Dessa forma, caberá ao desenvolvedor utilizar essa ferramenta e outros atributos do *Octalysis* (CHOU, 2015) citadas anteriormente para desenvolver o design completo de sua aplicação e, consequentemente, atingir de maneira eficiente a motivaçao esperada do usuário.

As propriedades selecionadas para serem utilizadas na criação do *Núcleo de Gamificação* do *framework* podem ser vistas na Figura 13.

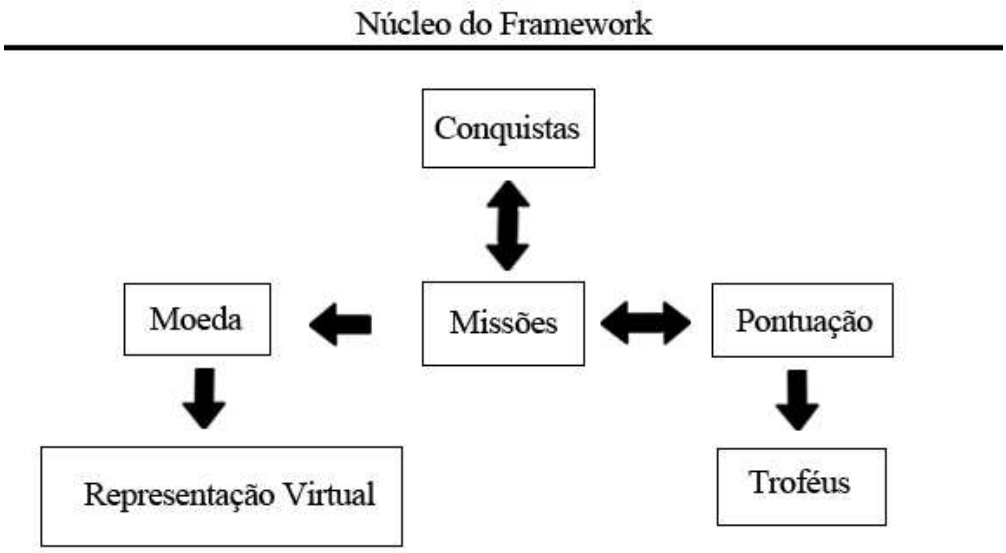

Figura 13 – Núcleo do *Framework*. Fonte: Elaborada pelo autor.

O objetivo desse módulo é prover as ferramentas necessárias para que o desenvolvedor possa utilizar as técnicas de design de projetos gamificados, citadas anteriormente na descrição do Octalysis. Para atingir esse objetivo foram selecionadas as seguintes características:

Missões: parte central do núcleo do *Gamifapp*, as missões são as atividades realizadas pelo usuário dentro do aplicativo. Essas missões são responsáveis por gerar pontuação, conquistas e moeda virtual. As missões também podem ser modificadas a partir do momento que o usuário atingir determinada pontuação, dessa forma a relação missão-pontuação se torna retroalimentada. O módulo de missão serve para auxiliar o desenvolvedor a atingir as características de *Significado* e *Realização* tratadas por Cho (2015).

Pontuação: são os pontos obtidos pelo usuário ao realizar alguma missão. Os pontos são utilizados para obter troféus, gerar novas missões ou aumentar o nível do usuário. A pontuação é uma forma de dar feedback sobre o seu desenvolvimento naquele aplicativo. O módulo de pontuação auxilia no desenvolvimento das seguintes características do *Octalysis* (CHOU, 2015): *Realização*, *Escassez* e *Perda*.

Troféus: é uma das formas de premiação utilizadas pelo *Gamifapp*, eles são obtidos através da pontuação. São uma forma de recompensa pelo desempenho do usuário nas missões realizadas no aplicativo. Os troféus junto aos pontos dão suporte a *Realização* e *Escassez* além de dar suporte a *Imprevisibilidade* junto às *Conquistas*.

Conquistas: é a segunda forma de premiação utilizada pelo *Gamifapp*. Diferente dos troféus, as conquistas não dependem da pontuação e sim de uma missão específica. As conquistas são recebidas ao realizar um desafio mais específico dentro de uma missão. As conquistas se conectam com as missões para dar apoio a aplicação dos conceitos de *Empoderamento*, *Propriedade* e *Significado*.

Moeda: é um dinheiro virtual recebido ao completar missões e utilizado para obter itens para incrementar o avatar do usuário ou para comprar novas chances de realizar missões. As moedas virtuais são uma ferramenta de suporte ao desenvolvedor que deseja atender às características de *Propriedade* e *Realização* ao se conectar com a representação virtual.

Representação Virtual: é uma forma de representar o usuário dentro do aplicativo. Por meio de um avatar o usuário se sente mais conectado à aplicação e ganha, por conseguinte, um objetivo extra de personalização e evolução desse avatar. Junto às moedas dão suporte às características de *Propriedade* e *Realização*.

Um dos atributos mais importantes citados no *Octalysis* (CHOU, 2015) é a interação social, entretanto, ela não foi selecionada como parte do núcleo central do *framework*. Como demonstrado na arquitetura de alto nível do *Gamifapp*, o núcleo central do *framework* se conecta a um núcleo específico para interação social chamado de núcleo de redes sociais. A escolha de

separar esse núcleo da parte central foi feita para dar a possibilidade do usuário utilizar APIs de redes sociais já existentes por meio da utilização dos componentes do núcleo de redes sociais.

