Universidade da Beira Interior
Departamento de Informática

Nº 10 - 2020: *LEGIONofLENDAS: Um Jogo de Avatars*

Elaborado por:

João Freire Nº 41330
Luís Espírito Santo Nº 41400
Pedro Lopes Nº 41237
Vinícius Andrade Nº 42065

Orientador:

Professor Doutor Pedro Inácio

14 de Dezembro de 2020

Conteúdo

Lista de Figuras

Acrónimos

IDE *Integrated Development Environment*

IP *Internet Protocol*

RF Requisitos Funcionais

RNF Requisitos Não Funcionais

SO Sistema Operativo

Capítulo 1

Introdução

1.1 Enquadramento

Este projecto, foi desenvolvido no âmbito da unidade curricular de Programação de Dispositivos Móveis. O projeto que nos foi atribuído foi o "LEGIONofLENDAS: Um Jogo de Avatars"que consiste num jogo onde acontecem combates entre Avatares. Avatares estes que à medida que mais pontos de experiência vão obtendo, o seu nível no jogo vai subindo, e as suas habilidades podem ser melhoradas. Os combates são feitos entre dois utilizadores, ou entre um utilizador e um bot

1.2 Objetivos

O objetivo deste projeto é o desenvolvimento de um jogo orientado para dispositivos móvies. Este jogo permitirá ao utilizador gerar um Avatar na primeira vez que utilize a App. Este Avatar começará com as habilidades mínimas e o seu nível zero. O objetivo é o utilizador participar em combates com outros Avatares, para que consiga obter mais pontos de experiência permitindo assim o melhoramento do seu avatar e também a subida de nível do mesmo. Cada luta, independente do resultado final, recompensará todos os participantes, beneficiando aquele que vença a luta.

Capítulo 2

Estado da Arte

2.1 Introdução

Neste capitulo é explicado de que forma nos baseamos para construir a nossa aplicação, desde o lado estetico ao modelo implementado pelo servidor.

2.2 Bases de Pesquisa

Numa fase inicial do projeto, foi feita uma analise para perceber como iria ser implementado o sistema de batalhas com base em rodadas alternadas de forma a que fosse possivel criar um ambiente energético apesar de toda a batalha ser calculada em milesimas de segundo.

Para tal fomos basear-nos no jogo *Shakes and Fidget* um *Role-playing-game* baseado por turnos, este jogo já existe desde 2009 no mercado e foi para muitos um ótimo jogo com um modelo adaptavel para dispositivos menores.

2.3 Sistema do jogo e diferenças

2.3.1 Economia e Niveis

Ao contrario do *Shakes and Fidget* que tem um sistema de niveis automático, nos decidimos que iriamos implementar um sistema manual de niveis onde o utilizador decide se deve ou não subir de nível, pelo simples facto de que a experiência que este ganha é a moeda de troca por itens no jogo, o que faz com que haja uma diferenciação em poder ter rum rank mais alto, ou poder ser o mais forte.

2.3.2 Sistema de itens

No nosso jogo em contraste ao *Shakes and Fidget* o nosso avatar pode ter infinitas armas e armaduras diferentes, onde todas contam para a batalha, por um lado isto parece ser problematico, mas não é visto que o sistema de combate esta preparado para fazer calculos que diminuem o o dano das armas.

2.3.3 Sistema de combate

O combate no nosso jogo funciona de forma semelhante a norma dos *Role-playing-games* onde os status e os itens tendem a influenciar o dano de cada utilizador.

No nosso caso temos um sistema peculiar que com base nos *Status* do jogador, o enimigo pode acabar por dar vida ao jogador pelo simples facto que este com as suas defesas é capaz de defender-se do ataque.

Dano e Defesa

Para tal usamos um sistema que calcula o dano com base nas defesas e que depois divide o dano deste em 50 porcento dano fisico e 50 porcento de dano magico, estes pos terem sofrido a redução de dano por base nas defesas.

Recompensas defensivas

Para evitar o incentivo a ter apenas dano o sistema de defesa recompensa o aumento das defesas diminuindo a punição da defesa do jogador.

Ruido no dano

Por fim para que tudo seja mais justo ainda aplicamos um sistema aleatorio que aumenta ou reduz uma percentagem do dano.

