CURSO DE SISTEMAS DE INFORMAÇÃO

APLICATIVO MÓVEL PARA APRENDIZADO INICIAL DE PROGRAMAÇÃO COM USO DE GAMIFICAÇÃO

JULIETA CHAVES DA ROCHA SILVA PAZ

Palmas

2022

CURSO DE SISTEMAS DE INFORMAÇÃO

APLICATIVO MÓVEL PARA APRENDIZADO INICIAL DE PROGRAMAÇÃO COM USO DE GAMIFICAÇÃO

JULIETA CHAVES DA ROCHA SILVA PAZ

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Jânio Elias Teixeira Junior.

Palmas

2022

CURSO DE SISTEMAS DE INFORMAÇÃO

APLICATIVO MÓVEL PARA APRENDIZADO INICIAL DE PROGRAMAÇÃO COM USO DE GAMIFICAÇÃO

JULIETA CHAVES DA ROCHA SILVA PAZ

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Jânio Elias Teixeira Junior.

Prof. Me. Jânio Elias Teixeira Júnior
Orientador

Professor
Convidado 1

Professor
Convidado 2

Palmas
2022

Dedico este trabalho à minha família e entes queridos, por todo o incentivo e carinho.

Agradecimentos

A Deus pela vida, aos meus pais que são a razão de ter chegado aqui. A todos os que passaram pelo meu caminho, principalmente à minha família e amigos que sempre me ajudaram a seguir em frente. Aos professores envolvidos neste projeto, por toda a ajuda, paciência e motivação durante seu desenvolvimento, em especial ao meu Professor Orientador, Me. Jânio Elias Teixeira Júnior.

“Isn’t it funny how day by day nothing changes but when you look back everything is different.“

(C. S. Lewis)

Resumo

Palavras-chave: Aplicações híbridas, Gamificação, Dispositivos móveis.

Abstract

Key-words: Cross-platform applications, Gamification, mobile devices.

Lista de ilustrações

Lista de abreviaturas e siglas

API - *Application Programming Interface*

Sumário

1 Introdução

Ao se iniciar os estudos na área de computação, é essencial construir bases fortes no início para que com o passar do tempo e com o aumento da complexidade nos conceitos de programação, estes sejam aprendidos com mais facilidade. A gamificação pode ser um aliado no aprendizado destes conceitos iniciais, ajudando o aluno a mater-se motivado e engajado nos estudos.

Com o constante desenvolvimento da computação, tem sido cada vez mais necessário o recrutamento de profissionais qualificados, principalmente nas áreas de programação, engenharia de software e banco de dados (DERUS; ALI, 2012). Porém, formar os profissionais necessários pode ser um processo complexo devido à dificuldade encontrada pelos alunos no aprendizado de programação. Alguns dos fatores que podem estar relacionados são a grande quantidade de alunos na turma, ou falta de recursos, e a falta de instrução individual. O que leva muitos alunos a desistirem dos cursos de programação (LAHTINEN; ALA-MUTKA; JÄRVINEN, 2005).

Derus e Ali (2012) demonstra que os estudantes de programação apontam terem maiores dificuldades no entendimento de conceitos básicos da estrutura de programação, no desenvolvimento de programas para completar alguma tarefa e no aprendizado da sintaxe de linguagens de programação.

Já Lahtinen, Ala-Mutka e Järvinen (2005) encontra os seguintes resultados em seu estudo. A dificuldades mais relatadas por estudantes no aprendizado de conceitos de programação são recursividade, ponteiros e referências, tipos de dados abstratos, tratamento de erro e utilização de bibliotecas da linguagem de programação.

Atualmente, os dispositivos móveis, como celulares e tablets, são largamente utilizados fazendo com que cada vez mais os sistemas computacionais sejam construídos para dispositivos móveis tendo em vista que são os aparelhos mais utilizados pela população.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo geral desenvolver uma aplicação móvel híbrida utilizando gamificação para o ensino de conceitos iniciais de programação com o intuito de facilitar sua aprendizagem para iniciantes de programação.

1.1.2 Objetivos Específicos

- Elaborar uma revisão bibliográfica sobre os assuntos relacionados a este trabalho.
- Estudar e descrever como a gamificação pode melhorar o desempenho no aprendizado de programação.
- Selecionar e descrever um conjunto de conceitos básicos na aprendizagem de programação necessárias para o desenvolvimento da aplicação.
- Produzir os diagramas de classe e de caso de uso.
- Construir uma prototipação para a aplicação a ser desenvolvida.
- Construir um WebService para dar suporte à aplicação móvel.
- Desenvolvimento de um aplicativo móvel.

2 Referencial Teórico

2.1 Gamificação

O termo Gamificação (do inglês Gamification) vem sendo oficialmente utilizada desde 2010 para descrever a aplicação de elementos de jogos em atividades que não envolvem jogos (FADEL et al., 2014). De forma mais detalhada, como apontado por Burke (2014), a gamificação é a utilização digital de mecanismos e *design* de experiência para engajar e motivar pessoas a alcançar seus objetivos. Alguns dos mecanismos de jogos que podem ser utilizados são distintivos e tabelas de classificação ou *ranking*.

Os mecanismos de jogos são como motores motivacionais para os indivíduos, o que contribui para o engajamento deles. Estar dedicado nas tarefas designadas é o que mede o nível de engajamento de uma pessoa, e é este engajamento que mede o sucesso do processo de gamificação (FADEL et al., 2014).

McGonigal (2011) discorre sobre as razões pela qual os seres humanos se interessam tanto por jogos. Para o autor, existem quatro características que definem um jogo, sendo elas o objetivo do jogo, as regras, o sistema de *feedback* e a participação voluntária. Todas as outras características de jogos tentam reforçar e melhorar estes quatro elementos principais. A exemplo disto, uma história bem contada faz com que o objetivo se torne mais atraente, as métricas de pontuação do jogo mais complexas fazem com que o sistema de *feedback* se torne mais motivador, conquistas e nívels aumentam as chances de sucesso, jogos *multiplayer* e massivamente *multiplayer* tornam-se mais imprevisíveis quando jogados por um tempo considerável. A definição destes quatro pilates de acordo com McGonigal (2011) é a seguinte:

- O **objetivo** é o resultado final para o qual o jogador trabalha para atingir, ele fornece um senso de propósito. Serve para manter sua atenção e o orientar no decorrer do jogo.
- **Regras** descrevem a maneira na qual os jogadores podem alcançar o objetivo. São delimitadas removendo ou limitando as maneiras óbvias para se chegar ao objetivo
- O **sistema de *feedback*** diz ao jogador o quão próximo ele está do objetivo. Pode estar presente na forma de pontos, níveis ou barra de progresso.
- **Participação voluntária** está relacionada à aceitação dos jogadores às regras, objetivo e ao *feedback*. Desta maneira como todos os participantes sabem o que se passa, é formado um consenso entre os múltiplos jogadores para que participem do

mesmo jogo. Além disso, o jogador tem a liberdade de entrar e sair quando quiser, fazendo com que a experiência de jogo seja segura e agradável mesmo que o trabalho seja desafiador.

2.1.1 Mecânicas de Gamificação

Um sistema gamificado é formado por uma série de ferramentas que quando usadas corretamente, conseguem ter obter uma resposta positiva dos jogadores. Zichermann e Cunningham (2011) descreve sete desses elementos primários, são eles os pontos, níveis, tabelas de liderança, emblemas, desafios/missões, integração e engajamento.

