Batalha Naval Android
Gestor Tarefas IOS

Trabalho Prático de Arquitetura Móveis

[21250078] – Bruno Coelho [a21250078@isec.pt]

2017/2018

Índice

1. INTRODUÇÃO 3

2. BATALHA NAVAL 4

2.1. MVC (MODEL-VIEW-CONTROLLER) 4
2.2. DIAGRAMAS DE CLASSES 4
2.3. GESTÃO DE ATIVIDADES 8
2.4. GESTÃO DE FICHEIROS 12
2.5. TECNOLOGIAS USADAS 13

3. GESTÃO DE TAREFAS IOS 13

3.1. GESTÃO DE FICHEIROS 13
3.2. RESULTADO FINAL 14

4. CONCLUSÕES 18

5. ANEXOS 19

5.1. BATALHA NAVAL - MANUAL DE UTILIZADOR 20
5.2. POSTER 30

1. Introdução

No âmbito da cadeira de Arquiteturas Móveis foram implementadas duas aplicações móveis de diferentes sistemas operativos, sendo estes: Android e iOS. Para o primeiro foi desenvolvido um jogo, o Batalha Naval com as regras tradicionais e algumas extra. Para o segundo foi implementado um gestor de tarefas que permitia a gestão destas.

2. Batalha Naval

No âmbito da cadeira de Arquiteturas móveis foi desenvolvido o famoso jogo de tabuleiro Batalha Naval. Antes de todo o jogo ter começado a ser implementado foi devidamente organizado e pensado. Esta organização consistiu na escolha de uma arquitetura, acabando esta por ser a arquitetura MVC (*Model – View – Controller*). Como auxílio à arquitetura, antes da implementação foram realizados os diagramas de classes e *mockups*[1], como nos próximos capítulos serão demonstrados.

2.1. MVC (Model-View-Controller)

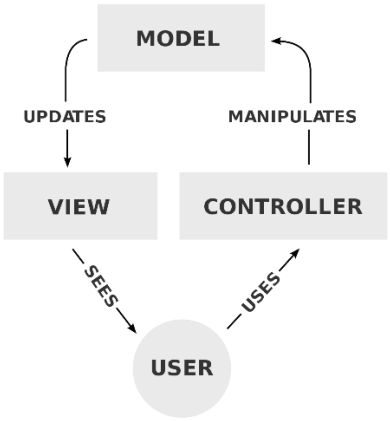


Figura 1 - Funcionamento do MVC

No projeto foi escolhido usar o padrão MVC devido à sua capacidade de divisão do projeto em camadas bem definidas. Cada uma delas, o *Model*, o *Controller* e a *View*, executa o que lhe é definido e nada mais do que isso. Para além do código ficar devidamente mais organizado e nos permitir uma fácil e rápida manutenção é nos permitido facilmente a expansão do projeto ou a mudança da sua *front-end*[2].

2.2. Diagramas de Classes

[1] *Mockup* - Protótipo que fornece parte de uma funcionalidade de um sistema, usado para demonstração e avaliação de design.
[2] *Front-end* - é a prática de converter dados em interface gráfica para que o utilizador visualize e interaja com dados.

Visto que um dos requisitos do jogo era a implementação do multijogador e que todas as mensagens trocadas entre os dois dispositivos deviam respeitar o formato JSON em cada classe que foi usada para comunicação é composta por um atributo designado de “objectType” do tipo de dados String. Neste atributo é armazenado o nome da classe a que pertence, sendo que é definido após a instanciação do respetivo objeto. Por exemplo: quando recebemos um objeto do tipo Position no formato JSON para saber o seu tipo de dados este é verificado através do atributo “objectType”.

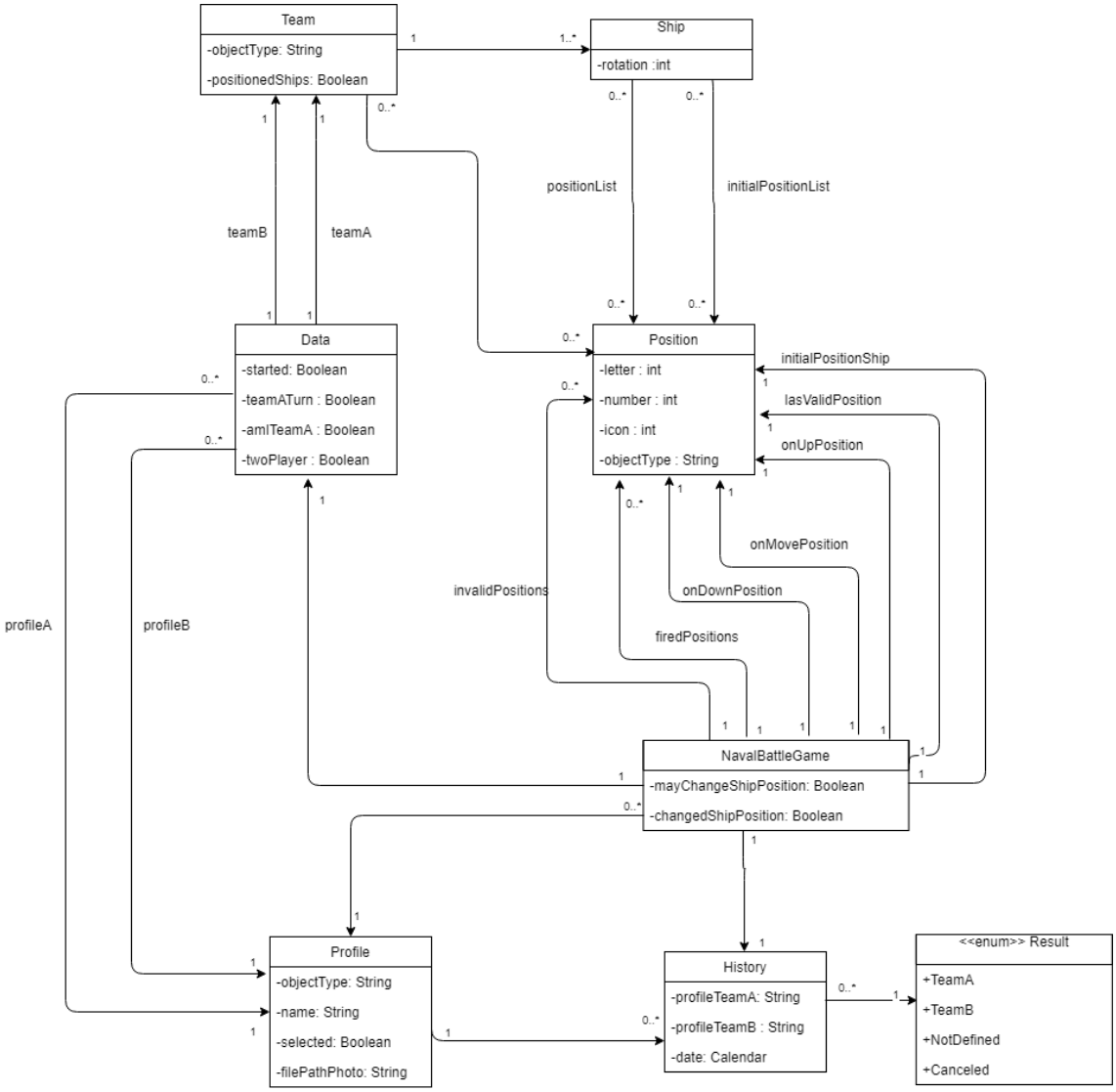


Figura 2 - Diagrama de Classes

Position

Nesta classe é guardada qualquer posição de um barco ou de posições disparadas da equipa. Estas posições são guardadas por dois atributos inteiros (number, letter) de modo a nos facilitar os cálculos, no entanto são apresentados na forma correta.