2.3.4 Sistema de pesquisa de combate

Lutas contra amigos

Ao contrario da maioria dos *RPG* nos temos um sistema de combate entre pessoas conhecidas, de forma local, ou seja, através de um sistema de *QRcodes*, onde cada jogador tem um identificador único *QRCode* e onde o seu amigo fazer um *Scan* do *QRCode* e estes entraram no modo combate, com uma pequena nuance onde quem partilha o *QRCode* tem de clicar num botão extra para iniciar a batalha do seu lado.

Lutas contra *Players* e *Bots*

Deste lado temos duas situações uma batalha contra um *Bot* que cira status semelhantes ao do jogador, e que recompensa de forma reduzida o jogador. Por outro lado temos a lutra contra outros jogadores onde o sistema tenta encontrar alguem com os status semelhantes ao do jogador para este jogar contra onde as recompensas são maiores, caso o sistema não encontre, este inicia uma batalha contra *bots*.

Lutas desvantajadas

Nesta situação a recompensa de vitória é minuscula visto não ser uma batalha equilibrada, desencentivando ao abuso desta pratica.

2.4 Conclusões

Após uma análise detalhada das diferenças acima descritas, verificámos que existem funcionalidades que o nosso projeto carece, por exemplo o facto de não podermos vender itens, ou o facto de não termos um sistema de guildas.

Neste caso usamos o *Shakes and Fidget* para exemplo modelo, mas existem no mercado outras aplicações com modelos mais diversificados que tambem inplementam sistemas parecidos ao nosso sistema de combate, que teem calculos mais balanceados que os nossos e sistemas de punição bem mais sufisticados

Assim talvez numa futura iteração ser-lo-ia possível ter um sistema mais uniforme para este modelo de jogo.

Capítulo 3

Engenharia de *Software*

3.1 Introdução

Este capítulo descreve as funcionalidades das aplicações desenvolvidas no âmbito deste projeto, de forma a que o utilizador as possa compreender melhor durante a sua utilização.

Neste capítulo é apresentada a descrição dos utilizadores, seguindo-se uma análise de requisitos das aplicações móveis desenvolvidas, o diagrama de caso de uso e o de sequencia. Para finalizar, é apresentado o esquema de armazenamento de dados.

3.2 Utilizadores

Neste projeto existe apenas 1 utilizador o cliente, o servidor é autonomo e está perparado para reagir a qualquer evento emitido pelo cliente, o cliente desencadeia sempre a ação e nunca o contrário.

3.3 Análise de Requisitos

Nesta secção, é feito um estudo sobre as necessidades de cada utilizador, para que possa ser feito o planeamento do projeto. Esta secção é essencial para entender o sucesso ou o fracasso do projeto. Nos próximos subcapítulos serão abordados dois tipos de requisitos, os funcionais e os não funcionais.

3.3.1 Requisitos Funcionais

Os Requisitos Funcionais (RF) descrevem aquilo que o sistema executa ou que é esperado executar. O seu objetivo é descrever as funcionalidades disponibilizadas pela aplicação ao respetivo utilizador, detalhando a interação entre o utilizador e a aplicação.

Os RF da aplicação cliente são:

- RF1: a aplicação deve permitir ao utilizador submeter um login e uma password, onde a segunda é armazenada de forma encriptada;
- RF1.1: no caso de o login ser aceite, a aplicação irá redirecionar o utilizador para a a tela principal do jogo;
- RF:1.2: no caso de o login não ser aceite, a aplicação irá ficar na mesma tela e mostrará uma mensagem de erro ao utilizador a avisar o insucesso;
- RF2: a aplicação permite ao utilizador lutar contra diferentes tipos de ambiente, amigos pessoas aleatorias, ou *bots*;
- RF2.1: a aplicação pede ao utlizador que caso queria jogar com amigos, para partilhar o qrcode pessoal ou ler um qrcode preparado no outro dispositivo;
- RF3: a aplicação tem que ter uma página com botões diferentes que redirecionem o utilizador para cada uma das seguintes telas: login, luta, inventário, loja, ranking e melhoramento de habilidades;
- RF4: a aplicação permite ao utilizador que este possa escolher quando quer subir de nivel;
- RF5: a aplicação permite ao utilizador, que este use o seus pontos de experiência para comprar itens;
- RF5.1: a aplicação permite ao utilizador ver os itens que este adquiriu.