2.1.1.1 Pontos

De acordo com Zichermann e Cunningham (2011), pontuação é uma estratégia importante e é utilizada em uma variedade de propósitos, podem ser aplicadas acumulando-se pontos compartilhados entre os jogadores ou somente entre o designer e o jogador. Existem diferentes tipos de formas de utilização de pontos, algumas das utilizadas na vida real são:

- Pontuação em dinheiro é usada para indicar a quantidade de dinheiro disponível no banco. Esta pontuação fica visível somente para o jogador e este pode dar pistas de o quanto possui para os outros jogadores através do uso acessórios ou objetos comprados com sua pontuação.
- Pontuação de videogame. Utilizada na maioria dos videogames, é uma forma de pontuação muito mais evidente e está sempre presente no canto da tela para que o jogador saiba o quão perto está do próximo nível, ou para comparar sua pontuação com a de outros jogadores, ou para mostrar o quanto falta para ganhar o jogo. É porém pouco utilizada nos sistemas na vida real da mesma forma que é utilizada em videogames.
- Pontuação de rede social. É apresentada na forma de quantificação de amizades ou seguidores dentro da rede social.

Zichermann e Cunningham (2011) aponta que nos sistemas gamificados, o sistema de pontuação pode ser evidente, direto ou altamente motivacional, porém ele pode ser utilizado no segundo plano do sistema sem estar evidente ao usuário. Nos sistemas gamificados podem ser encontrados cinco tipos de diferentes de pontuação que são descritos por Zichermann e Cunningham (2011). Sendo eles:

- Pontos de Experiencia (do inglês Experience points, também conhecido como XP), é o tipo mais importante de pontuação. Este tipo não funciona como uma moeda no

sistema, mas sua utilidade é classificar, guiar e observar o jogador. Todo o proresso feito pelo usuário dentro do sistema irá agregar XP, e esta pontuação, em geral, não pode ser diminuída. Ao se utilizar este tipo de pontuação nas atividades do sitema, o jogador tem seus objetivos alinhados ao esperado do sistema em longo prazo.

- Pontos resgatáveis (do inglês Redeemable points ou RP) tem natureza diferente do XP pois não é uma pontuação que está sempre subindo. Este sistema é geralmente chamado de moeda, ou dinheiro e é ele quem forma a economia virtual. Por esse motivo, este sistema deve ser observado e administrado de perto para que não hajam problemas de inflação ou deflação muito altos.

- Pontos de habilidade estão relacionados à atividades específicas dentro de um jogo. Funcionam como pontos bonus que permitem que jogador ganhe experiência e ou prêmios por meio de atividades não centrais.

- Pontos de Karma tem como objetivo criar um caminho comportamental altruísta por meio dos jogadores. São principalmente utilizados em rotinas regulares, por exemplo, ganhe três pontos karma fazendo *check-in* mensal.

- Pontos de Reputação são utilizados quando o sistema necessita estabelecer confiança entre duas ou mais partes que não se pode garantir ou administrar. Seu propósito é agir como um *proxy*.

Werbach e Hunter (2012) aborda a importância de se entender a natureza do sistema de pontos para que o objetivo do sistema gamificado seja atingido. Caso o objetivo seja encorajar a competição, pode-se utilizar pontuação que esteja visível para outros jogadores. Ou então se o objetivo é deixar o usuário sempre fisgado pelo sistema de constante feedback, pode-se utilizar pontos para dar um senso de progressão, sem mostrar a pontuação para outros usuários.

Figura 1 – Capturas de tela da aplicação Duolingo demonstrando sistemas de pontuação

Na imagem 1 são apresentadas duas capturas de tela da aplicação Duolingo[1], uma plataforma para o aprendizado de línguas que utiliza várias mecânicas de gamificação que são interessantes para entender de forma prática como são implementadas. Aqui podem ser observadas certos tipos de sistema de pontuação acima citados.

Na primeira captura pode-se observar a presença dos Pontos de experiência (XP), que são atingidos a cada vez que se completa uma lição. Como pode-se ver, após completar a lição, o usuário irá ganhar mais 10 pontos de experiência, e esta pontuação só serve para provar a habilidade adquirida pelo jogador.

Na segunda captura é possível perceber uma loja, e sua moeda são as chamadas jóias que são Pontos resgatáveis que o usuário recebe quando finaliza suas lições. E há algo a ser apontado sobre essa moeda. No passado da história do duolingo, sua moeda era chamada Lingot, porém ela sofreu inflação fazendo com que os usuários tivessem muitos Lingots e não tivessem como gastá-los. Então para melhorar a economia da aplicação, teve-se que criar uma nova moeda e remodelar o sistema como um todo para que continuasse sendo uma plataforma sustentável no ensino de línguas (HELPFULDUO, 2017).

2.1.1.2 Níveis

Os níveis são importantes para que os usuários tenham um marcador que indique onde eles se encontram na experiência de jogo com o passar do tempo. Eles geralmente indicam o progresso feito no jogo (ZICHERMANN; CUNNINGHAM, 2011).

[1] https://pt.duolingo.com/

Ainda segundo Zichermann e Cunningham (2011), níveis de dificuldade não são lineares, mas a dificuldade aumenta de forma curvilínea. A complexidade nos jogos sofre transições de um nível para o outro, e esta estratégia se mostrou extremamente engajadora nos jogos. Ela funciona da seguinte forma, a dificuldade aumenta exponencialmente a cada nível e diminui com o passar do tempo.

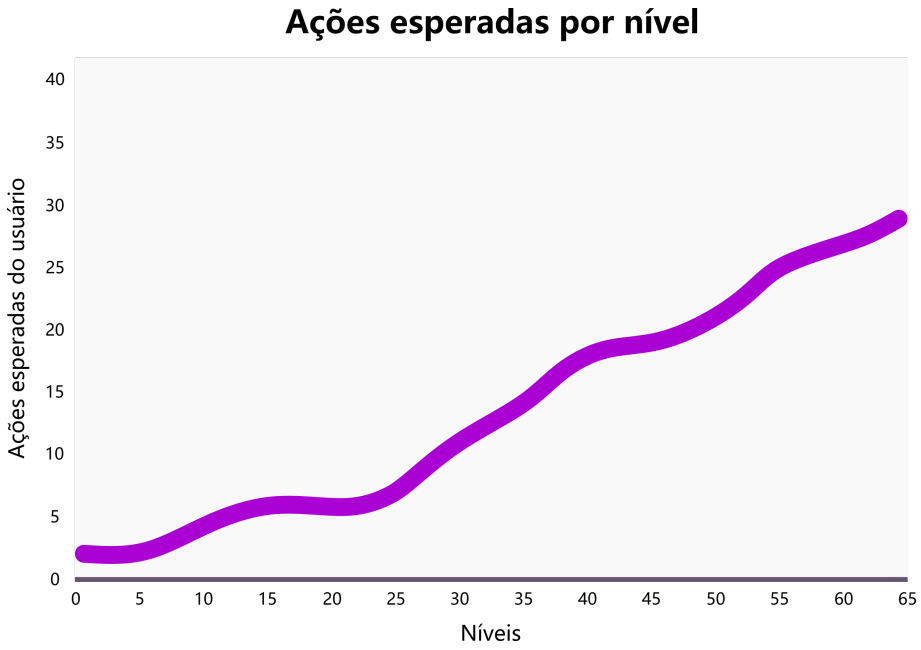

Figura 2 – Níveis de complexidade em jogos
Fonte: Adaptada de Zichermann e Cunningham (2011)

Na imagem 2 é demonstrado um conceito básico da progressão de níveis, sendo que não se trata de uma progressão linear ou exponencial, mas sim uma progressão onde a dificuldade aumenta e diminui para que seja possível o aprendizado. Segundo Zichermann e Cunningham (2011), é importante que os desenvolvedores de sistemas gamificados entendam como os níveis tradicionais nos jogos funcionam pois são uma ferramenta poderosa ao se desenvolver um sistema gamificado.