5.4 IMPLEMENTAÇÃO - TROFÉUS E PONTUAÇÃO

Para validar a arquitetura proposta anteriormente para o Gamifapp foram selecionados dois módulos na plataforma *Android*. Os módulos selecionados para implementação e posterior validação foram os módulos de *Troféus* e *Pontuação*. Eles foram selecionados, pois são normalmente utilizados em aplicações gamificadas como *Duolingo*, *Fitocracy* e *Habitica*, aplicativos citados anteriormente e utilizados para extração de características importantes na etapa de criação da arquitetura.

Como descrito na seção de *Metodologia* foram feitos dois ciclos no desenvolvimento do *framework*. No primeiro ciclo o *framework* foi feito pensado no modelo caixa-branca onde o desenvolvedor teria acesso ao *framework* por meio de classes genéricas de *UsuarioGenerico* e *TrofeuGenerico*. Sobrescrevendo métodos dessas classes o desenvolvedor poderia moldar o *framework* para as suas necessidades enquanto estivesse desenvolvendo o seu aplicativo. Entretanto, foi notado na etapa de avaliação do modelo que seria difícil posteriormente estender todas as conexões necessárias para uma implementação completa do *Gamiapp*. Além disso, também foi notado que o usuário do *framework* teria mais dificuldade na utilização devido à falta de centralização do acesso às ferramentas do *Gamifapp*, portanto quanto maior fossem as funcionalidades mais complexo seria utiliza-lo.

Levando em conta os problemas descritos acima, foi feita uma nova versão do *framework* num modelo caixa-cinza. Utilizando o padrão de projeto *Facade* (fachada) foi possível centralizar a conexão do desenvolvedor com o *framework* por meio da classe Gamifapp, dessa forma todos os módulos teriam uma classe onde se conectar e também existiria uma classe de acesso ao desenvolvedor, facilitando assim a utilização das ferramentas providas pelo *Gamifapp* para a implementação de um aplicativo gamificado (Figura 14).

Para uma boa qualidade do código do *framework* seguindo padrões de desenvolvimento foram selecionados alguns padrões de projeto para a sua implementação. Foram eles

Facade, *Data Access Object*, *Value Object* e *Template Method*. Além dos padrões de projeto, foram selecionadas duas das técnicas de desenvolvimento no capítulo de *Framework*, são elas *Reflexão* e *Linguagem específica de domínio*.

O Padrão de projeto *Facade*, como explicado anteriormente, cria uma classe de fachada para facilitar o acesso do desenvolvedor às funcionalidades disponibilizadas pela classe abstrata *Gamifapp*. Dessa forma toda a conexão entre o aplicativo e o *framework* ficam centralizados na classe *Gamifapp* e sua extensão criada pelo desenvolvedor, dentro da sua aplicação.

O *Value Object* foi utilizado para encapsular os dados dos Troféus e Usuários nas classes *TrofeuVO* e *UsuarioVO*, para que esses dados pudessem ser passados entre as camadas de *framework* e aplicação por meio da classe Gamifapp. A implementação dessas classes pode ser vista nos Apêndices A e B.

O padrão *Data Access Object* foi utilizado em conjunto com o *framework* selecionado para persistência de dados, o Lightweight Object Relational Mapping (ORMLite), para fazer o gerenciamento entre os dados utilizados pela aplicação e a sua persistência no banco de dados. Foram criadas três classes usando esse padrão, são elas *UsuarioDAO*, *TrofeuDAO* e *UsuarioTrofeuDAO*. Um exemplo da implementação delas pode ser vista na classe *UsuarioDAO* no Apêndice C. O *framework* ORMLite foi selecionado devido à sua confiabilidade, por ser um *framework* difundido e utilizado por desenvolvedores de aplicativos ao redor do mundo. A utilização do ORMLite por diversos programadores traz confiabilidade a esse módulo, o que o torna uma boa opção para o componente de persistência de dados.

O último padrão de projeto a ser utilizado foi o *Template Method*. Ele funciona criando uma classe *template* que faz chamadas internas à classes abstratas, essas que serão implementadas pelo desenvolvedor nos *hot-spots*. É o método *template* que permite a customização do *framework* para o contexto do aplicativo que o desenvolvedor deseja. O método *template* da classe abstrata *Gamifapp* é o método *addPontos* (Código-fonte 1). Ele faz chamadas aos métodos concretos *calculaPontos* e *checkTrofeus* que serão implementados na classe concreta feita pelo desenvolvedor ao herdar da classe abstrata *Gamifapp*.

```
1 public boolean addPontos(Object object, int pontos){
2         UsuarioVO usuarioVO = null;
3         if(object instanceof String){
4             usuarioVO = usuarioDAO.getUsuario((String) object);
```

```
5           }else if (object.getClass() == Integer.class){
6               usuarioVO = usuarioDAO.getUsuario(Long.valueOf((Integer
                    )object));
7           }
8           if(usuarioVO == null){
9               return false;
10          }
11          Boolean newTrophy = false;
12          usuarioVO.setPontos(calculaPontos(usuarioVO, pontos));
13          usuarioDAO.update(usuarioVO);
14          List<TrofeuVO> listTrofeus = trofeuDAO.getTrofeus();
15          for(TrofeuVO tVO : listTrofeus){
16              if(checkTrofeu(tVO,usuarioVO)){
17                  usuarioTrofeuDAO.add(new UsuarioTrofeu(usuarioVO,
                        tVO));
18                  newTrophy = true;
19              }
20          }
21          return newTrophy;
22      }
```

Código-fonte 1 – Método *addPontos: Template Method*

Além dos padrões de projeto descritos anteriormente, foram também utilizadas as técnicas *Reflexão* e *Linguagem específica de domínio*. A *Reflexão* foi utilizada no método *addPontos* (Código-fonte 1) para determinar qual o tipo de objeto recebido e assim extrair os dados do banco de dados via *Data Access Object*.