É também armazenado a cor da célula, podendo esta ser toda preta, ou seja, posição de um barco ou preta com uma cruz, ou seja, posição do barco, mas atingida. Visto que esta classe

é usada para troca de mensagens é composta também pelo atributo objectType, que já foi explicado a sua utilidade.

Ship

Para cada barco será armazenado a respetiva rotação, um conjunto de posições, sendo estas as que o barco ocupa e um conjunto de posições iniciais, apenas para ter os barcos posicionados e o utilizador definir as suas novas posições. Esta classe tem métodos como: getPointPosition(), getAdjacentPositions(), entre outros que serão implementados pelas classes: ShipFive; ShipThree; ShipTwo; ShipOne.

Para cada classe existe um "PointPosition". Este ponto é uma posição central de um barco, que é usado por exemplo para definir a sua posição, ou seja:

Temos um barco com três posições, o seu ponto central é o do meio (identificado com a cruz verde) como mostra na Figura 3. Clicando em qualquer ponto do barco e caso o arrastemos para a posição laranja usamos o método setPointPosition(letter, number) e este, consoante o tipo de barco, automaticamente define todas as suas posições como estão mostradas na Figura 4.

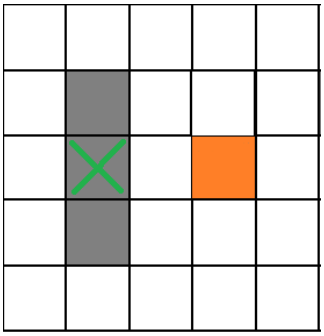


Figura 3

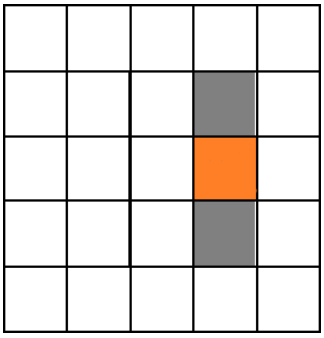


Figura 4

Para cada classe (ShipFive; ShipThree; ShipTwo; ShipOne) contém as seguintes características:

- A classe ShipFive é o barco que contém cinco posições.
- A classe ShipThree é o barco que contém três posições.
- A classe ShipTwo é o barco que contém duas posições.
- A classe ShipOne é o barco que contém uma posição.

O *point position* de cada uma é representado na figura X por uma cruz verde.

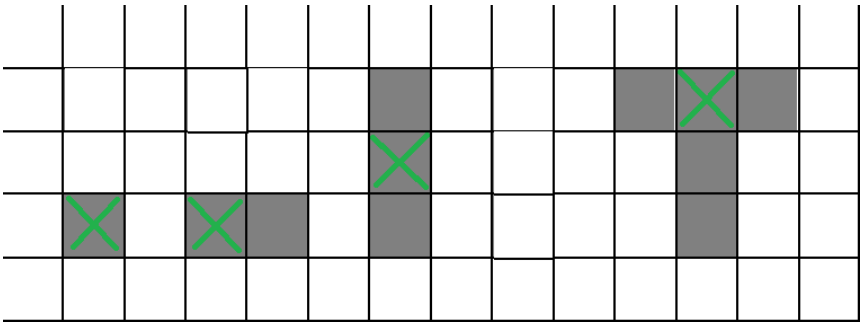


Figura 5

Team

Na classe team é armazenado um conjunto de barcos, um conjunto de posições disparadas e uma variável booleana que indica se a equipa já posicionou os barcos. Esta última variável tem como objetivo indicar no multijogador se a equipa já definiu as posições dos barcos, ou seja: O jogador da equipa A está a posicionar os barcos, caso este tenha recebido uma equipa e esteja indicado que já se encontra com os barcos posicionados é sinal que está à espera do jogador da equipa A.

Message

Foi criada a classe Message com intuito de transmitir comandos de jogo, por exemplo “StartGame”, “Next Turn” entre outros. Não era de todo necessário de uma classe só para estes comandos, mas deste modo facilmente também se pode expandir o jogo, por exemplo ter um chat entre jogadores enquanto estão a jogar.

Data

Esta classe guarda todos os dados relativos ao jogo em que está em acção, como a vez de quem está a jogar, as equipas, os perfis, entre outros.

NavalBattleGame

Sempre que é iniciada uma nova partida é instanciado um novo objeto desta classe. Esta classe é composta por todos os métodos que controlam o jogo.

2.3. Gestão de Atividades

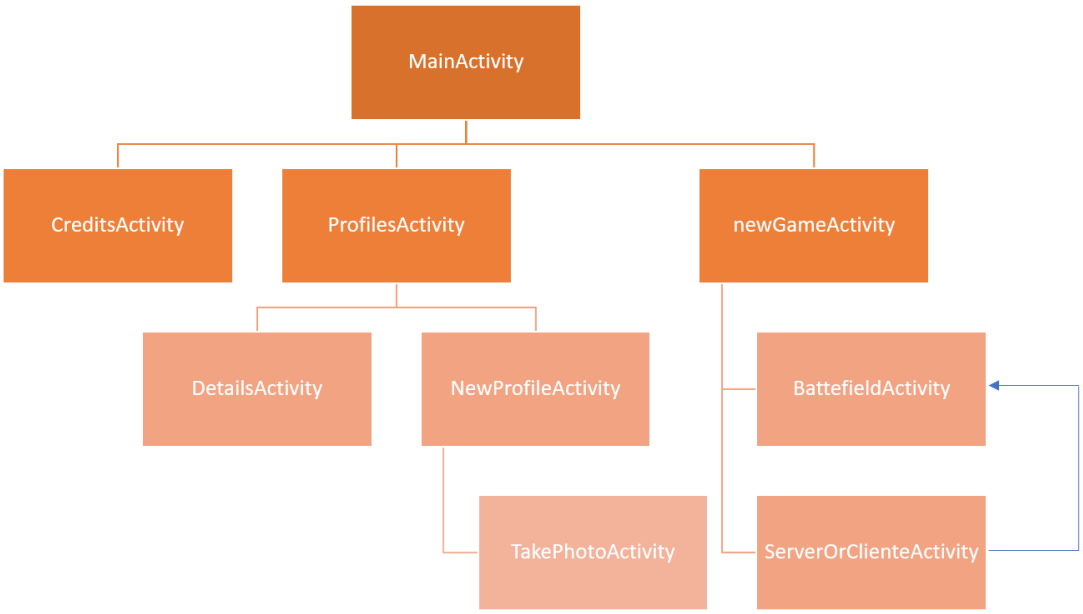


Figura 6 – Organização de Activities

MainActivity

Na MainActivity temos acesso às atividades CreditsActiviy, ProfilesActivity e newGameActivity a partir de dois botões. É a primeira actividade a ser exibida. Caso não esteja nenhum perfil selecionado ou criado ao tentar aceder ao newGameActivity este automaticamente é direcionado para a ProfilesActivity.