Para McGonigal (2011), quanto mais se vence em um jogo, menos divertido ele se torna, mas à medida que se falha no jogo, a diversão continua. É como um paradoxo, os jogos são feitos para serem aprendidos e para que o usuário continue melhorando até que consiga dominar o jogo. Porém, quanto mais o jogador domina o jogo, menos desafiador ele se torna. Com níveis mais difíceis, pode-se criar uma idéia de uma diversão desafiadora, que permanece viva por um tempo, mas que à medida que o jogador melhora, é inevitável que o desafio perca sua eficácia.

Alguns dos exemplos de utilização de níveis na vida real citadas por Zichermann e Cunningham (2011) são as cores dos cartões de crédito da Empresa Amperican Express. Onde elas caracterizam o nível que o cartão se encontra, e as pessoas podem facilmente distinguí-los somente pela cor, sendo elas verde, ouro, platina e preta. Outro exemplo

citado são os níveis na Universidade que possuem uma sequência clara tanto para quem faz parte da instituição quanto para quem está fora. Por último, o autor fala sobre os Militares e Escoteiros, que possuem um dos mais aperfeiçoados sistemas de nível entre as instituições. Possuem distintivos e medalhas, até mesmo seus uniformes indicam o nível que se encontram.

Um tipo de sistema de nível são as barras de progresso que estão presentes em toda a Internet. Constumam ser utilizadas na forma de porcentagem para informar o usuário o quão próximo ele se encontra de completar algo. São bastante utilizadas para encorajar usuários de um novo sistema a completar o cadastro de seus dados na plataforma (ZICHERMANN; CUNNINGHAM, 2011).

2.1.1.3 Tabelas de Classificação

As tabelas de classificação tem como objetivo fazerem simples comparações. É de fato um sistema muito fácil de ser reconhecido pela maioria das pessoas sem necessidade de muita explicação pois são compostos por uma lista ordenada e cada elemento possui um nome e uma pontuação ao lado, é fácil de entender que se trata de um sistema de classificação (ZICHERMANN; CUNNINGHAM, 2011).

Zichermann e Cunningham (2011) fala sobre os dois tipos de tabelas de classificação mais utilizadas atualmente. A primeira delas são as tabelas de não-desincentivo, bastante diferente das tabelas de classificação usadas nas máquinas de fliperama há alguns anos. Elas são ferramentas importantes para criar incentivo social ao invés de desincentivar, para tal, o usuário é colocado no meio da tabela, não importando se sua pontuação é 81 ou 200000. Abaixo do jogador, estarão seus amigos, e acima será mostrado o quão próximo o jogador se encontra da próxima melhor pontuação. Caso o jogador realmente esteja entre os top 10 ou top 20, a tabela deve refletir isto com clareza, mostrando sua pontuação literal.

O segundo tipo de tabela de classificação apresentada por Zichermann e Cunningham (2011) são as tabelas de classificação infinitas. Nos jogos de fliperamas, era complicado mostrar a posição de todos os jogadores pois com o passar do tempo, a pontuação de cada jogador é ultrapassada e este jogador cai de posição. Este problema tem algumas soluções na atualidade. O autor fala sobre as maneiras de contornar esta situação com exemplos de tabelas de classificação em jogos. Cita o jogo Doodle Jump que possui um ranking que pode ser visualizado de três maneiras, de forma local, social (pontuação dos amigos) e global. Também cita outro jogo chamado Flight Control, que mostra a pontuação de outros jogadores no mesmo nível do jogador, sendo uma estratégia necessária em um jogo que tem milhões de jogadores.

Werbach e Hunter (2012) apresenta as vantagens e desvantagens de se utilizar tabelas de classificação. Por um lado, os usuários geralmente querem saber onde se

encontram em relação aos outros usuários. Desta maneira, se a performance do usuário é importante para o jogo, uma tabela de classificação irá apresentar publicamente a performance dos usuários. Se implementadas de maneira correta, no contexto adequado, as tabelas de classificação podem ser forte motivadores. Por outro lado, este sistema pode ser um forte desmotivador pois o usuário pode pensar que não conseguirá alcançar os usuários que estão no topo. O autor também fala que estudos mostram que ao se introduzir tabelas de classificação nos ambientes de negócio sem outros elementos de gamificação atrelados, isto causa a redução de performance.

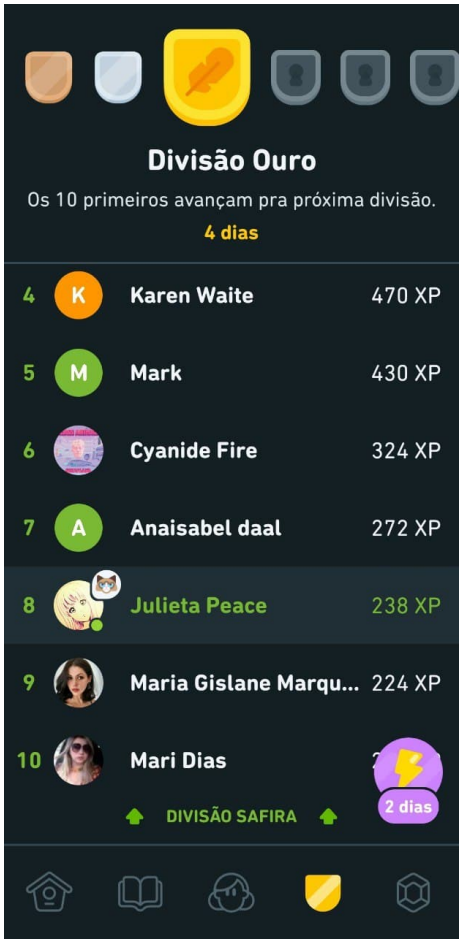

Figura 3 – Captura de tela da aplicação Duolingo demonstrando uma tabela de classificação

A Figura 3 mostra mais uma captura de tela da aplicação Duolingo. Nela pode-se perceber uma tabela de classificação que tem diversos tipos de divisão, pondendo-se observar as divisões bronze, prata, ouro e safira (sendo que as divisões seguintes ainda não foram alcançadas, portanto estão bloqueadas). Este sistema de classificação funciona da seguinte forma: a cada semana os usuários tem a chance de ficar entre os dez primeiros colocados de uma divisão e conseguem isto ganhando pontos de experiência. Caso o usuário fique classificado entre os últimos colocados, ele volta para a divisão anterior, caso o usuário consiga ficar classificado nos top 10, ele vai para a próxima divisão.

2.1.1.4 Distintivos

Para Werbach e Hunter (2012), os distintivos são representações visuais para algum tipo de conquista dentro do sistema gamificado. Alguns deles demarcam algum nível de pontuação atingido (distintivos e conquistas geralmente são os termos utilizados e possuem o mesmo significado). Outros tipos de distintivos indicam certo tipo de atividade, como o serviço Foursquare descrito pelo autor. Trata-se de um sistema que engaja usuários com negócios locais a fazerem check-in em determinadas localizações através de seus celulares. Um usuário alcança o distintivo "Aventureiro"quando faz check-in em dez lugares registrados no sistema Foursquare. Outro sistema citado pelo autor é o Fitbit, que monitora a quantidade de passos o usuário atinge. Por exemplo, o usuário recebe um distintivo ao alcançar 10000 passos em um dia.