A *Linguagem específica de domínio* foi utilizada no desenvolvimento das classes *TrofeuGenerico* e *UsuarioGenerico*. Essas classes são utilizadas para conectar os *Value Objects* com o aplicativo do desenvolvedor por meio da classe *Gamifapp*. A LED foi utilizada para simplificar a adição dos dados nos *Value Objects* como pode ser visto em Código-fonte 2. Como ilustrado, no lugar de dar um *set* para cada dado a ser inserido, o desenvolvedor apenas utiliza uma *Method Chain* para alcançar o mesmo resultado.

```
1 gamification.addUsuario()
```

```
2       .nome("TCC")
3       .pontos(0)
4       .add();
```

Código-fonte 2 – Adição de dados utilizando LED

Para validar toda a implementação descrita nesse capítulo foi feita a refatoração de dois aplicativos. Esses aplicativos e a forma como o *Gamifapp* foi utilizado para refatorá-los será descrita no próximo capítulo, *Avaliação do Modelo de Framework*.

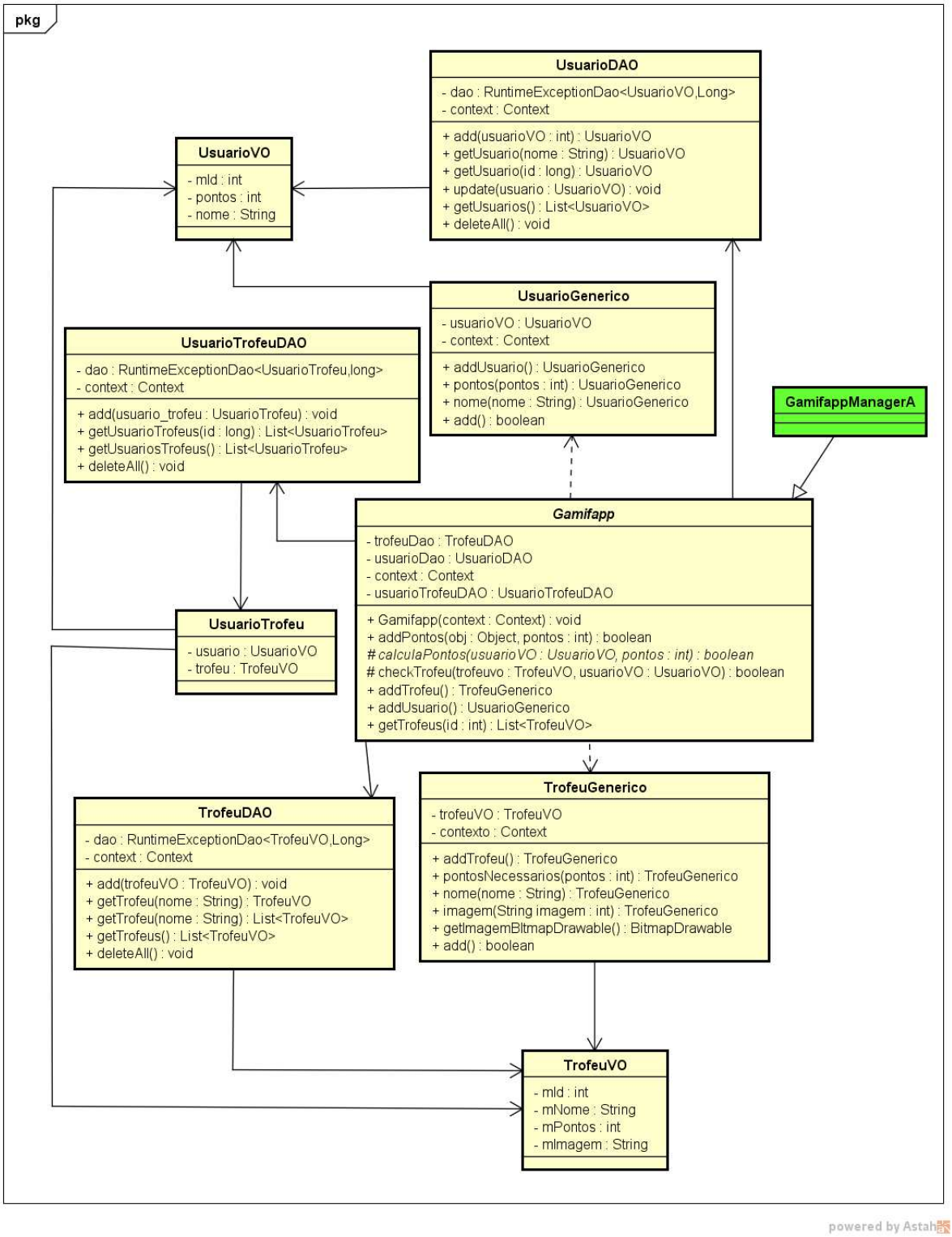

Figura 14 – Diagrama de Classes dos módulos Pontuação e Troféu - *Gamifapp*. Fonte: Elaborada pelo autor.