3.3.2 Requisitos Não Funcionais

Os Requisitos Não Funcionais (RNF) descrevem quais são as restrições do sistema, assim como as suas capacidades globais e as suas propriedades. Abrangem medidas de desempenho, segurança, tempos de resposta e volume de dados.

Os RNF da aplicação cliente são:

- RNF1: a aplicação requer o acesso à *internet*;

- RNF2: a aplicação requer o acesso à câmara;
- RNF3: a aplicação requer o acesso ao sistema de vibrações;
- RNF4: a aplicação necessita de x *megabytes* para a respetiva instalação;
- RNF5: o Sistema Operativo (SO) móvel compatível com a aplicação móvel é o *Android*;
- RNF6: o *smartphone*, para poder executar a aplicação, terá de ter instalado, no mínimo, a versão *Android* 7.0 *Nougat* do SO;
- RNF7: a aplicação está otimizada para *smartphones*;
- RNF8: a aplicação só permite uma luta por vêz visto que requer a execuções de animações.

3.4 Diagrama de Casos de Uso

O objetivo deste diagrama de caso de uso é representar os requisitos da aplicação. Na Figura 3.1 é apresentado o diagrama dos casos de uso do projeto em questão, onde a aplicação cliente faz pedidos ao servidor e este da respostas a tais pedidos demonstrando a aplicação de forma geral.

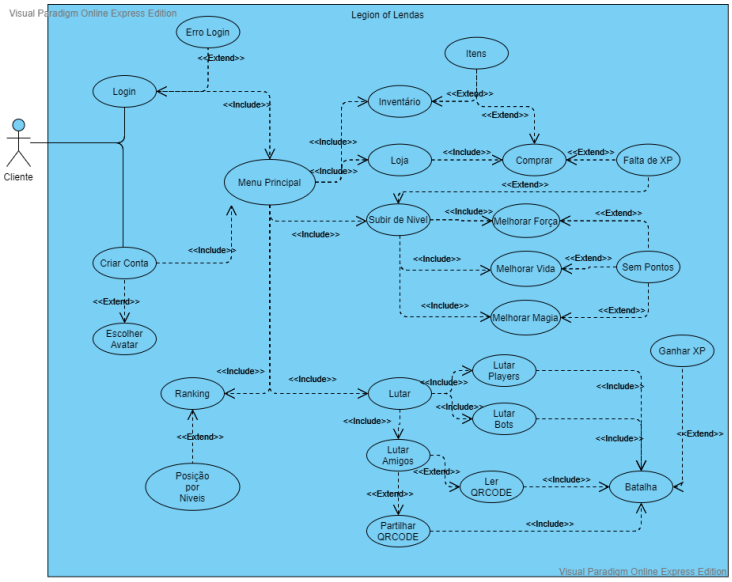

Figura 3.1: Diagrama dos casos de uso do projeto.

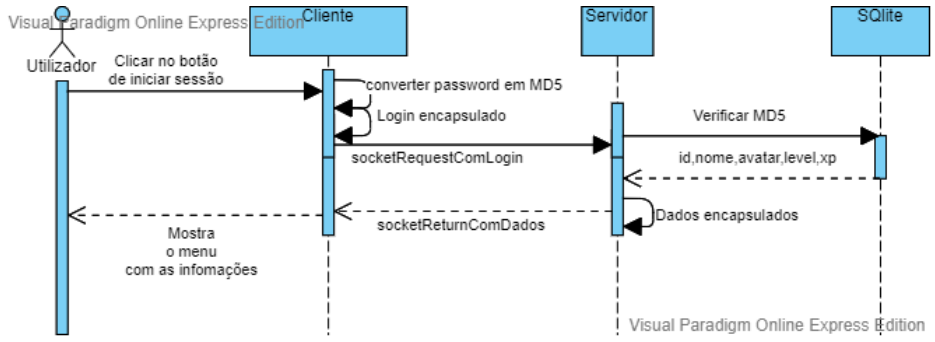

Figura 3.2: Diagrama de sequênci do projeto com base no login.

3.5 Diagrama de Sequencia

O objetivo deste diagrama de sequencia é descrever como a aplicação com apenas a interação do utilizador responde. Na Figura 3.2 é apresentado o diagrama de sequencia que demonstra o funcionamento do sistema de login da app.