2.1.1.5 Integração

A integração trata-se de como será feito o recrutamento de novatos para o sistema. É um passo na gamificação que deve ser muito bem pensado pois são nos primeiros minutos de uso de um sistema que a maioria dos usuário decide se vai ou não continuar a utilizá-lo (ZICHERMANN; CUNNINGHAM, 2011). O autor faz algumas observações sobre esta etapa da gamificação, listadas a seguir:

- O primeiro minuto em que o usuário utiliza o sistema não deve ser utilizado para fazer tutoriais, mas deve ser um momento para o usuário fazer descobertas, sendo capaz de chegar ao objetivo central da aplicação, e isto é possível quando no primeiro minuto de uso, o sistema já ofereça algo que represente um valor ao usuário. Este valor pode ser oferecido em forma de prêmios, conquistas ou itens virtuais (dentre outros).

- O autor também discorre sobre o problema nos sistemas, sendo ele exigir o cadastro do usuário na plataforma para que possa utilizá-la.

- Outro erro que pode ser cometido é inundar o sistema com informações, que fazem o contrário de gerar um efeito positivo para usuários iniciantes, pois estes não tem tempo para ler extensos textos explicativos. A iniciativa a ser tomada deve ser a de deixar o usuário ter uma experiência interativa.

- Por último o autor fala sobre não deixar que o usuário falhe na primeira intereção com o sistema. Como em um jogo, a primeira ação deve ser simples e impossível de se errar, dessa forma, o usuário não se sentirá confuso. Após o sucesso ao se completar a primeira tarefa, o usuário deve receber algum tipo de prêmio ou reforço, por exemplo uma frase de encorajamento como "você fez muito bem!".

- De forma geral, os passos a serem tomados para estabelecer um caminho claro para os jogadores devem ser: ação, premiação, ação, ação, premiação, cadastro no sistema, e por último, convidar amigos.

2.1.1.6 Desafios e Missões

Para Zichermann e Cunningham (2011), os desafios e missões tem como objetivo direcionar os jogadores dentro do mundo da experiência gamificada de modo que consigam se sentir desafiados mas sem estarem perdidos. Implementar desafios no sistema gamificado pode adicionar profundeza e significado para o usuário. Os desafios tem que ser projetados de maneira a oferecer uma diversidade de opções interessantes tanto para usuários que completam desafio atrás desafio em uma tentativa de vencer o jogo, quanto para aqueles que fazem apenas um desafio para manterem o interesse.

Figura 4 – Captura de tela do jogo Brawl Stars ilustrando a utilização de Missões

Na Figura 4 pode-se observar desafios aplicados no jogo Brawl Stars. O propósito destes é guiar o jogador de forma que experiencie diversos modos de batalha, utilize diferentes personagens e socialize com outros jogadores, além de recompensá-lo após o cumprimento das missões. Sem estes desafios, é provável que muitos jogadores perdessem interesse por não conseguirem avançar no jogo por falta de instruções ou motivações.

2.2 Aplicações Móveis Multiplataforma

Aplicações móveis são programas computacionais construídos para serem executados em dispositivos móveis tais como telefones celulares, tablets, ou até relógios. O processo de construção desses programas pode ser complicado tendo em vista que não existe somente um tipo de sistema operacional para estes dispositivos (CLOW, 2019).

São dois os sistemas operacionais mais utilizados na atualidade, sendo eles o Android e iOS. No passado construção de um aplicativo para ambas as plataformas deveria ser feita de forma nativa e para isto eram necessários programadores de Objective-C ou Swift para aplicativos no iOS, e Java ou Kotlin para aplicativos no Android. O que implicava no encarecimento do projeto, já que era necessário manter duas equipes diferentes para o desenvolvimento e manutenção de cada plataforma, e também fazia necessário manter os dois códigos sincronizados (caso houvesse alguma mudança no código para iOS, era necessário fazer a mesma mudança para o código Android) (CLOW, 2019).

Foi então que começaram a surgir soluções para sanar este problema. Algumas delas são Flutter[2], criada pela Google e utiliza Dart, React Native[3], criada pela empresa Facebook e é baseada em javascript, Xamarin[4], criada pela Microsoft e utiliza C# e Firemonkey[5], criada pela Embarcadero, utiliza a linguagem Delphi.

As tecnologias acima citadas são frameworks que permitem o desenvolvimento de aplicativos através de uma base de código ou um só projeto. Também permitem que os custos ao se construir a aplicação sejam diminuídos pois trata-se de um projeto e não n projetos para diferentes plataformas. O projeto tem consistência pois a interface e funcionalidades são as mesmas para todas as plataformas (MIOLA, 2020).

Apesar dos benefícios muito atrativos, é importante saber que a utilização desses frameworks para construção de aplicações móveis também tem seus pontos negativos. Miola (2020) fala sobre dois deles, o primeiro ponto é que, a performance de um aplicativo multiplataforma pode ser reduzida em comparação com um aplicativo nativo pelo fato de uma aplicação nativa ter contato direto com o dispositivo. Essa diferença de performance pode ser causada pois uma aplicação multiplataforma talvez precise utilizar uma ponte para se comunicar com os componentes do Sistema Operacional. Já o segundo ponto negativo citado pelo autor está relacionado à atualizações menos frequentes devido ao fato de que a solução do framework precisa primeiro ser atualizada pelos seus desenvolvedores. Por exemplo a empresa do sistema operacional faz uma grande atualização em seu OS realizando mudanças ou implementando novas funcionalidades, os desenvolvedores da aplicação que utilizam seu framework de desenvolvimento de aplicação multiplataforma precisarão esperar até que o framework esteja atualizado para poder utilizar as novas funcionalidades, e isto pode levar algum tempo.

[2] https://flutter.dev/?gclsrc=ds&gclsrc=ds
[3] https://reactnative.dev/
[4] https://dotnet.microsoft.com/apps/xamarin
[5] https://www.embarcadero.com/br/free-tools/firemonkey-stencils

2.3 Flutter

Flutter é um framework desenvolvido pela Google e abrange o desenvolvimento de aplicativos multiplataforma para Android e iOS. Sua primeira versão estável foi divulgada no final de 2018 e seu diferencial em relação a outras soluções similares é que o Flutter consegue renderizar os próprios componentes de UI (do inglês User Interface, interface do usuário). De forma prática, quando o Flutter exibe um botão, é ele mesmo quem renderiza este botão, não o sistema operacional como comumente ocorre em outras soluções (ZAMMETTI, 2020).

Zammetti (2020) discorre sobre os quatro pilares do framework. O primeiro deles é a linguagem Dart. O segundo é a engine principal do Flutter, uma base de código construída quase que totalmente em C++, de forma que tem performance com desempenho praticamente nativo, e faz uso da engine gráfica Skia para fazer a renderização. O terceiro pilar é a interface que fica sobre o SDK nativo de ambas as plataformas, denominado biblioteca foundation, e tem como objetivo remover as diferença entre as API's das plataformas nativas de forma que a API do flutter lide com as chamadas de funcionalidades nativas, como abrir a câmera do dispositivo. O último pilar são os widgets.