6 AVALIAÇÃO DO MODELO DE FRAMEWORK

Neste capítulo será retratada a aplicação do *Gamifapp* em dois aplicativos, o *Movie List* e o *Click me!*. Estes aplicativos foram desenvolvidos primeiramente sem a utilização do *Gamifapp* e em seguida foram refatorados utilizando os componentes de *Troféu* e *Pontuação* descritos no capítulo anterior.

6.1 MÉTODO DE VALIDAÇÃO

Para a validação do modelo de *framework* foram implementados dois aplicativos simples, o *Click Me!* e o *Movie List*. O *Click Me!* é um aplicativo simples onde o usuário tem que clicar num *box* para receber pontos e ao salvar esses pontos ele pode receber troféu se alcançar uma pontuação específica (Figura 15). Nos casos de teste foram utilizados 3 troféus, um para 50 pontos, um para 250 e um para 500. O *Movie List* é um aplicativo que contém uma lista de filmes com as datas que eles foram ou serão lançados. O aplicativo permite que o usuário selecione um filme para ver mais informações sobre este e marque se ele quer acompanhar o lançamento deste filme ou se ele já o assistiu. Ao realizar essas ações o usuário pode receber troféus dependendo da quantidade de filmes marcados (Figura 16). No aplicativo de testes o usuário recebe troféu ao marcar um filme e três ou mais filmes.

A seleção de quais componentes do *Gamifapp* seriam implementados na validação aconteceu após a implementação destes dois aplicativos, pois somente após a extração das similaridades e análise de relevância é que foi possível enxergar quais módulos seriam relevantes a serem implementados para essa etapa do desenvolvimento do projeto. Dessa forma, os componentes selecionados foram *Troféus* e *Pontuação*. Eles foram selecionados por serem dois módulos comuns a todos os aplicativos estudados, além de serem utilizados comumente em projetos gamificados.

6.2 UTILIZANDO O GAMIFAPP - CLICK ME! E MOVIE LIST

Como descrito no capítulo 5 o *framework* foi desenvolvido utilizando o padrão de projeto *facade*. Dessa forma toda a conexão entre os aplicativos *Click me!* e *Movie List* ocorre por meio dessa classe. Para realizar essa conexão é necessário extender essa classe abstrata *Gamifapp* por meio de herança, gerando assim a classe onde serão desenvolvidos os métodos abstrados da classe e gerando os métodos utilizados pelo *Template Method*. O aplicativo *Click me!* estende

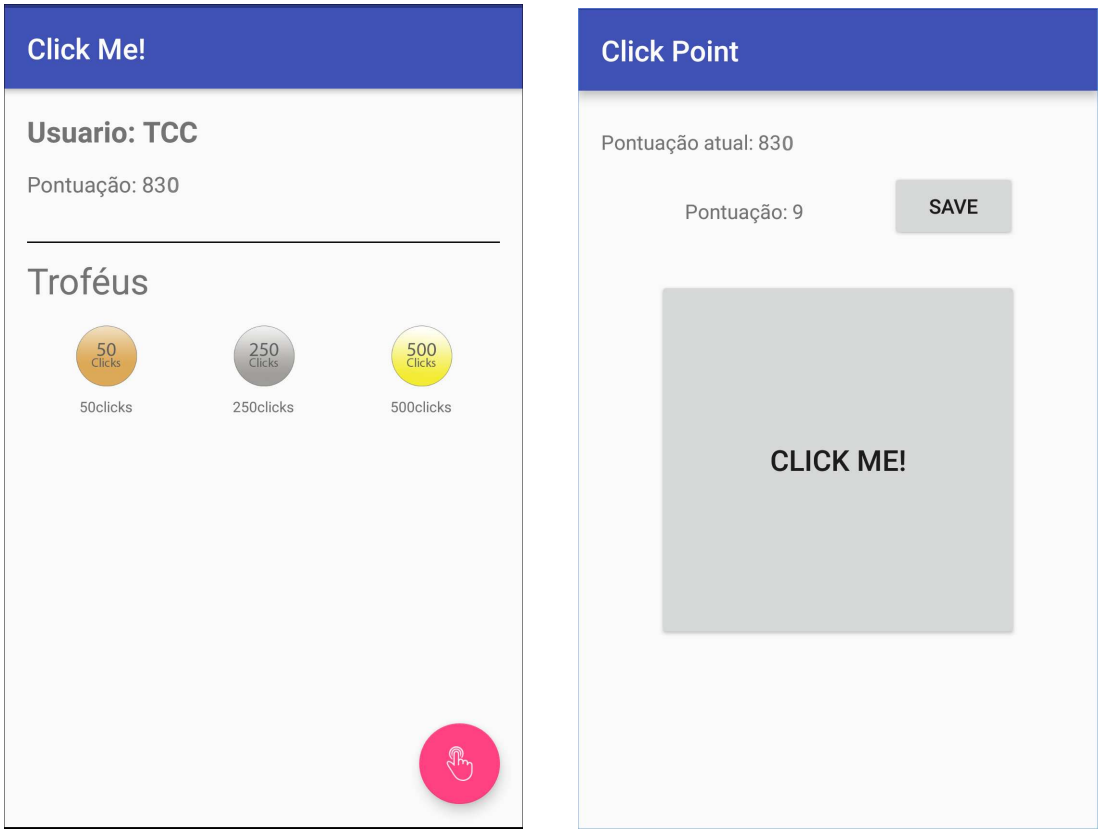

Figura 15 – Click me! - Tela de perfil e de clique para ganhar pontos, respectivamente