CreditsActivity

Nesta atividade apenas foi desenvolvida com intuito de mostrar informação relativa aos créditos do jogo.

ProfilesActivity

Na ProfilesActivity é apresentada uma lista de todos os perfis existentes, mostrando uma foto caso esta exista e o seu nome. Toda a respetiva informação desta lista é lida de um ficheiro com uma lista de perfis serializados. Após a finalização desta atividade a o ficheiro é devidamente atualizado.

Caso seja clicado no ícone dos detalhes de perfil este é direcionado para a DetailsActivity. Caso seja clicado num botão do menu para adicionar um perfil este é direcionado para a NewProfileActivity. Por fim, caso seja clicado no ícone de eliminar é exibida uma janela de confirmação, caso seja confirmado o perfil é eliminado.

DetailsActivity

Nesta atividade é apresentado o perfil com pormenor, mostrando a foto do perfil, o nome e todo o seu histórico de jogos. Caso seja clicado num histórico é apresentado com pormenor os dados do jogo, como o dia, hora, os jogadores e quem ganhou.

NewProfileActivity

Nesta atividade é pedido o nome que o utilizador deseja para o perfil e caso este deseja tirar uma foto basta clicar no respetivo botão para tirar foto.

TakePhotoActivity

Atividade que contém apenas código para o utilizador poder tirar uma fotografia e armazenar no telemóvel, para futuramente poder ser usada.

NewGameActivity

Atividade que contém apenas dois botões que o Utilizador pode selecionar consoante se deseja iniciar um jogo contra o telemóvel ou online. Caso este queira jogar apenas com o telemóvel é direcionado para a atividade BattlefieldActivity. Caso contrário é direcionado para a ServerOnClientActivity. Caso o jogador queira jogar offline automaticamente é a equipa A.

ServerOnClientActivity

Atividade que apenas contém dois botões para o utilizador poder escolher se deseja ser ele a criar o jogo ou não. O jogador que for a criar o jogo automaticamente é a equipa A.

BattlefieldActivity

Esta atividade é a que contem o jogo em si, ou seja, é onde está implementado todo o jogo online e o controlador do jogo, mais concretamente da área de jogo que é um objeto do tipo de dados BattlefieldView, sendo esta uma extensão da classe View.

BattlefieldView

É nesta classe que é apresentado o tabuleiro do jogo. Quando é instanciada são obtidas as equipas. Caso o seja a vez do computador a jogar é realizada logo a sua jogada.

Nesta *view* inicialmente é desenhado o tabuleiro, a seguir a equipa que na qual é a vez de jogar e por fim as suas posições disparadas. Caso seja o fim do jogo é lançada uma AlertDialog a indicar quem ganhou o jogo.

Para mostrar posições disparadas, barcos, saber onde o utilizador quer disparar, entre outros, foi necessário a realização de alguns cálculos. Por exemplo para desenhar um barco é necessário percorrer as suas posições e para cada uma realizar os seguintes cálculos:

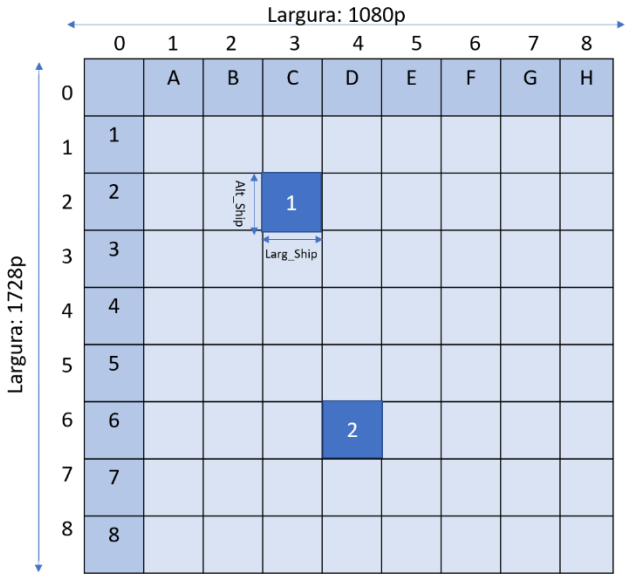


*Figura 7 - Imaginando que estamos a usar um telemóvel com resolução 1720*1080p e que o 1 e o 2 seriam posições de barcos*

1

Imaginando que a posição do barco estava em:
Letra = 3 (C no verdadeiro tabuleiro)
Número = 2 (2 no verdadeiro tabuleiro)
A posição do barco seria desenhada em:
Alt_Ship (altura da posição do barco) = 1728 / 9
Larg_Ship (largura da posição do barco) = 1080 / 9
Letra = 3*(9/Larg_Ship)
Número = 2 * (9/Alt_Ship)

2

Imaginando que a posição do barco estava em:
Letra = 4 (D no verdadeiro tabuleiro)
Número = 6 (6 no verdadeiro tabuleiro)
A posição do barco seria desenhada em:
Alt_Ship (altura da posição do barco) = 1728 / 9
Larg_Ship (largura da posição do barco) = 1080 / 9
Letra = 4*(9/Larg_Ship)
Número = 6 * (9/Alt_Ship)

Nos métodos onDown, onMove, onUp, ou seja, quando o utilizador clica no ecrã é necessário realizar cálculos para sabermos se clicou numa posição de um barco ou não. O seguinte esquema mostra como são realizados:

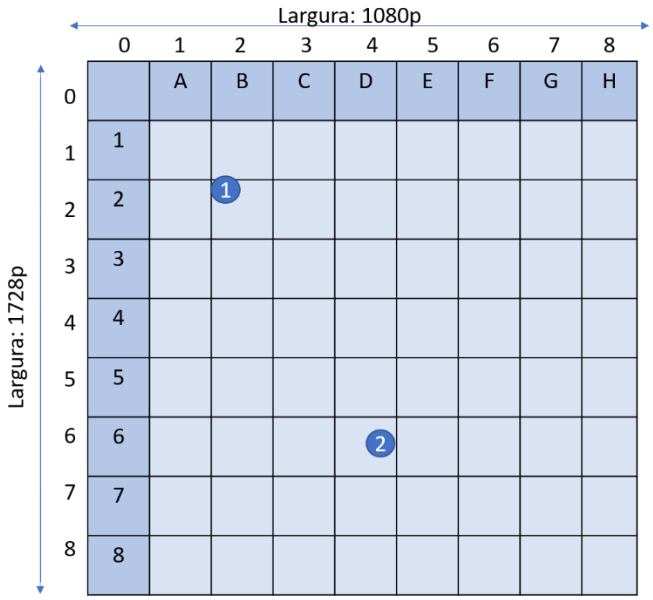
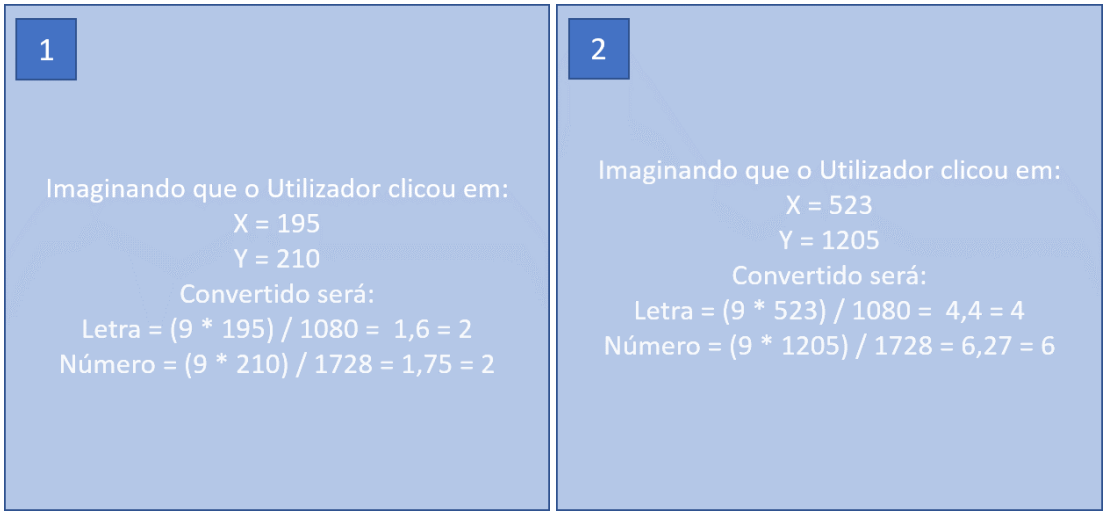