As aplicações construídas com Flutter são executadas a 60 frames por segundo, e 120 frames por segundo caso o dispositivo suporte esta velocidade, e isto mostra o quão rápido e eficiente Flutter pode ser. Além disso, os aplicativos são desenvolvidos a partir de um único código base e são compilados para código de ARM nativo, podem acessar API's específicas do iOS e Android (como localização pelo GPS, biblioteca de imagens) que se comunicam através de canais da plataforma. Todas estas funcionalidades fazem com que aplicações construídas com o framework possuam uma performance como a de aplicativos nativos para ambos os sistemas operacionais (NAPOLI, 2019).

2.3.0.1 Dart

Criada pela Google em 2011. Dart é uma linguagem orientada a objetos, que faz uso garbage-collector e é ideal para criação de aplicações para serem executadas em uma variedade de plataformas. Por exemplo, pode ser executada em um browser na web como JavaScript; também pode ser usada como uma aplicação interpretada, ou em um aplicativo nativo (CLOW, 2019).

Alessandria (2021) discorre sobre a criação da linguagem que tinha como objetivo substituir JavaScript que era padrão para desenvolvimento de aplicações web. O autor desenvolve mais este raciocínio apontando que JavaScript é uma linagem flexível mas sua falta de sistema de tipos e sua gramática aparentemente simples resulta em muitos problemas ao se gerenciar projetos a medida que crescem. O autor ainda fala sobre a a facilidade ao se familiarizar com Dart caso o desenvolvedor já conheça alguma linguagem

com um estilo similar à linguagem C.

2.3.1 Widgets

A Interface de Usuário do Flutter é construída utilizando-se widgets. De acordo com Napoli (2019), pode-se comparar um widget a um bloco de LEGO, ao encaixar e juntar vários blocos é possível criar um objeto, e para alterar a aparência e comportamento do objeto, pode-se adionar ou combinar outros blocos diferentes. Uma aplicação construída com Flutter utiliza esses blocos denominados widgets, cada um deles é uma declaração imutável da Interface de Usuário. De forma mais mais técnica, o autor define widgets como instruções para diferentes partes da Interface de Usuário. Ao se adicionar widgets uns aos outros, cria-se a árvore de widgets. O autor também fala sobre algumas das funções de widgets que o Flutter possui, pode-se citar alguns dos mais comuns em uma aplicação:

- widgets para estruturar elementos, como list, grid, text, e button;
- widgets com elementos de entrada de dados como form, form fields e keyboard listeners;
- aqueles com elementos de estilo como font type, size, weight, color, border e shadow;
- widgets para organização da Interface de Usuário, tais como row, column, stack, centering e padding;
- widgets para interações gestuais de toque;
- elemetos de assets, images e icons.

Existem dois tipos principais de widgets usados ao se construir a Interface de Usuário, sendo eles o StatelessWidget e o StatefulWidget. O primeiro é usado quando não é necessário mudar os estado do widget, e o segundo é utilizado quando os valores (do estado) mudam. Para gerenciar o estado, o framework observa o estado do widget e o compara ao estado anterior para determinar a quantidade mínima de mudanças a serem feitas. Desta maneira o Flutter faz a intermediação entre o estado da UI e reconstrói apenas os widgets que devem ser mudados (NAPOLI, 2019).

Ainda segundo Napoli (2019) alguns dos widgets do tipo StatelessWidget são Text, Button, Icon e Image. Todos estes são utilizados quando não há mudança nos dados, ou seja, a informação inicial é sempre a mesma, são widgets sem estado e com valores finais. Já os widgets to tipo StatefulWidget são usados quando os dados podem mudar com o passar do tempo, fazendo uma chamada ao método setState() para fazer a propagação das mudanças pela UI.

2.4 Web Services

Os web services são os serviços que fazem as conexões entre programas, conectando-os por longas distancias ao redor do globo, transportando grandes quantidades de dados de forma barata. Isto torna a comunicação mais rápida, melhor e mais produtiva, tanto para os negócios, quanto para usuários da rede no cotidiano. Os web services estão melhorando a internet de forma que fazem possível a comunicação entre programas, desta maneira, aplicações em diversas localidades disponíveis na internet podem ser diretamente integradas e contactadas como se fossem parte de um único e grande sistema. (NEWCOMER, 2002).

Richardson e Ruby (2008) descreve dois tipos de arquitetura utilizadas na construção de web services. O primeiro denominado RESTful, arquitetura orientada a recurso. Na arquitetura do tipo RESTful, a informação do método é passada dentro do método HTTP (do inglês Hyper Text Transfer Protocol). Já nas arquiteturas orientadas a recurso, as informações do escopo são enviadas dentro da URI. Desta maneira, podem ser feitas requisições com apenas uma linha de HTTP. O autor fala sobre uma situação onde se implementa o serviço de certa forma resultando em um serviço que não é RESTful, isto ocorre quando o método HTTP não coincide com a informação do método, e caso a informação de escopo não esteja presente na URI, o serviço não pode sr considerado orientado a recurso.

Outra arquitetura abordada por Richardson e Ruby (2008) são as do estilo RPC. Um web service deste tipo aceita envelopes cheio de dados de seu cliente e envia um envelope de volta. Seu método e informação de escopo permanecem dentro do envelope ou nos selos do envelope. O tipo mais popular de envelope utilizado é o HTTP já que todos os tipos de web service relevantes utilizam o HTTP. Outro tipo de formato de envelope muito conhecido é o SOAP, de forma que quando se envia um documento SOAP através do HTTP, o envelope SOAP é enviado dentro do envelope HTTP. Todos os serviços do tipo RPC (do inglês, Remote Procedure Calls) definem um vocabulário totalmente novo, da mesma forma que se escreve um programa de computador, criam-se funções com nomes diferentes. Contrastando com web services RESTful que compartilham um vocabulário comum de métodos HTTP, de modo que cada objeto em um web service deste tipo responde à mesma interface básica.

Richardson e Ruby (2008) classifica os web services pelas tecnologias que os formam, sendo elas:

- HTTP É utilizado por todos os web services mas seu uso é feito de maneira diferente para cada um. Uma requisição a um web service RESTful coloca as informações do método no método HTTP e a informação de escopo na URI. Já os web services do tipo RPC tendem a ignorar o método HTTP, tentando encontrar o método e as informações de escopo na URI, no cabeçalho HTTP ou no corpo-entidade. Também

pode ser usado de maneira a fazer o HTTP um envelope contendo um documento, ou pode ser utilizado como um envelope sem nome contendo outro envelope.

- URI URI (do inglês Uniform Resource Identifier) é o termo definido pelo protocolo HTTP e é utilizado pelo autor para se referir ao termo mais utilizado URL. Sendo que na rede, uma URI também é uma URL. URIs são usadas por todos os tipos de web services, porém de maneiras distintas. Serviços do tipo RESTful orientado a recuro, expõe a URI para cada dado que o cliente possa querer operar. Já serviços do tipo RPC expõe uma URI para cada processo capaz de utilizar RPC, e o único que o faz geralmente é o serviço "endpoint".

- XML-RCP Alguns web services, legados em sua maioria, utilizam XML-RPC sobre HTTP. É um formato de estrutura de dados que representa chamadas de função e retorno de valores. É explicitamente projetado para web services do tipo RPC.

- SOAP Bastante utilizado sobre HTTP, SOAP é um tipo de formato de envelope, assim como o HTTP, porém é baseado em envelopes no formato de XML. Segundo o autor, os web services atuais que utilizam SOAP possuem uma arquitetura do tipo RPC. O que pode ser controverso pois a maioria dos programadores que fazem uso do SOAP tendem a pensar que a arquitetura RPC está em decadência e preferem descrever SOAP como um serviço "orientado a mensagem"ou "orientado a documento". Porém, todos os web services são orientados a mensagem já que o HTTP em si é orientado a mensagem. Um envelope SOAP pode conter qualquer tipo de dado XML, e todos os serviços deste tipo contem uma descrição de uma chamada RPC.