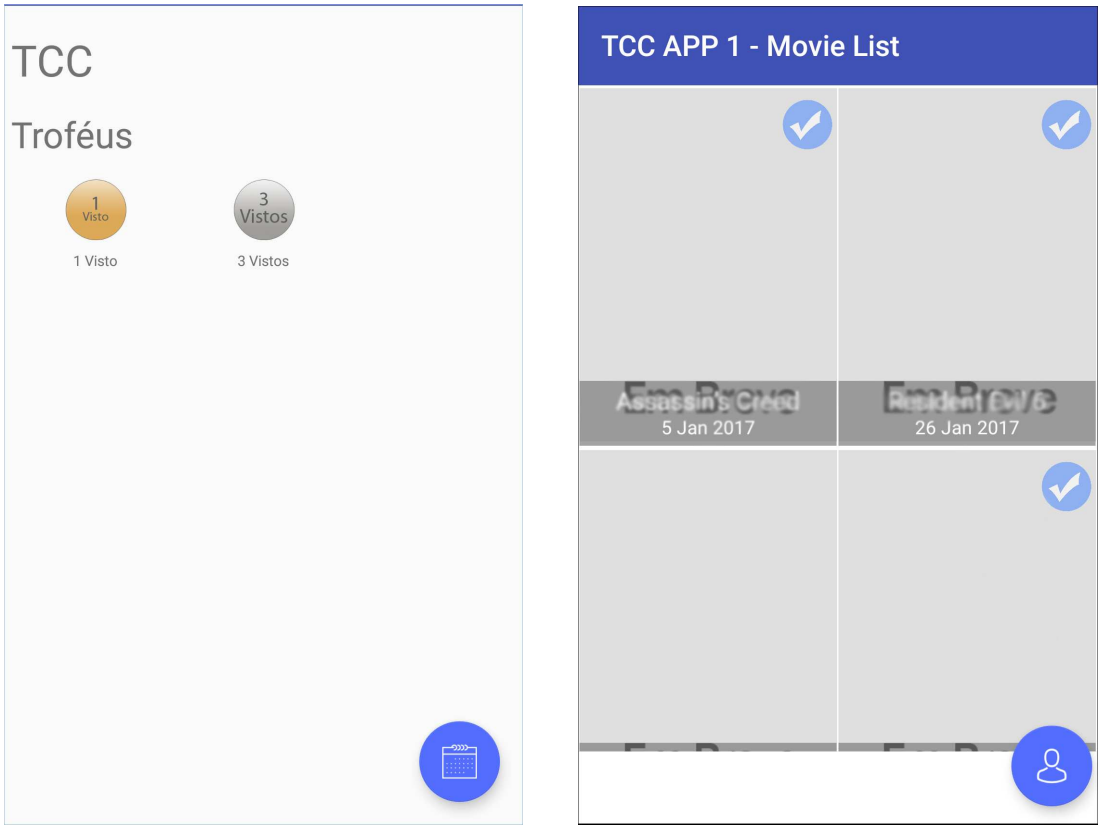

Figura 16 – Movie List - Tela de perfil e de calendário de filmes, respectivamente

essa classe na sua classe *GamifappClickMe* enquanto o *MovieList* na classe *GamifappMovieList*.

Para adição de Usuarios (classe onde também ficam salvos os pontos, que irão interagir com os troféus) e Trofeús ambos os aplicativos utilizam os métodos *addUsuario* e *addTrofeu* que são implementados na classe *Gamifapp* esses métodos retornam a instância das classes *UsuarioGenerico* e *TrofeuGenerico* dando início ao *Method Chain* de criação desses objetos.

Cada troféu adicionado tem a sua pontuação necessária, ou seja, a pontuação que o usuário deve atingir para obtê-lo, seu nome e sua imagem. Ao adicionar uma imagem o desenvolvedor deve adicionar o arquivo da imagem na pasta *Assets* do *Android* (criando-a caso ela não exista) e passar o nome dessa imagem como parâmetro no momento de criação do troféu (Código-fonte 3).

```
1    private void criaTrofeus(){
2        gamification.addTrofeu()
3                .nome("50clicks")
4                .pontosNecessarios(50)
5                .imagem("m50clicks.png")
6                .add();
7
8        gamification.addTrofeu()
9                .nome("250clicks")
10               .pontosNecessarios(250)
11               .imagem("m250clicks.png")
12               .add();
13
14       gamification.addTrofeu()
15               .nome("500clicks")
16               .imagem("m500clicks.png")
17               .pontosNecessarios(500)
18               .add();
19   }
```

Código-fonte 3 – Exemplo da utilização do *Gamifapp* para adição de Troféus

Na criação do *UsuarioVO*, o desenvolvedor deve adicionar apenas o nome e a

quantidade de pontos inicial desse usuário, pois são as únicas características necessárias para os compontentes do framework desenvolvidos para a validação.