*Figura 8 - Imaginando que estamos a usar um telemóvel com resolução 1720*1080pe que o Utilizador clicou na posição 1 e 2.*



Após o utilizador clicar numa posição do tabuleiro esta é tratada de modo a ter um jogo fluído e completo. No método onDown é apenas verificado se o utilizador clicou em algum barco, caso seja a sua vez de jogar.

Caso seja a sua vez de jogar no método onMove é armazenada a posição onde o utilizador está a apontar com objetivo de atualizar a nova posição do barco e suas as posições válidas e a validações, neste caso, se o Utilizador está a pontar para uma posição dentro da tabela. Finalmente, no método onUp é que são realizadas as operações mais importantes. Caso o utilizador(es) estejam a escolher a posição da equipa é verificado se o utilizador não pretendia realizar uma rotação, se a posição se encontra dentro da tabela de jogo, atualização às posições inválidas entre outras operações. Caso o jogo já tenha começado esta posição é considerada uma das 3 possíveis que são possíveis quando é a vez de jogar de uma equipa.

Caso o utilizador esteja a jogar sozinho, ou seja, contra o telemóvel, as três posições possíveis que cada equipa pode disparar são aleatórias. Concluindo, após estas condições todas é verificado o fim do jogo para cada equipa.

2.4. Gestão de Ficheiros

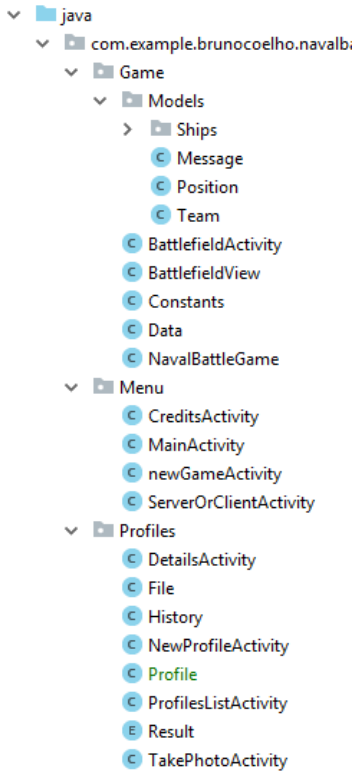


Figura 9 - Pastas e Ficheiros do Jogo

Todos os ficheiros foram devidamente separados devidamente pelas suas funções. Estes foram divididos em 3 pastas, sendo estas: “Game”, “Menu”, “Profiles”.

A pasta Game contém todas as classes, inclusive atividades, que fazem parte do jogo em si. Nesta pasta existe uma chamada “Models”, sendo esta composta por todos os modelos do jogo, como “Position”, “Team”, entre outras visíveis na Figura 9 - Pastas e Ficheiros do Jogo. Como para cada tipo de barco foi criado um conjunto de classes extensíveis de uma, foram juntadas numa só pasta nomeada de “Ships”.

Na pasta “Menu” estão presentes todas as classes que tratam as respetivas atividades do menu. A pasta “Profiles” é constituída por todas as classes que tratam das funcionalidades da gestão de perfis e as suas auxiliares.

Para a implementação do jogo foi necessário da implementação de duas classes auxiliares, sendo estas: Constants, File. A classe Constants é composta por todas variáveis estáticas usadas no jogo, que deste modo facilmente podem ser alteradas. A classe File é composta por todos os métodos usados para a leitura e escrita no ficheiro com a lista de perfis.

Esta gestão de ficheiros e pastas foi divida deste modo de modo a que exista uma fácil manutenção e futuro melhoramento do jogo, caso exista.

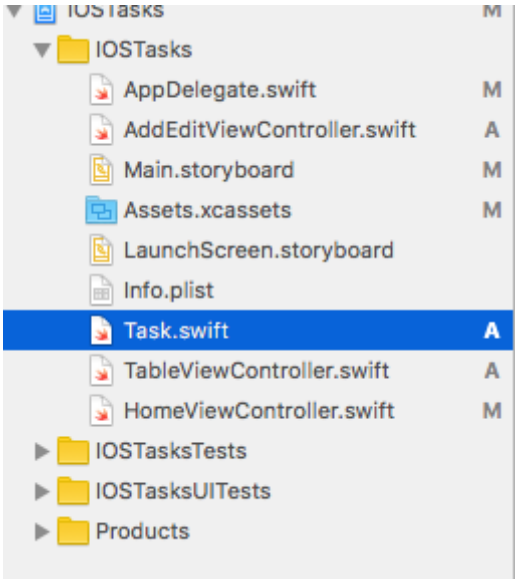
2.5. Tecnologias Usadas

No desenvolvimento deste jogo foram utilizadas duas tecnologias com objetivo de facilitar o seu desenvolvimento. As tecnologias foram:

- **Serialização -** Processo usado para armazenar uma lista de históricos num ficheiro binário. A serialização consiste em converter objetos em Java para um conjunto de bytes.
- **GSON -** Biblioteca Java *open-source* para serializar e desserializar objetos Java para JSON. Esta biblioteca foi usada para converter objetos por exemplo do tipo de dados Position, Team entre outros no formato JSON no modo jogo de 2 jogadores. Foi também usada para converter os objetos de JSON no respetivo objeto para futuramente poder ser usado e tratado.

3. Gestão de Tarefas IOS

3.1. Gestão de Ficheiros



AppDelegate

O ficheiro AppDelegate é composto por métodos de arranque e conclusão da aplicação. Quando a aplicação arranca vai procurar se existe um ficheiro com nome "myFile.dat". Caso exista é sinal que já existem tarefas anteriormente criadas e é feita a sua leitura. Caso a app esteja a fechar é criado o ficheiro "myFile.dat" com lista de todas as tarefas anteriormente preenchida a partir da leitura do ficheiro com alterações ou não. Visto que foi necessário converter datas em strings e vice-versa em vários ficheiros estes métodos foram colocados neste ficheiro de modo a reutilizar código.

addEditViewController

addEditViewController é composto por métodos que tem como objetivo o tratamento de novas ou de tarefas anteriormente criadas. Caso este controlador receba uma task (selctedRow !=-1) é sinal que está a editar uma anteriormente criada, caso contrário é sinal que está a criar uma nova tarefa. É neste controaldor que também tem o evento que faz a devida validação do tamanho da descrição da tarefa.