- WSDL WSDL (do inglês Web Service Description Language) é um vocabulário em XML usado para descrever web services baseado em SOAP. A maioria dos serviços SOAP expõe um arquivo WSDL, e estes serviços seriam muito difíceis de se usar sem ter arquivo WSDL como guia. Este arquivo possui uma grande responsabilidade, a de manter a associação com o RPC.

- WADL do inglês Web Application Description Language, o WADL é um vocabulário XML que descreve web services RESTful. O autor diz que o WADL não é tão necessário a estes serviços quanto o RPC para serviços SOAP pois serviços do tipo RESTful são mais simples.

2.5 Python

Python é uma linguagem de programação para uso geral que é muito utilizada como linguagem de script. Python também é muito definido como sendo uma linguagem script orientada a objetos (LUTZ, 2013).

De acordo com Lutz (2013), dentre as razões de escolha de aprendizagem da linguagem Python, o autor cita algumas das mais mencionadas pelos adeptos à linguagem. A primeira seria a qualidade de software, já que para eles Python tem seu foco na legibilidade e coerência. Os códigos em Python são projetados para serem legíveis, tornando-os reusáveis e sustentáveis em comparação com outras linguagens tradicionais. Além de possuir suporte a mecanismos de reuso de software mais avançados como orientação a objeto e funções. Outro motivo descrito pelo autor é a produtividade ganhada com Python se comparada com outras linguagens estáticas ou tipadas como C, C++ ou java. Pode-se escrever muito menos código para realizar as mesmas tarefas em comparação às linguagens citadas, o que resulta em menos códigos para debugar e menos esforço para fazer manutenção.

Outras razões citadas são a portabilidade por Lutz (2013) de código, que pode ser copiado para diversas plataformas sem muitos problemas; Possui uma variedade de bibliotecas padrão pre-instaladas, além de possibilitar a utilização de uma variedade de biblioteca de terceiros que abrem diversas possibilidades de desenvolvimento.

3 Metodologia

Neste capítulo, serão apresentados os procedimentos metodológicos a serem utilizados na pesquisa e na aplicação a ser desenvolvida posteriormente. Para isto, será feita uma contextualização sobre o tema, a descrição do tipo de pesquisa que será realizada no projeto e também os materiais a serem utilizados.

Gerhardt e Silveira (2009) discorre sobre o significado da metodologia. Enuncia que esta tem por objetivo estudar a organização dos caminhos e instrumentos a serem percorridos durante a realização da pesquisa. Diz ainda que há uma diferença entre os termos metodologia e métodos, de maneira que a metodologia tem interesse pela validade do caminho utilizado para atingir o fim proposto pela pesquisa, e os métodos são os procedimentos utilizados na pesquisa.

3.1 Caracterização da Metodologia

Para o desenvolvimento deste trabalho, o método de pesquisa a ser utilizado é o descritivo com abordagem qualitativa. De acordo com (GIL, 2008) um dos principais objetivos de uma pesquisa descritiva é apresentar as características de determinado fenômeno. Já a abordagem qualitativa objetiva descrever a complexidade do problema em questão, sem a utilização de métodos estatísticos para a realização da análise (MAZUCATO, 2018).

O procedimento de pesquisa a ser utilizado é o levantamento bibliográfico e é realizado utilizando-se de fontes como livros, artigos, anais de congressos, documentos, etc. Segundo Mazucato (2018), o levantamento bibliográfico é composto de duas etapas. A primeira etapa corresponde à busca de bibliografia básica, que compreende o momento em que são encontradas as fontes primárias que servem como apoio ao estudo, fazendo com que o pesquisador consiga se habituar aos conceitos e noções a serem estudadas. A segunda etapa é onde se estudam as fontes secundárias, que complementam o conhecimento previamente adquirido de forma a melhor detalhar e complementar o estudo com os assuntos que não ficaram claros ou que não trouxeram o conhecimento específico relacionado ao tema a ser desenvolvido.

Este projeto tem como natureza a pesquisa aplicada, que segundo Gerhardt e Silveira (2009) possui o objetivo de criar conhecimentos para serem aplicados em algo prático que soluciona problemas específicos. De acordo com Gil (2008), a pesquisa aplicada se apoia na pesquisa pura, pois depende de suas descobertas, o que impulsiona seu desenvolvimento. Já que na pesquisa pura procura-se o conhecimento científico sem haver

preocupação com a aplicação deste.

3.2 Etapas do Projeto

A fim de se fazer uma melhor definição das etapas presentes durante o desenvolvimento deste projeto, foi feita uma divisão em quatro etapas como mostrado na Figura 5.

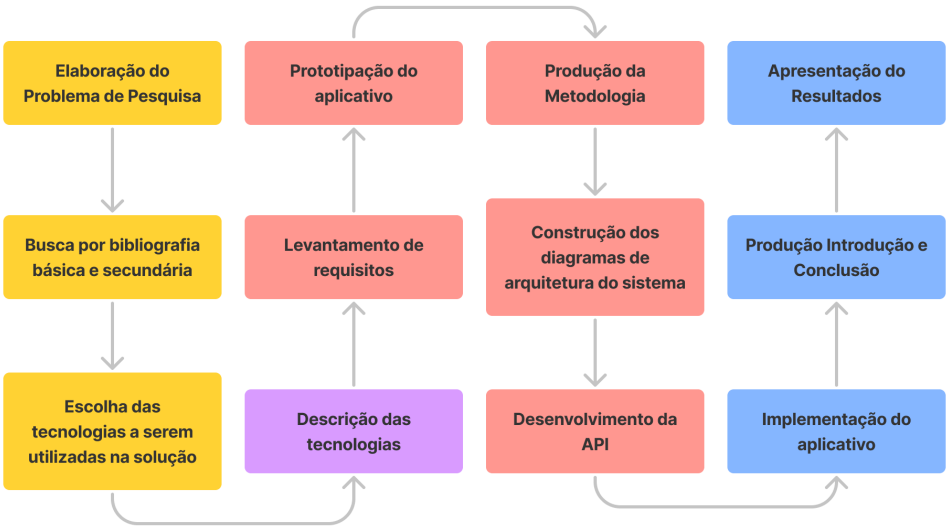

Figura 5 – Etapas da pesquisa Fonte: Autoria própria

A primeira etapa compreende o momento de preparação e delimitação do tema onde será feito o primeiro contato com revisão de literatura que dará suporte à definição do problema. Neste momento, é feita uma pesquisa para se obter familiaridade com o tema abordado por outros autores principalmente por meio de trabalhos acadêmicos. Neste ponto do trabalho, foi decidido que o tema seria desenvolvido tendo como ponto de partida a implementação de uma aplicação móvel com elementos de gamificação, e por isso, faz-se necessário adquirir conhecimentos básicos para entender como estes elementos funcionam e observá-los em outros sistemas gamificados.

Na segunda etapa, será realizada a elaboração do projeto de pesquisa, onde se inicia a escrita dos objetivos e referencial teórico, que irá descrever as tecnologias escolhidas para o desenvolvimento da solução proposta para o problema. O momento onde se constrói o referencial teórico é de extrema importância pois ele deve apresentar com clareza a

sua relação com a resolução do problema. Nesta etapa, será apresentado conceitos como gamificação, tecnologias de desenvolvimento multiplataforma, *web services* e a linguagem de programação que se pretende ensinar por meio da gamificação.