Tanto na criação de um *UsuarioVO* quanto na criação de um *TrofeuVO* o desenvolvedor deve utilizar o método *add*, que é responsável por adicionar esse *Value Object* ao banco de dados por meio das classes *Data Access Object*.

A forma de gerar pontos para o usuário é definida pelo desenvolvedor. A interação entre aplicativo e as outras funcionalidades do *framework* é realizada a partir da implementação dos métodos abstratos da classe herdada do *Gamifapp*. No caso do *Click me!* cada clique no botão gera um ponto e esses pontos são salvos na conta do usuário no momento em que ele pressiona o botão *save*. Essa regra é definida ao implementar o método *CalculaPontos* na classe que herda da *Gamifapp*, implementar as interações entre o usuário e o jogo e as chamadas desse método com seus respectivos parâmetros.

O aplicativo *Movie List* permite que o usuário selecione acompanhar um filme que ainda vai ser lançado e receber notificações sobre o lançamento dele. Ao acompanhar o filme o usuário recebe pontos e ao deixar de acompanhar, graças a flexibilização do framework *Gamifapp* é permitida a remoção, facilitando a retirada de um troféu no caso do usuário deixar de acompanhar um filme. Essa regra será mais detalhada posteriormente. Ambos os aplicativos utilizam uma implementação similar do método *calculaPontos* que pode ser vista no Código-fonte 4.

```
1 @Override
2     protected int calculaPontos(UsuarioVO usuarioVO, int pontos) {
3         return usuarioVO.getPontos() + pontos;
4     }
```

Código-fonte 4 – Método calculaPontos

No que tange a percepção de que um usuário atingiu a pontuação necessária para receber um troféu os aplicativos diferem. Isso ocorre devido à necessidade de remoção de troféu de um usuário do *Movie List* caso ele deixe de acompanhar algum filme, enquanto no *Click me!* a pontuação é sempre cumulativa e não decresce em nenhum momento. Entretanto, foi estabelecido que o método *checkTrofeu* seria utilizado da mesma forma em ambos e que a regra seria tratada no perfil do usuário na checagem para exibição dos troféus. Isso foi feito pois adições e retiradas de troféus nas tabelas do banco de dados toda vez que o usuário passasse

a acompanhar ou deixasse de acompanhar um filme seriam muito custosas para um aplicativo móvel. A implementação desse método pode ser vista em Código-fonte 5.

```
1  @Override
2      protected boolean checkTrofeu(TrofeuVO trofeuVO, UsuarioVO
           usuarioVO) {
3          if(trofeuVO.getPontos() <= usuarioVO.getPontos()) {
4              List<TrofeuVO> tList = this.getTrofeus(usuarioVO.getId
                   ());
5              if(tList == null){
6                  return true;
7              }
8              for (TrofeuVO t : tList) {
9                  if (trofeuVO.getNome().equals(t.getNome())) {
10                     return false;
11                 }
12             }
13         }else{
14             return false;
15         }
16         return true;
17     }
```

Código-fonte 5 – Método *checkTrofeu* implementado no *Click me!*

Na seção a seguir será feita uma avaliação do uso do *Gamifapp* na refatoração dos aplicativos.

6.3 AVALIAÇÃO DO GAMIFAPP

Durante a refatoração dos aplicativos foi possível notar que a utilização de um *framework* nesse contexto tem algumas vantagens perceptíveis. A primeira delas é a organização do código, pois enquanto na primeira versão às funcionalidades relacionadas a gamificação tinham que ficar espalhadas em diversas partes do código, o que deixaria mais complicado o seu entendimeneto e manutenção, após a refatoração a maior parte desse código foi removido.

Isso foi feito pois o *framework* já trazia as soluções implementadas, permitindo o uso dessas funcionalidades de forma mais limpa no código. Uma outra vantagem é a facilidade de utilizar um *framework* que segue padrões de projeto e tem apenas um canal de comunicação com o aplicativo, pois dessa forma todo o uso das funcionalidades fica centralizado na utilização do objeto da classe de conexão herdada do *Gamifapp*. Uma outra vantagem é a confiabilidade do código, pois a partir do momento que ele já funciona em um contexto, é mais fácil replicá-lo de forma rápida e funcional num outro contexto e/ou aplicativo.

Por não possuir todos os componentes implementados não foi possível ter uma noção concreta de como seria a utilização de todas as funcionalidades utilizando apenas uma classe de conexão entre aplicativo e *framework*. Entretanto pode-se perceber a flexibilidade que a utilização desse padrão traz no desenvolvimento do *framework* e na utilização dele na implementação dos aplicativos, pois todos os métodos abstratos são implementados em uma mesma classe concreta ao herdar da classe abstrata do *framework* chamada de *Gamifapp*.

7 CONSIDERAÇÕES FINAIS

A revisão bibliográfica feita durante esse projeto demonstrou a necessidade de criação de ferramentas de reuso de software com foco na área de gamificação, pois mesmo existindo mais de vinte *frameworks* de gamificação, não há muito material com foco na área de aplicativos móveis. Desta forma, um framework focado no domínio de aplicativos se torna importante para os desenvolvedores tanto como ferramenta de desenvolvimento, quanto como um facilitador para o conhecimento das formas de utilização de gamificação em aplicações. Isso ocorre pois os detalhamentos das características centrais desse tema são estudados durante o desenvolvimento do modelo do *framework* para atingir resultados que de fato possam alcançar retornos positivos no uso de gamificação em aplicativos.