Task

No ficheiro Task.swift é onde está definida a tarefa. Este ficheiro é composto por todos os atributos que a definem, getters e setters e por fim por métodos de auxilio para a serialização e deserialização do objeto em si, que são usados para guardar e ler informação do ficheiro .dat.

TableViewController

TableViewController é composto por métodos em que tratam toda a informação de apresentação das tarefas.

HomeViewController

HomeViewVontroller é composto por métodos que definem o tipo de filtro que será aplicado nas tarefas, consoante os botões.

3.2. Resultado Final

O trabalho proposto no âmbito da linguagem foi todo implementado. Nos seguintes pontos estão representadas as funcionalidades que considerei mais importantes:

- **Página principal**

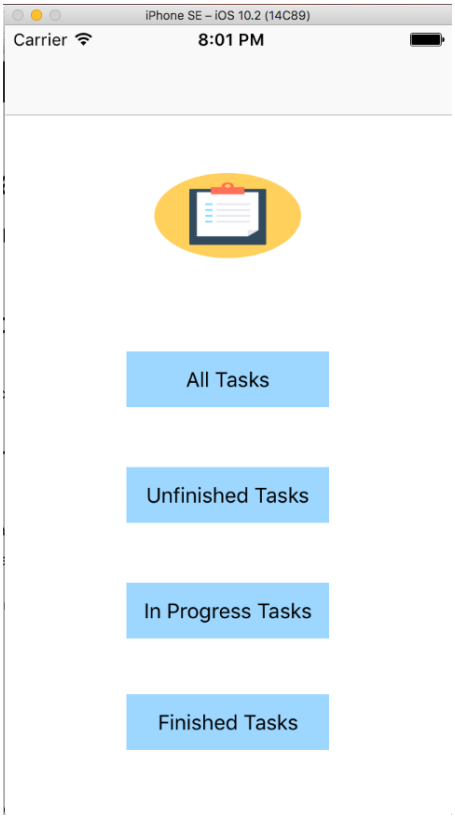


Figura 10 - Ecrã inicial

- **Listagem de Tarefas, neste caso todas estão listadas. Segunda imagem já mostra a reordenação de tarefas.**

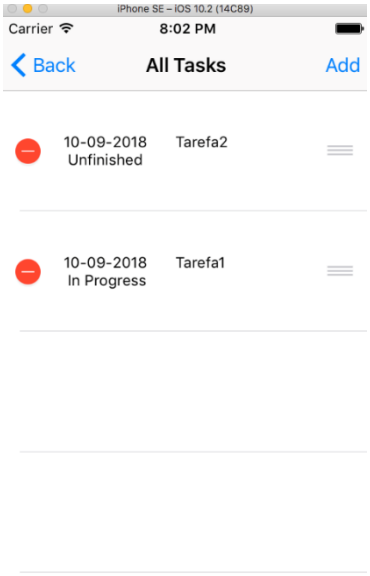


Figura 11 - Listagem de todas as tarefas

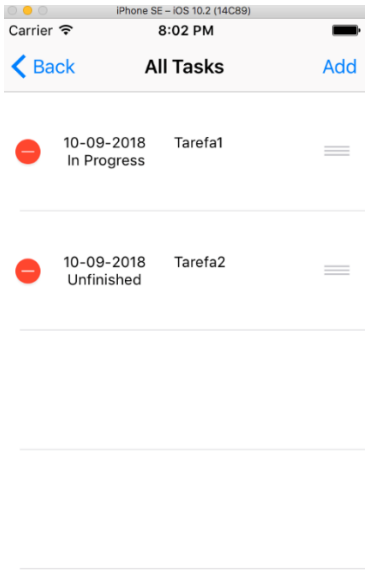


Figura 12 - Exemplo de reordenação de tarefas com a mesma data

- **Adicionar Tarefa e Editar tarefas**

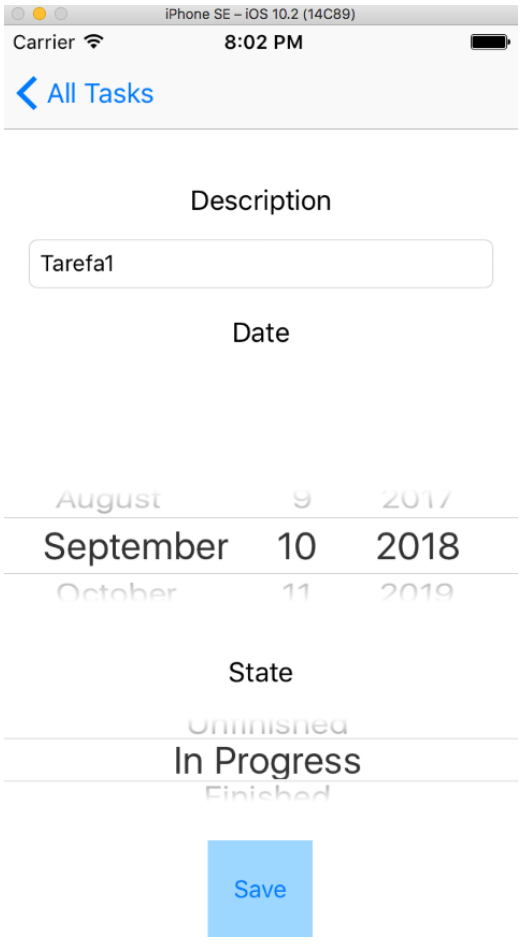


Figura 13 - Edição de tarefas

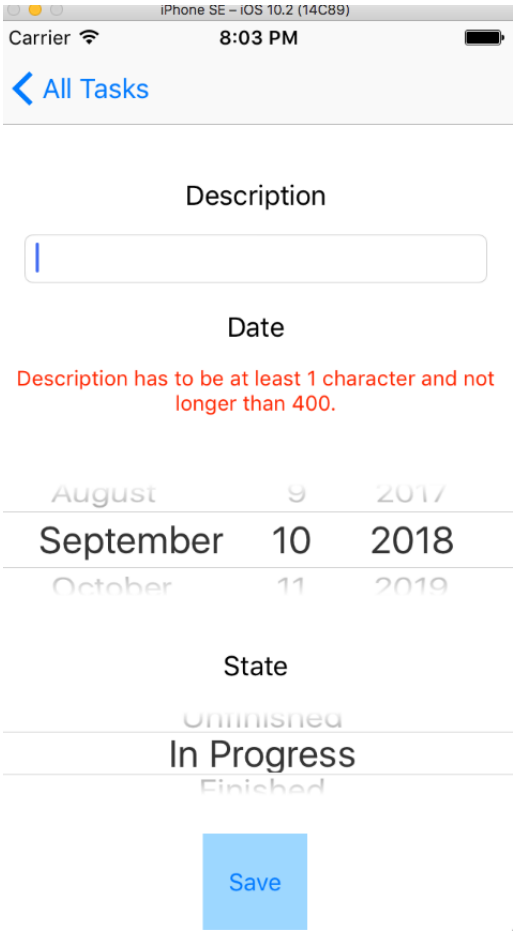


Figura 14 - Inserção de tarefas

- **Remover Tarefas**

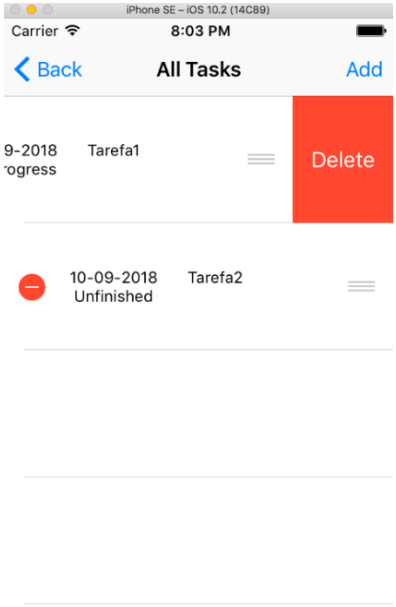


Figura 15 - Remoção de Tarefas

4. Conclusões

Após algumas semanas e dificuldades encontradas e ultrapassadas foi concluído o jogo em Android e o gestor de tarefas em IOS com sucesso. No anexo 5.1 é possível observar um manual do utilizador relativamente ao jogo e no 5.2 o seu poster.