3.6 Esquema de Armazenamento de Dados

Neste projeto, foi usada como base de dados a *SQlite*. A mesma é uma base de dados relacional, ou seja, todos os seus dados são armazenados de forma estruturada. Neste caso foram implementados um conjunto de tabelas responsaveis pelo controlo do fluxo do sistema, onde essas tabelas englobam o sistema de utilizadores, itens, status e aquisições de itens, para os avatares.

3.7 Conclusões

O presente capítulo tem uma grande importância, dado que é o que permite compreender a arquitetura e funcionalidades do sistema, e é o que deu a conhecer as implementações efetuadas. Para uma melhor compreensão da descrição feita para cada funcionalidade implementada, foi feita esta apresentação detalhada, recorrendo a figuras.

Capítulo 4

Tecnologias e Ferramentas Utilizadas

4.1 Introdução

Neste capítulo são abordadas as tecnologias usadas na implementação do projeto, assim como as ferramentas que nos auxiliaram na implementação destas aplicações. Ainda é sucintamente abordado o tema da linguagem de programação que foi utilizada.

A aplicação foi desenvolvida para o SO *Android*. Para o efeito, foi utilizado o *Integrated Development Environment* (IDE) *Android Studio* em ambas as aplicações que foram implementadas na sua linguagem nativa *JAVA*. A base de dados remota foi implementada através de um mecanismo estatico no caso o *SQlite*.

4.2 Tecnologias

4.2.1 Linguagem de Programação

Java

Existem algumas linguagens nativas e não nativas para desenvolver aplicações móveis, no entanto privilegiámos a linguagem Java, pelo facto, de ter o suporte dado na unidade curricular de Programação de Dispositivos Móveis. É uma linguagem que é suportada pelo *Android* nativamente e oferece todos os recursos de orientação a objectos. Esta linguagem foi desenvolvida na década de 90 pela empresa *Sun Microsystems* e posteriormente foi adquirida pela *Oracle*. Actualmente, encontra-se na sua versão 15.0.1.

4.2.2 Servidor

Python

Python é uma linguagem de *scripting* multiparadigma desenvolvida na época de 90 com muita pouplaridade que permite aos utilizadores criarem *scripts* pequenos com capacide de diminuir o tempo de desenvolvimento de aplicações ou servidores. Esta neste caso não foi usada no sentido de *scripting*, mas como servidor que recebe do cliente implementado em *java* através de *sockets*, pedidos e este é responsável por filtrar e controlar o fluxo da aplicação e do *SQlite*.

4.2.3 *Android*

O *Android* é um SO baseado em Linux e em outros *softwares* de código aberto. Este é essencialmente desenhado para dispositivos com ecrãs sensíveis ao toque, tais como *smartphones*, *tables*, *smartwatch*, *smart tv*, está neste momento a ter cada vez evidência na indústria automóvel e também nas *Internet of Things*. Quem está por trás deste sistema é a multinacional *Google* que tem desenvolvido o sistema, dando suporte e até criando aplicações para uma experiência mais enriquecedora do SO, como por exemplo o *Google Play*, a loja de aplicações oficial, entre outras aplicações nativas.

Com isto, é possível instalar aplicações desenvolvidas por qualquer programador, quer pela loja oficial, quer por outro tipo de lojas, como por exemplo, o *Apoite* ou o *Kindle*. Em alternativa podemos tirar partido do próprio instalador de pacotes embutido no *Android* que permite instalar aplicações sem utilizar qualquer tipo de loja.

Neste momento é o SO mais usado do mundo, está presente em milhares de dispositivos e apresenta uma tendência de crescimento nas suas diversas áreas da sua implementação.

4.2.4 *SQlite3*

SQlite3 é uma base de dados SQL de leve porto encorporada em todos os aparelhos android com o objetivo de ter uma gestão de dados de forma eficiente e rapida. Esta tem diversas implementações de forma a ser o mais portavel possível.

4.2.5 *Git*

Git, [2] Esta tecnologia é um sistema de controlo de versões de ficheiros que permite desenvolver projetos, nos quais diversas pessoas podem participar e contribuir em simultâneo, editando e criando novos ficheiros sem correr o risco que

alterações possam ser sobrescritas. A grande vantagem é justamente permitir que duas pessoas possam editar o mesmo documento ao mesmo tempo, agilizando o trabalho desenvolvido por equipas.

4.2.6 *GitHub*

O *GitHub* [2] é um serviço *Web* que oferece diversas funcionalidades extras aplicadas no contexto do *git*. É possível utilizá-lo gratuitamente para hospedar os projetos de um modo público ou privado. Isto permite que o projecto interaja com mais colaboradores dentro da comunidade, que poderão contribuir para o desenvolvimento, informando de possíveis *bugs* ou até mesmo enviando código.