O *Gamifapp* tem como principal objetivo gerar, por meio de reuso de código, ferramentas que possam dar suporte ao desenvolvimento de aplicativos gamificados. O *Gamifapp* realiza essa tarefa retirando do escopo de desenvolvimento do aplicativo a criação das funcionalidades gamificadas e gerando um conhecimento (*know how*) de como conectar essas funcionalidades. Isso deixa a cargo do desenvolvedor pensar em como integra-las no seu contexto, criando o *design* necessário para alcançar o engajamento dos usuários.

A metodologia aplicada foi de grande importância no desenvolvimento do projeto principalmente graças aos ciclos de desenvolvimento e avaliação, como pôde ser visto durante o Capítulo 5. Eles foram responsáveis por possibilitar a percepção de falhas no projeto inicial, dessa forma foi factível reformular o modelo, tornando-o mais robusto e confiável.

Acredita-se que com a utilização do *Gamifapp* o desenvolvedor poderá alcançar de forma mais fácil e confiavél as funcionalidades necessárias para a implantação da gamificação no seu aplicativo. Levando isso em conta, o modelo proposto no projeto traz ganhos pertinentes aos desenvolvedores de aplicativos, que terão um modelo confiável para obter apoio durante o desenvolvimento de seus aplicativos. A partir do modelo proposto, será possível desenvolver o código dessa arquitetura de *framework* para as variadas plataformas móveis, assim dando suporte aos diversos profissionais da área.

O projeto também contribui ao desenvolver a primeira etapa de implementação desse modelo, feito na plataforma *Android* por meio da códificação dos módulos *Troféus* e *Pontuação*. Pois dessa forma, foi possível aplicar os componentes do *Gamifapp* implementados na refatoração de dois aplicativos, o *Click me!* e o *Movie List*. Isso possibilitou a validação da utilização do *framework* no desenvolvimento de aplicativos, mostrando que ele cumpre o que

propõe em sua arquitetura, ao atingir os resultados propostos durante o refatoramento. Ademais, o fato de utilizar padrões de projeto conhecidos e técnicas de desenvolvimento de *framework* durante a implementação dos módulos demonstra a possibilidade de gerar uma ferramenta robusta para os desenvolvedores.

Os módulos implementados no desenvolvimento do modelo já trazem funcionalidades que podem ser utilizadas por desenvolvedores. Entretanto, para o desenvolvimento completo de um aplicativo gamificado será necessária a utilização dos outros módulos. Outros objetos de pesquisa que não puderam ser desenvolvidos durante este projeto foram, a expansão do código para outras plataformas e análise de eficiência no uso do *framework*. Levando isso em conta, propõe-se os seguintes projetos futuros:

1. A implementação dos outros componentes do núcleo de gamificação;
2. A conexão com APIs de redes socais para facilitar a interação social sugerida no *framework Octalysis*;
3. A validação do modelo de *framework* em outras plataformas móveis como *Windows 10* e *iOS*;
4. Avaliação do framework a partir do uso de outros desenvolvedores, para realizar possíveis melhorias.

REFERÊNCIAS

ARAÚJO, C. **Entendendo Anotações- Revista Easy Java Magazine 25**. 2012. Disponível em: <http://www.devmedia.com.br/entendendo-anotacoes-revista-easy-java-magazine-25/26772>. Acesso em: 11 Jun. 2012.

CHOU, Y.-K. **Octalysis – Complete Gamification Framework**. 2015. Disponível em: <http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/>. Acesso em: 16 Set. 2016.

CSIKSZENTMIHALYI, M.; HARPER; ROW. Flow: The psychology of optimal experience. **Global Learnign Communities 2000**, 1990.

DETERDING, S.; DIXON, D.; KHALED, R.; NACKE, L. From game design elements to gamefulness: Defining "gamification". In: **Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments**. New York, NY, USA: ACM, 2011. (MindTrek '11), p. 9–15. ISBN 978-1-4503-0816-8. Disponível em: <http://doi.acm.org/10.1145/2181037.2181040>. Acesso em: 11 Jun. 2016.

FARDO, M. L. **A GAMIFICAÇÃO COMO MÉTODO: ESTUDO DE ELEMENTOS DOS GAMES APLICADOS EM PROCESSOS DE ENSINO E APRENDIZAGEM**. Dissertação (Mestrado) — Universidade de Caxias do Sul, 2013.

FILHO, C. B. Reuso de software - baseado nas diretrizes do mps.br nível e. Universidade Estadual de Londrina, 2013.

GONÇALVES, L. S. Um framework de efeitos sonoros para a plataforma android : Droidsoundfx. Universidade do Estado da Bahia, 2013.

HAMARI, J.; KOIVISTO, J.; SARSA, H. Does gamification work? — a literature review of empirical studies on gamification. **Proceedings of the Annual Hawaii International Conference on System Sciences**, p. 3025–3034, 2014.

JOON, Y. Y.; YONG, K. S.; JA, C. G.; SOOK, C. E.; JIN, K. C.; DONG, K. S. A uml-based object-oriented framework development methodology. 1998.