5. Anexos

5.1. Batalha Naval - Manual de Utilizador

Quando iniciamos o jogo deparamos com uma atividade inicial em que é possível começar um jogo novo, selecionar/criar um perfil e ver informações detalhadas da aplicação, mais concretamente os créditos.

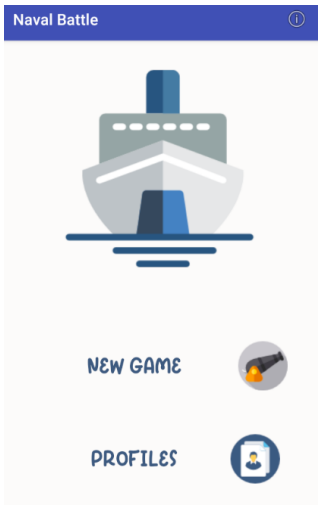


Figura 16 - Painel Principal

Caso queiramos consultar os créditos basta clicar no "i" e somos levados a uma atividade com todos os créditos.

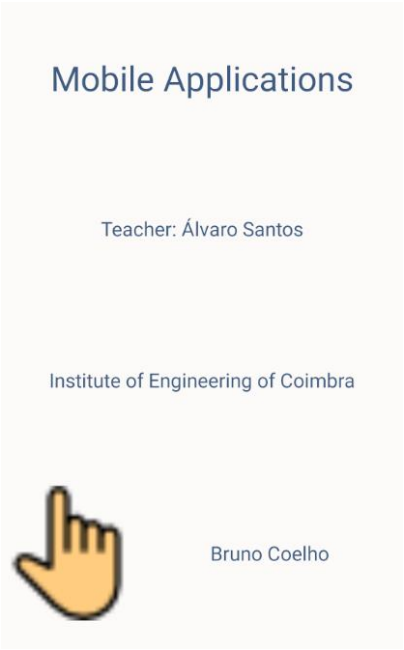


Figura 17- Créditos

Quando acedemos aos perfis é nos apresentada uma lista de perfis existentes sendo que para cada um é possível definir como perfil predefinido, passando a jogar com este nos futuros jogos, é possível ver os seus detalhes, mostrando a fotografia caso exista o seu histórico.

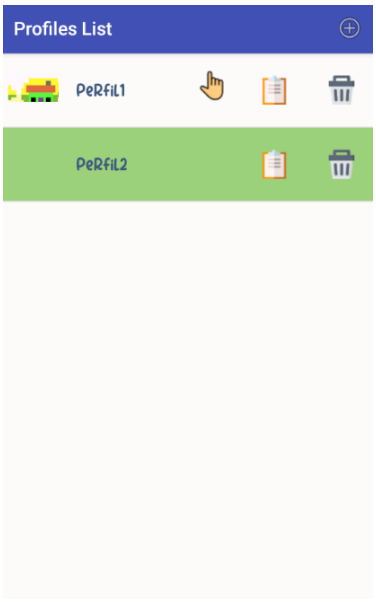


Figura 18 - Listagem de Perfis

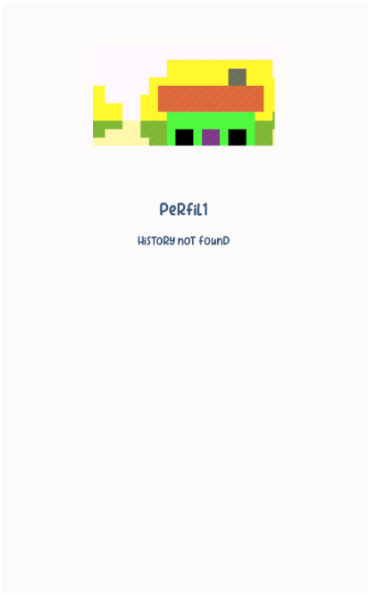


Figura 19 - Histórico sem nenhum jogo.

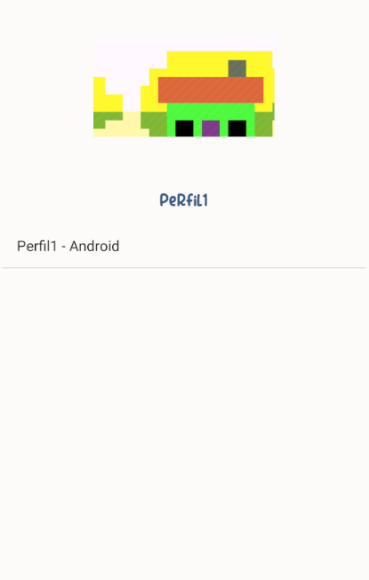


Figura 20 - Histórico com um jogo finalizado

Para cada jogo registado no histórico é possível ver com detalhes quem ganhou, o dia e as horas entre outros dados em que se realizou o jogo.

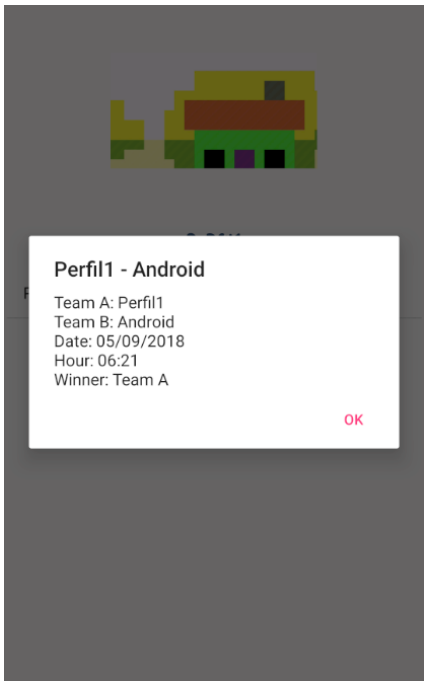


Figura 21 - Detalhes de um Histórico

Por fim, é possível eliminar perfis, que após uma confirmação do Utilizador o perfil é eliminado permanentemente.