4.3 Ferramentas Utilizadas

4.3.1 *Android Studio*

O Android Studio é o IDE oficial para *Android*, desenvolvido pela *IntelliJIDEA* da *JetBrains*. Tem versões disponíveis para os SO *Windows*, *Linux* e *MAC OS*, e é o sucessor que veio substituir o antigo IDE oficial para *Android*, *Eclipse Android Development Tools*. Este IDE foi lançado em 2013 na conferência *GoogleI/O* e a sua primeira versão estável (1.0) foi disponibilizada em dezembro de 2014. Em maio de 2019, a linguagem de programação *Kotlin* substituiu a *Java* como linguagem preferencial para o desenvolvimento em *Android*, ainda que esta segunda continue a ser suportada.

4.3.2 *Sockets*

Sockets[1] são ferramentas que permitem que a comunicação entre outros *Sockets* sobre *TCP*, de forma a evitar a perda de dados mas com a vantagem de ser mais rapido em relação a outras ferramentas, como o caso das *REST-APIs*.

4.3.3 *Outras ferramentas*

Aqui listamos outras ferramentas que nos foram úteis na elaboração deste projeto:

- *Microsoft TEAMS*: É uma aplicação gratuita de voz sobre *Internet Protocol* (IP), onde é possível ter salas de voz que permite a várias pessoas estarem numa conversação, em simultâneo;

- *Adobe Photoshop*: É uma aplicação para edição de imagens;

- *Overleaf*: É um editor de LaTeX*online*, onde foi escrito este relatório e permite que várias contas em simultâneo consigam elaborar o mesmo.
- *Vscode*: É um editor de texto com ferramentas para o auxílio da programação e com *Plugins* que permitem ao utilizador adicionar ferramentas extras as incorporadas com esse editor.
- *PyCharm*: É um dos vários ambientes de desenvolvimento integrado desenvolvidos pela *JetBrains*, o seu foco é a línguagem de scripting *Python* e tem suporte integrado para *Plugins*

4.4 Conclusões

Neste capítulo foram descritas tanto as tecnologias usadas como as ferramentas que auxiliaram no desenvolvimento do projeto. O *Android Studio* e a linguagem de programação *java* foram abordadas no decorrer da unidade curricular, o que foi uma mais valia para o uso destas. *Python* com os *Sockets* e o *SQlite* permitiram com que o fluxo fosse gerido e controlado internamente sem recurso a qualquer outro tipo de *Software*, o *Git/GitHub* serviram exatamente para estarmos sincronizados com o trabalho uns dos outros o que possibilitou uma forte colaboração em equipa.

Capítulo 5

Implementação

5.1 Introdução

No presente capítulo serão detalhadamente abordadas as implementações feitas. Abordaremos a comunicação com a base de dados, a autenticação do *login* do administrador, a verificação do código de acesso ao formulário, as funcionalidades, até ao *design* das interfaces.

5.2 Registo de Utilizador

O registo do utilizador acontece, na parte do cliente, dentro da classe *CharCreationInfo*, aí as informações fornecidas pelo mesmo são verificadas com a base de dados e, se possível, é então criado o utilizador com atributos pré-definidos. As informações escolhidas pelo utilizador reduzem-se apenas ao nome, password e ao avatar. Todas estas informações são então guardadas na base de dados onde estarão disponíveis para futuras operações e acessos.

5.3 Melhoria de habilidades

A consulta de informações por parte do participante é feita através da classe *LevelUp*, que, com base nos pontos de experiência possuidos pelo utilizador, lhe permite melhorar as suas habilidades. Esta função apresenta uma particularidade na nossa aplicação: o *XP*, funcionando como moeda de troca do nosso jogo, é ganho unicamente pela participação do utilizador em batalhas e a evolução do utilizador por vários níveis não é automática, tendo o utilizador que escolher entre esta evolução ou a compra de items na loja (Quando o XP não lhe permite ambas).

5.4 Consulta de Ranking

A representação de uma lista de jogadores constituíndo a nossa versão de um *Ranking* acontece na classe Ranking, aí os utilizadores são organizados de forma descendente consoante o nível associado com a sua conta, mostrando o nível o nome e o icone do utilizador.