A execução do projeto é a terceira etapa, e nesta fase é posto em prática toda a teorização feita previamente. Durante esta etapa será realizado o planejamento das funcionalidades do sistema, começando com o levantamento de requisitos e prototipação da aplicação móvel, e outros diagramas que possam auxiliar no desenvolvimento. Neste passo também é iniciada a produção da metodologia, que descreverá a maneira como o trabalho será desenvolvido e o tipo de pesquisa utilizada. Após isto, será feita a implementação do sistema proposto, que envolve o desenvolvimento da API que dará suporte ao aplicativo.

No quarto e último passo, será realizada a implementação do aplicativo móvel. Após isto, serão apresentados os resultados obtidos. Englobando o desenvolvimento da introdução, a descrição dos procedimentos realizados na implementação do sistema, as dificuldades encontradas durante o projeto e a conclusão que reflete as experiências do autor ao se desenvolver o trabalho. Por fim serão feitas as correções finais e a apresentação do projeto.

3.3 Materiais

Tabela 1 – Materiais

Descrição	**Modelo**
Sistema Operacional	Windows 10 Home Single Language

4 Resultados

4.1 Levantamento de Requisitos

4.1.1 Requisitos Funcionais

Tabela 2 – Requisitos Funcionais

ID	**Descrição**
RF001	Gerenciar lições (cadastrar / editar / deletar lições)
RF002	Gerenciar administradores (cadastrar / editar / deletar administradores)
RF003	Gerenciar usuários (cadastrar / editar / deletar usuários)
RF004	Gerenciar distintivos (cadastrar / editar / deletar distintivos)
RF005	Gerenciar desafios semanais (cadastrar / editar / deletar desafios)
RF006	Permitir usuários não cadastrados utilizar o aplicativo
RF007	Acumular pontos de experiência ao se completar uma lição
RF008	Mostrar notificação para lembrar o usuário de praticar diariamente
RF009	Mostrar tabela de classificação
RF0010	Mostrar distintivos para indicar conquistas dos usuários
RF0011	Usuários podem realizar auto cadastro
RF0012	O usuário pode reforçar uma lição para ganhar 3 estrelas e acumular mais pontos de experiência

4.1.2 Requisitos Não Funcionais

Tabela 3 – Requisitos não Funcionais

ID	**Descrição**
RNF001	A API REST deve ser construída utilizando-se do Framework Flask e linguagem Python
RFN002	O arquivo de comunicação utilizado pela API deve ser do tipo JSON
RFN003	A aplicação móvel deve ser desenvolvida com tecnologia de desenvolvimento multiplataforma através do Framework Flutter
RFN004	A linguagem a ser ensinada no aplicativo deve ser Python versão 3
RFN005	O aplicativo deve funcionar quando houver conexão com a internet

4.2 Diagramas

4.2.1 Diagrama de Caso de Uso

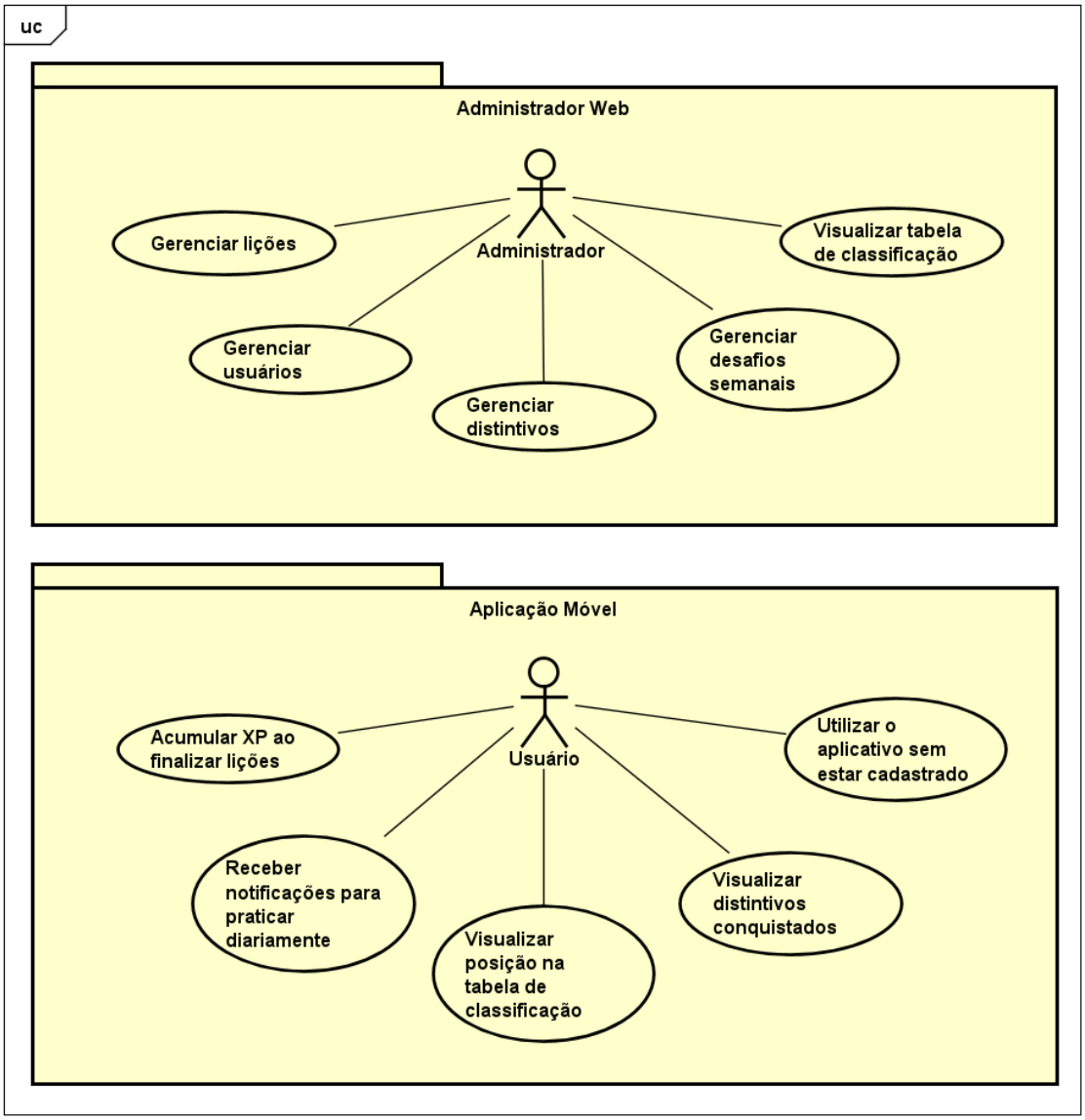

Figura 6 – Diagrama de Caso de Uso Fonte: Autoria própria

4.2.2 Diagrama de Classe

4.2.3 Diagrama de Atividade

4.3 Planejamento Das Lições De Python

Tabela 4 – Hello World

Descrição	Funcionamento da função print()
Exemplo	preencher

Tabela 5 – Variáveis

Descrição	Regras para se nomear uma variável; Mudança de valores
Exemplo	preencher

Tabela 6 – Operadores de expressão

Descrição	Utilizando variáveis simples (X e Y) combinadas com operadores tais como "or", "and", "not", "==", "!="
Exemplo	preencher

Tabela 7 – Tipos de dados - parte 1

Descrição	Utilizando as operações de multiplicação, divisão, adição e subtração enquanto faz-se a introdução às variáveis Integer e Float
Exemplo	Qual o Output para o seguinte comando: print((5 + 3) / 2)

Tabela 8 – Tipos de dados - parte 2

Descrição	Introdução aos tipos String e Boolean e como devem ser utilizados
Exemplo	Qual das opções abaixo trata-se de um valor String? Qual o tipo de dado da variável "x"?