LARMAN, C. **Utilizando UML e Padrões**. Bookman, 2007. ISBN 9788560031528. Disponível em: <https://books.google.com.br/books?id=ZHtcynS03DIC>. Acesso em: 20 Set. 2016.

LAW, F. L.; KASIRUN, Z. M.; GAN, C. K. Gamification towards sustainable mobile application. **2011 5th Malaysian Conference in Software Engineering, MySEC 2011**, p. 349–353, 2011.

LIMA, R. A. de J. Percept: Um framework para o desenvolvimento de aplicações de computação perceptiva. Universidade do Estado da Bahia, 2015.

MARKIEWICZ, M. E.; LUCENA, C. J. P. de. Object oriented framework development. **Crossroads**, v. 7, p. 3–9, 2001.

MORA, A.; RIERA, D.; GONZÁLEZ, C.; ARNEDO-MORENO, J. A literature review of gamification design frameworks. In: IEEE. **Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on**. [S.l.], 2015. p. 1–8.

RIVERA, J.; MEULEN, R. van der. **Gartner Says Mobile App Stores Will See Annual Downloads Reach 102 Billion in 2013**. 2013. Retrieved March 03, 2016, from http://www.gartner.com/newsroom/id/2592315. Acesso em: 10 Mai. 2016.

STANOJEVIĆ, V.; VLAJIĆ, S.; MILIĆ, M.; OGNJANOVIĆ, M. Guidelines for framework development process. In: IEEE. **Software Engineering Conference in Russia (CEE-SECR), 2011 7th Central and Eastern European**. [S.l.], 2011. p. 1–9.

UPADHYAYA, P. **By the Numbers: Sizing Up the App Economy in 2015**. 2016. Retrieved March 03, 2016, from http://rewrite.ca.com/us/articles/application-economy/by-the-numbers-sizing-up-the-app-economy-in-2015.html. Acesso em: 10 Mai. 2016.

APÊNDICES

APÊNDICE A – Classe TrofeuVO

```
1  class TrofeuVO implements Serializable {
2
3      public static final String FIELD_ID = "id";
4      public static final String FIELD_NOME = "nome";
5      public static final String FIELD_IMG = "imagem";
6      public static final String FIELD_PONTUACAO = "pontuacao";
7
8      @DatabaseField(generatedId = true, columnName = FIELD_ID)
9      private int mId;
10
11     @DatabaseField(columnName = FIELD_NOME)
12     private String mNome;
13
14     @DatabaseField(columnName = FIELD_IMG)
15     private String mImagem;
16
17     @DatabaseField(columnName = FIELD_PONTUACAO)
18     private int mPontos;
19
20     private BitmapDrawable imagemDraw;
21
22     public TrofeuVO(){
23         mNome = null;
24         mImagem = null;
25         mPontos = -1;
26     }
27
28     public int getmId() {
29         return mId;
30     }
```

APÊNDICE B – Classe UsuarioVO

```
1   public class UsuarioVO implements Serializable {
2   public static final String FIELD_PONTOS = "pontos";
3   public static final String FIELD_NOME = "nome";
4   public static final String FIELD_ID = "id";
5
6   @DatabaseField(generatedId = true ,columnName = FIELD_ID)
7   private int mId;
8
9   @DatabaseField(columnName = FIELD_PONTOS)
10  private int pontos;
11
12  @DatabaseField(columnName = FIELD_NOME)
13  private String nome;
14
15  public UsuarioVO(){}
16
17  public int getId() {
18      return mId;
19  }
20
21  public void setId(int mId) {
22      this.mId = mId;
23  }
24
25  public int getPontos() {
26      return pontos;
27  }
28
29  public void setPontos(int pontos) {
30      this.pontos = pontos;
31  }
32
33  public String getNome() {
```

```
34          return nome;
35      }
36
37      public void setNome(String nome) {
38          this.nome = nome;
39      }
40  }
```

APÊNDICE C – Classe UsuarioDAO

```
1  public class UsuarioDAO {
2
3      private RuntimeExceptionDao<UsuarioVO, Long> dao;
4      private Context context;
5
6      public UsuarioDAO(Context context)
7      {
8          this.context = context;
9          this.dao = new DatabaseHelper(context).
               getUsuarioExceptionDao();
10     }
11
12     public void add(UsuarioVO usuario)
13     {
14         this.dao.create(usuario);
15     }
16
17     public UsuarioVO getUsuario(Long id)
18     {
19         return dao.queryForId(id);
20     }
21
22     public UsuarioVO getUsuario(String nome)
23     {
24         QueryBuilder<UsuarioVO, Long> qb = dao.queryBuilder();
25         try {
26             UsuarioVO usuario = (qb.where().eq(UsuarioVO.FIELD_NOME
                   , nome).query()).get(0);
27             return usuario;
28         } catch (SQLException e) {
29             throw new RuntimeException(e);
30         }
31     }
```

```
32
33      public void update(UsuarioVO usuario){
34          this.dao.update(usuario);
35      }
36
37      public List<UsuarioVO> getUsuarios()
38      {
39          try
40          {
41              List<UsuarioVO> usuarios = new ArrayList<>(dao.
                    queryForAll());
42              return usuarios;
43          }
44          catch (RuntimeException e)
45          {
46              throw new RuntimeException(e);
47          }
48      }
49
50      public void deleteAll()
51      {
52          dao.delete(getUsuarios());
53      }
54
55  }
```