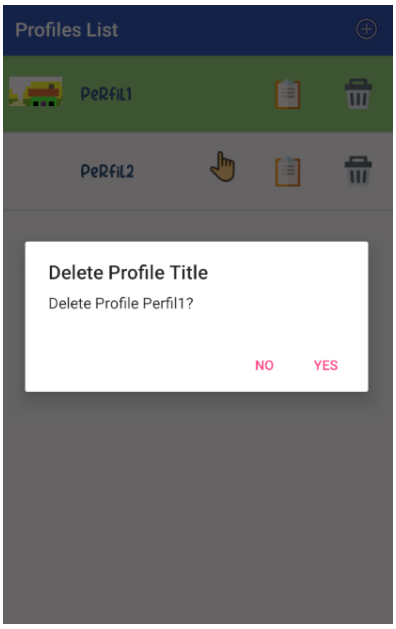


Figura 22 - Confirmação de Eliminação de Perfil

Caso queiramos iniciar um jogo basta clicar no respetivo botão. Caso ainda não tenhamos selecionado um perfil ou criado um perfil o jogo automaticamente transporta o utilizador à listagem de perfis, indicando que é necessário criar/selecionar um perfil. Caso anteriormente já tenhamos criado e selecionado um perfil o sistema automaticamente vai buscar este último perfil usado.

Figura 23 - Inicio de jogo sem perfil selecionado

Após clicar no novo jogo é perguntado ao utilizador se este deseja jogar sozinho ou com outros jogadores. Caso queira jogar sozinho o jogo começa sendo a equipa A, caso contrário é perguntado ao Utilizador se este deseja criar um jogo ou ligar-se a um, sendo que o que cria o jogo é a equipa A.

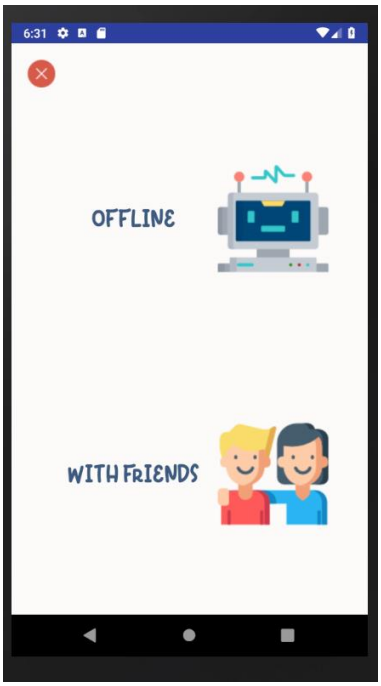


Figura 24 - Menu para selecionar tipo de jogo

Figura 25 - Menu para que tipo de jogador vai ser, caso seja multijogador

Quando o jogo começa são apresentados o nome e a fotografia, caso exista, dos jogadores e a que equipa pertencem. Inicialmente o Utilizador pode escolher a posição dos barcos, que na qual são os elementos a azul marinho. Caso existam posições inválidas, ou seja, algum barco esteja em posições em que não possa estar, estas são apresentadas de cor vermelha. Caso o utilizador clique em cima de um barco este é rodado 90º no sentido do relógio.

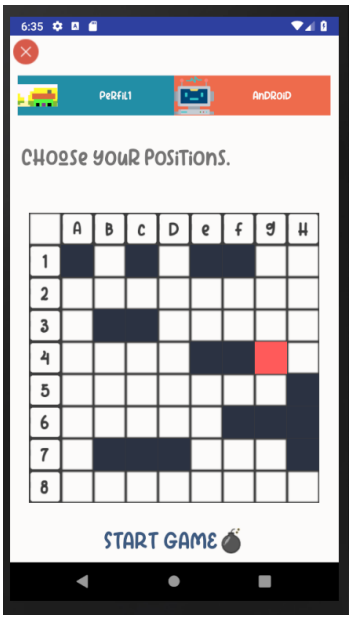


Figura 26 - Mudar Posição dos Barcos

Após o clique botão de começar o jogo, caso existam posições inválidas o Utilizador é notificado. Caso contrário, se for um jogo contra o telemóvel o jogo inicia. Se for contra mais que um jogador e este ainda já tenha clicado para iniciar o jogo o jogo é inicializado, caso contrário aparece um painel a indicar para o Utilizador esperar e quando o outro jogador estiver pronto esta fecha-se automaticamente.

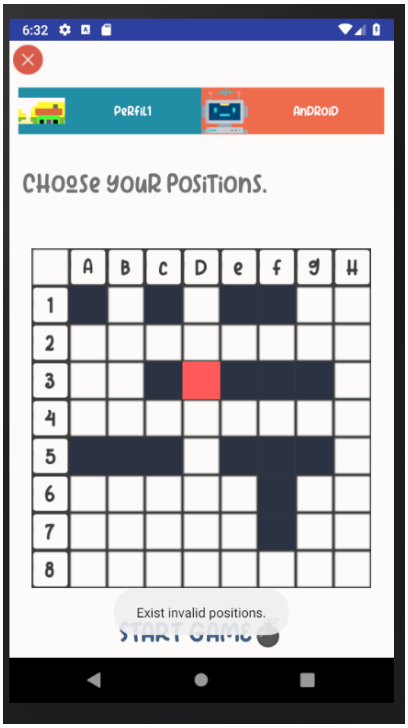


Figura 27 - Início de Jogo com Posições Inválidas

Figura 28 - À espera de outro jogador no multijogador

Após os dois jogadores terem escolhido a posição dos seus barcos a batalha começa sendo que quem começa a jogar é aleatório.

Caso seja a vez do Utilizador jogar e este clicar em posições válidas (não anteriormente disparadas) é apresentado o respetivo ícone de posição disparada. Após três tiros disparados este pode mudar de turno. Só é apresentado ao utilizador se acertou após mudar de turno.

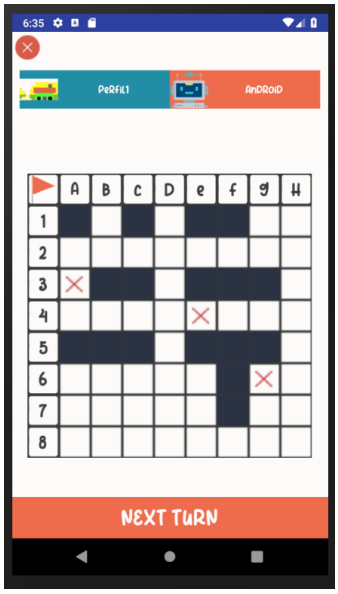


Figura 29 – Posições disparadas pelo adversário

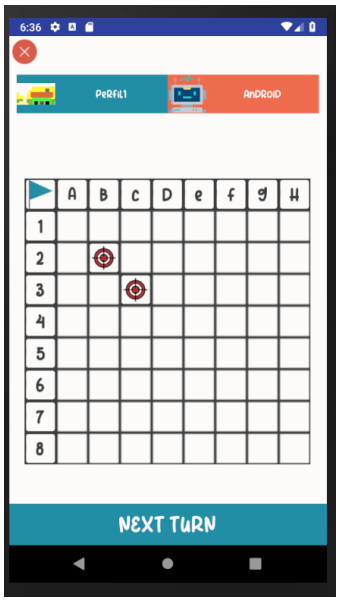


Figura 30 - Posições disparadas

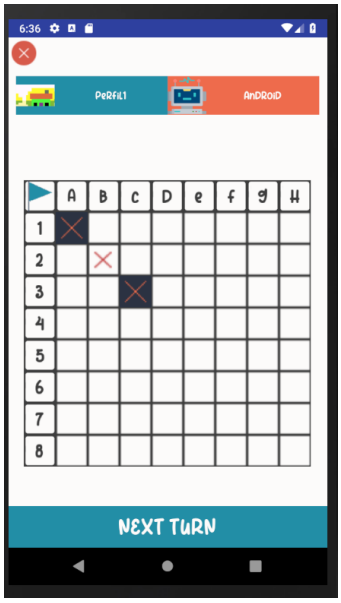


Figura 31 - posições disparadas pelo adversário

Após a mudança de turno caso seja um jogo multijogador são exibidas as posições que o outro jogador esteja a disparar, caso contrário são apresentadas as jogadas que o telemóvel realizou.