Tabela 9 – Exponenciação

Descrição	Explicação sobre como realizar exponenciação
Exemplo	Qual o resultado para o código a seguir? print(9 ** 2)

Tabela 10 – Ferramentas numéricas (import math)

Descrição	Explicação sobre funções como math.sqrt(), pow(), sum(), min() e max()
Exemplo	Qual das funções abaixo realiza a seguinte operação matemática? $\sqrt{49}$ (Questão de múltipla escolha)

Tabela 11 – Conversões entre tipos de dados

Descrição	Conversões de integer para float, float para string
Exemplo	Qual será o Output da conversão a seguir? print(int(1.784)) (Questão de múltipla escolha)

4.4 Considerações Finais

5 Conclusão

Referências

ALESSANDRIA, S. *Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart*. [S.l.]: Packt Publishing, 2021.

BURKE, B. Gamify: How gamification motivates people to do extraordinary things. bibliomotion. *Inc., Apr*, 2014.

CLOW, M. *Learn Google Flutter Fast: 65 Example Apps*. [S.l.]: Amazon Fulfillment, 2019.

DERUS, S.; ALI, A. M. Difficulties in learning programming: Views of students. In: *1st International Conference on Current Issues in Education (ICCIE 2012)*. [S.l.: s.n.], 2012.

FADEL, L. M. et al. *Gamificação na educação*. [S.l.]: Pimenta Cultural, 2014.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. [S.l.]: Plageder, 2009.

GIL, A. C. *Métodos e técnicas de pesquisa social*. [S.l.]: 6. ed. Ediitora Atlas SA, 2008.

HELPFULDUO. *Gems and Health FAQ*. 2017. Disponível em: <https://forum.duolingo.com/comment/22874501/Gems-and-Health-FAQ#:~:text=You\%20earn\%20Gems\%20by\%20reaching,if\%20you\%20finish\%20new\%20skills).>

LAHTINEN, E. et al. A study of the difficulties of novice programmers. *Acm sigcse bulletin*, ACM New York, NY, USA, v. 37, n. 3, 2005.

LUTZ, M. *Learning python: Powerful object-oriented programming*. [S.l.]: "O'Reilly Media, Inc.", 2013.

MAZUCATO, T. Metodologia da pesquisa e do trabalho científico. *1a. ed. Penápolis: UNEPE*, 2018.

MCGONIGAL, J. *Reality is broken: Why games make us better and how they can change the world*. [S.l.]: Penguin, 2011.

MIOLA, A. *Flutter Complete Reference: Create Beautiful, Fast and Native Apps for Any Device*. Independently Published, 2020. ISBN 9798691939952. Disponível em: <https://books.google.com.br/books?id=y372zQEACAAJ>.

NAPOLI, M. L. *Beginning Flutter: A Hands On Guide To App Development*. [S.l.]: John Wiley & Sons, 2019.

NEWCOMER, E. *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. [S.l.]: Addison-Wesley Professional, 2002.

RICHARDSON, L.; RUBY, S. *RESTful web services*. [S.l.]: "O'Reilly Media, Inc.", 2008.

WERBACH, K.; HUNTER, D. *For the win: How game thinking can revolutionize your business*. [S.l.]: Wharton digital press, 2012.

ZAMMETTI, F. *Flutter na prática: Melhore seu desenvolvimento mobile com o SDK open source mais recente do Google*. Novatec Editora, 2020. ISBN 9788575228234. Disponível em: <https://books.google.com.br/books?id=Pi7KDwAAQBAJ>.

ZICHERMANN, G.; CUNNINGHAM, C. *Gamification by design: Implementing game mechanics in web and mobile apps*. [S.l.]: "O'Reilly Media, Inc.", 2011.

6 Apêncice A Prototipação

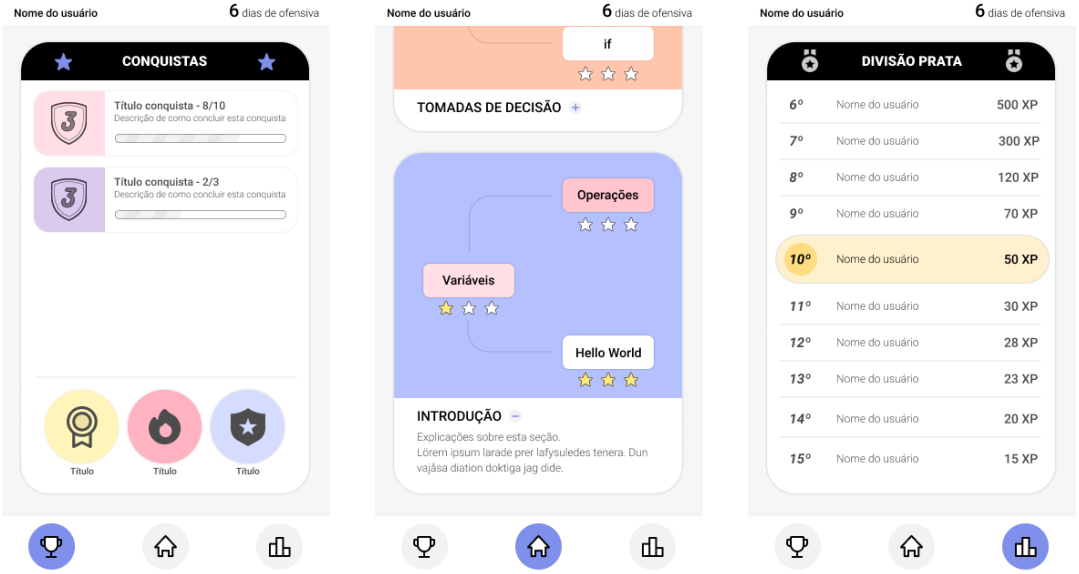

Figura 7 – Prototipação 001

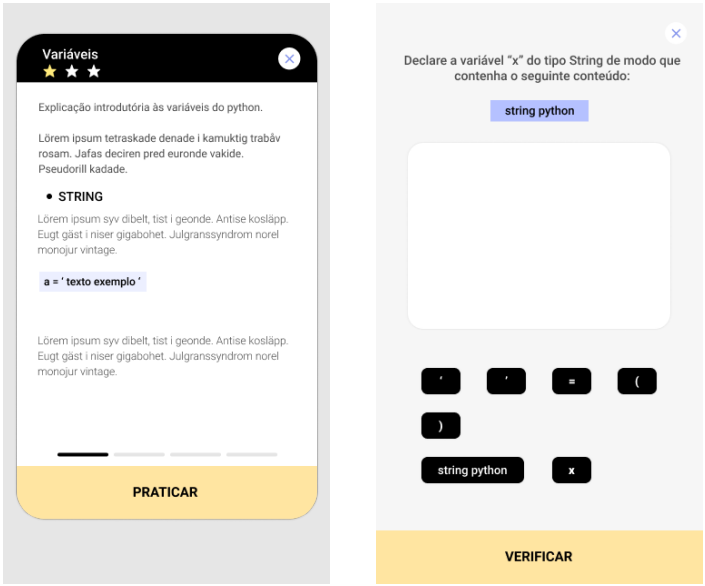

Figura 8 – Prototipação 001