5.5 Obter novos items

A obtenção de novos items ocorre através da loja, onde ficam visiveis os *status* do item, o preço, o icone e um butão que permite ao utilizador comprar, por infortunio não foi feito um aviso caso o utilizador tenha sucesso ou insucesso na compra, mas em caso de sucesso está sempre disponivel para consulta do item no inventario.

Atualmente encontram-se disponiveis apenas alguns itens, visto que, a inserção dos itens está a ser feita diretamente na base de dados.

5.6 Consulta de inventário

A consulta do inventário do utilizador acontece na classe *Inventario*, esta classe faz uma chamada à base de dados pedindo informações de todos os itens referentes ao utilizador em questão, recebida uma string como resposta este trata então de a representar ao utilizador de forma organizada, tendo cada uma um ícone e a representação textual dos seus atributos tais como: Força, Magia, Defesa, Defesa Mágica e Vida.

5.7 Luta

A nossa aplicação apresenta ao utilizador a escolha entre 3 diferentes tipos de luta, estas são abordadas na classe MenuLutaOpcoes, que chama a/s respectiva/s actividade/s consoante o tipo de luta.

As lutas *PVP* e contra *BOTS* são abordadas directamente pela actividade parseBatalha depois de haver comunicação com o servidor sobre os elementos da batalha (onde ela efectivamente toma lugar), enquanto que o tipo de luta contra amigos invoca primeiro as classes QrCamera (para o utilizador que vai dar scan ao *QRCode*) e QrcodeStarter (para o utilizador que vai gerar o mesmo) passando depois pela mesma chamada à base-de-danos e, ultimadamente, à classe ParseBatalha.

A classe ParseBatalha não realiza a batalha (embora seja esta a possível percepção do utilizador)

5.8 Base de Dados

A base de dados utilizada no desenvolvimento deste projeto é o *Sqlite3* , como já referido anteriormente, implementamos tabelas como por exemplo a *batalhaLOG* que permite ao sistema explicar quando ocorreu uma batalha nova, ou por exemplo a tabela loja que explica ao cliente que aqueles itens vão estar disponiveis.

5.9 Conclusões

Neste capítulo foram apresentados pormenores importantes sobre o projeto desenvolvido.

Faltou-se referir os problemas que houve na capacidade do servidor aguentar muitas operações visto que do lado do servidor não está tudo distruido por varias threads....

A ligação da aplicação móvel à *Internet* é fundamental para a conexão do jogador ao sistema visto que todas as informações são processadas do lado do servidor.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

Este projecto teve como principal objetivo a criação de um jogo, onde colocamos em prática todo as competencias adquiridas no decorrer da Unidade Curricular.

Experiência com este tipo de aplicações, os elementos do grupo apenas tinham no ótica de utilizador, sendo que a maior adversidade foi, sem dúvida, o planeamento do funcionamento da app. Idealizar a e detalhar todas as funcionalidades do jogo, bem como a parte gráfica (Avatares, etc), necessitou de um esforço não só criativo por parte do grupo mas também de obtenção de informações e modelos de funcionamento de outras apps similares.

Em suma, como a primeira "aventura"do grupo no desenvolvimento de apps deste tipo, a experiência foi bastante satisfatória, buscando sempre uma aplicação simples, intuitiva e funcional.

6.2 Trabalho Futuro

Como dito no ponto anterior, sendo este o primeiro projeto a nível de uma aplicação para dispositivos móveis, existem sim muitas melhorias que poderiam ser implementadas na aplicação. Com mais alguma experiência e conhecimento no desenvolvimento de aplicações para dispositivos móveis, conseguiremos implementar uma aplicação muito mais robusta, com uma parte gráfica ainda mais apelativa, introduzindo animações, mais opções de avatares ou até mesmo conseguir modificar o nosso Avatar, como acontece em muitas aplicações similares disponíveis no mercado. O modo de "luta"também poderá ser melhorado, implementando mais opções e funcionalidades, e por exemplo criar uma espécie de campeonato. Na nossa opinião o jogo está num nível satisfatório, mas deixa em aberto a posibilidade de vir a ser bastante melhorado.

Bibliografia

[1] Python Software Foundation. socket — Low-level networking interface, 2020. [*Online*] https://docs.python.org/3/library/socket.html.

[2] Daniel Schmitz. Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar, 2015. [*Online*] https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/.