Caso alguém acerte em barcos nos três tiros que tenha disparado é lhe permitido alterar a posição de um dos barcos, apenas uma posição e válida.

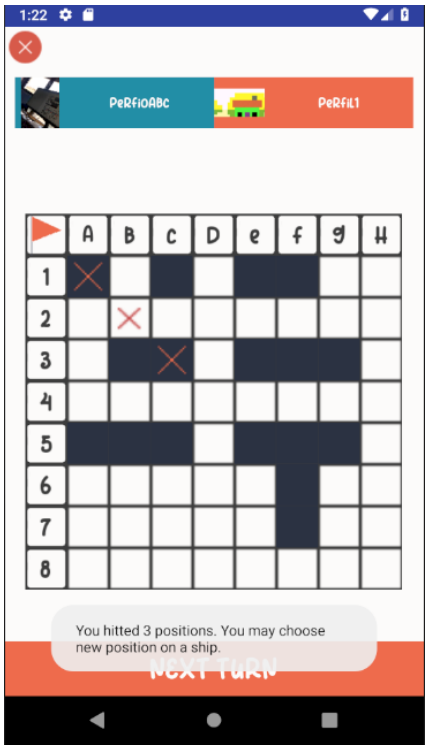


Figura 32 -Após acertar 3 tiros

Caso alguém destrua todos os barcos da equipa adversária é terminado o jogo, mostrando uma janela a indicar o vencedor e um botão para começar um jogo novo.

Caso seja um jogo multijogador e exista algum erro de ligação o jogador começa a jogar contra o telemóvel após ser notificado do acontecimento.

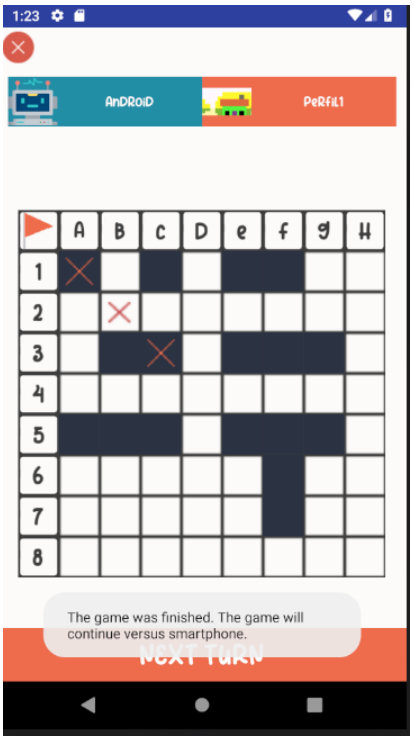
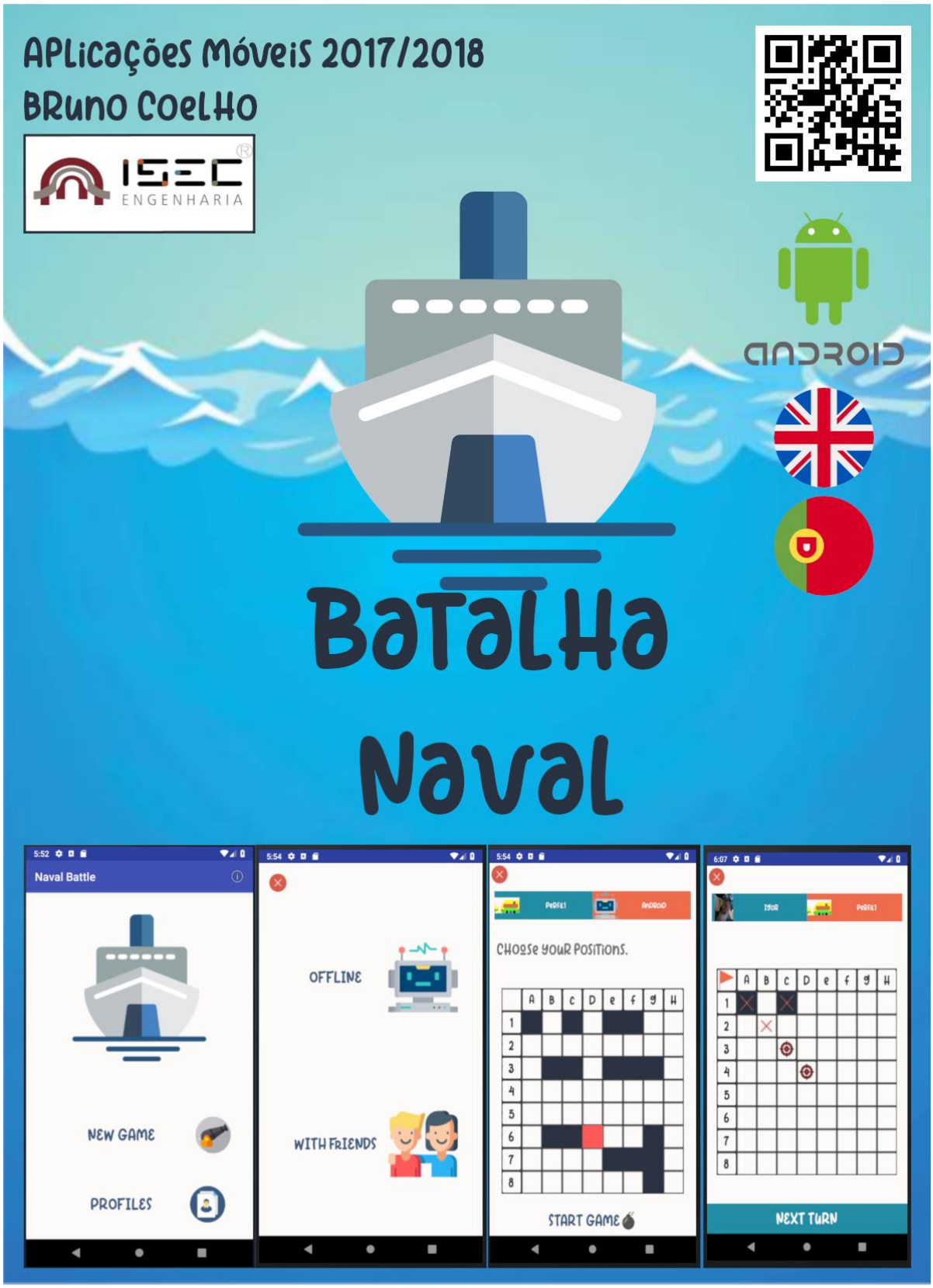


Figura 33 -- Erro de ligação

5.2. Poster


APLicações Móveis 2017/2018
BRuno CoeLHo
ISEC ENGENHARIA
ANDROID
BaTaLHa Naval
Naval Battle
NEW GAME
PROFILES
OFFLINE
WITH FRIENDS
Choose your Positions.
START GAME
NEXT TURN