Laboratório de Desenvolvimento de Software

Trabalho Prático
Grupo 15

Fábio Mendes, 8170157
José Baltar, 8170212
Rodrigo Coelho, 8170282

Índice

Índice de Figuras

1. Introdução

1.1. Apresentação

Para a execução e desenvolvimento deste projeto, incluído no âmbito da Unidade de Curricular "Laboratório de Desenvolvimento de Software" e de tema livre à escolha do grupo, foi escolhido o desenvolvimento de um de um videojogo de estratégia multijogador, do estilo *Tabletop*, no qual quatro jogadores poderão competir entre si até que exista apenas um vencedor.

Neste relatório irão ser explicadas as tecnologias e as decisões de engenharia iniciais que foram tomadas para a resolução do projeto, tendo em conta que se trata da primeira milestone, é a principal componente do trabalho desenvolvido até à data.

1.2. Contextualização

Para o desenvolvimento do projeto, a única tecnologia que será obrigatória de implementação será a utilização da *framework* .NET Core, que será utilizada para o Serviço WEB e para o desenvolvimento da API Rest que pretendemos desenvolver, assim como para com a comunicação com o jogo.

Para o desenvolvimento do frontend, optamos pela utilização da *framework* AngularJS.

Na parte do jogo, iremos utilizar o *game engine* Unity, de livre utilização, de forma a utilizar as suas componentes gráficas. O objetivo da sua utilização é a remoção do foco no desenvolvimento gráfico do jogo, podendo assim deixar mais tempo para o desenvolvimento da componente lógica e crítica do jogo, utilizando C# e comunicando com o servidor através de Sockets.

1.3. Conceitos do jogo

Neste tópico são abordados os principais conceitos que fazem parte da estrutura do jogo.

Começando pelo mapa, este terá sempre uma forma fixa e Hexagonal com três lados maiores e três lados mais pequenos. Três fações (jogadores) localizadas nos lados mais pequenos, com outra fação centrada no mapa. O terreno entre as diversas fações é neutral e poderá ser anexado por estas, até se tornar o centro do confronto.

Existem, portanto, quatro Fações que representam os quatros jogadores de cada partida, sendo estas e as suas características:

- Remnant-> Cor Vermelha, centrada no centro do mapa, Ponto forte: Defesa, Tropas de Infantaria Pesada e Cerco; Condições de Derrota: Derrota Total (Aniquilação da Fação);

- Confederation-> Cor Amarela, Sul do mapa; Ponto forte: Defensiva, Tropas de Míssil e Infantaria Ligeira; Condições de Derrota: Conquista de Centro de Hegemonia ("Capital"), Capitulação ou Perda de 30% sobre Manpower Original
- Royalists-> Cor Azul, Nordeste do mapa; Ponto forte: Ataque, Tropas de Infantaria Pesada e Míssil; Condições de Derrota: Conquista de Centro de Hegemonia ("Capital"), Capitulação ou Perda de 50% sobre Ouro Original;
- Hordes-> Cor Verde, Noroeste do Mapa; Ponto forte: Ataque, Tropas de Cavalaria Míssil e Ligeira; Condições de Derrota: Conquista de Centro de Hegemonia ("Capital"), Capitulação ou Perda de 60% da Moral da População

Em relação aos Recursos disponíveis, existem quatro que podem ser gerados e utilizados: Ouro e Metal (Minas), Madeira (Serraria), Comida (Quintas) e Manpower (Cidades). Os recursos são gerados pelas diversas regiões que o jogador ocupa. Devido a esse facto, certas regiões são especializadas em tipos de recursos diferentes, sendo que apresentam diferentes bónus dependendo do tipo de região. Os recursos são usados unicamente para criar e manter unidades de tropas e exércitos.

A nível diplomático, cada Fação encontra-se, inicialmente, em paz com as restantes. É através de declarações de guerra que a guerra começa, quando as fações criam fronteiras possíveis umas com as outras. Paz pode ser declarada apenas 1 vez entre cada fação. Alianças oficiais entre jogadores não são possíveis.

Existem vários tipos de tropas, todos eles disponíveis a todas as fações, sendo estas:

- Infantaria Pesada
- Infantaria Ligeira
- Míssil
- Cerco
- Cavalaria Pesada
- Cavalaria Ligeira
- Cavalaria Míssil

Cada fação tem, no entanto, bónus diferentes com base nos seus pontos fortes, para cada tipo de unidade, como referido anteriormente no tópico referente às mesmas.

As fações lutam através de Exércitos, compostos por diversos tipos de tropas, até um máximo de 15 unidades. Para se criar um exército é necessário usar os recursos disponíveis, numa taxa fixa. Para se treinar tropas, um exército não pode estar em movimento, ficando estacionado na região em que presentemente se encontra. Dependendo dos tipos de unidades a que lhe são associadas, a sua velocidade de locomoção varia grandemente. Por exemplo,

unidades de cavalaria são capazes de transpor grandes dimensões em menor tempo, ao passo que Infantaria Pesada demoraria mais tempo a conseguir fazer o mesmo percurso. Os exércitos têm uma *manpower* associado a cada unidade. Esse *manpower* é o total de "tropas", ou equivalente a um HP e a Força, simultaneamente. Em combate, este será drenado com base no resultado dos exércitos. Para ser reabastecido, o exército terá de ser “reparado” para recrutar novas tropas para unidades. Cada região apenas pode ter um exército estacionado. Um exército só se poderá mover para uma região diretamente vizinha e sem barreiras ou outros exércitos estacionados. É possível atacar uma região, vizinha, caso tenha exércitos estacionado, combate será iniciado. Caso não tenha, a região é anexada sem qualquer custo.

2. Pesquisa e Arquitetura

2.1. Descrição inicial dos componentes do projeto

Como versão final, a arquitetura assenta num Serviço Web (ASP.NET Core) responsável pelo armazenamento e gestão de toda a informação relativa ao Sistema, desde dados de autenticação e utilizadores até ao histórico de jogos. Comunicam com o serviço tanto os Clientes, descritos mais à frente, como o Servidor Dedicado de jogo.

Então, sendo o jogo um RTS (Real-Time Strategy) multiplayer, deverá existir uma ferramenta para que os jogadores se consigam organizar entre si, ou automaticamente, numa *sala* para dar início ao jogo. Sendo assim, foi decidido o desenvolvimento de um *Game Client* em browser para o efeito. Este *Game Client* será responsável por disponibilizar aos Utilizadores uma interface de acesso aos seus dados pessoais, histórico de jogos, autenticação e ao sistema de *salas* ou *matchmaking*. Um segundo Cliente será a própria interface do jogo, desenvolvida em Unity, que fornece todos os mecanismos de interação gráfica com as mecânicas do jogo, sendo que todo o processamento se irá realizar no próprio servidor.

Finalmente, a comunicação entre os jogos dentro do servidor dedicado será feita através de Sockets. E sendo que uma *sala* em princípio deverá conter um sistema de chat, também serão utilizados Sockets neste contexto. Ou seja, o sistema de *salas* e o servidor dedicado ao jogo estarão contidos num mesmo serviço desenvolvido utilizando Sockets.

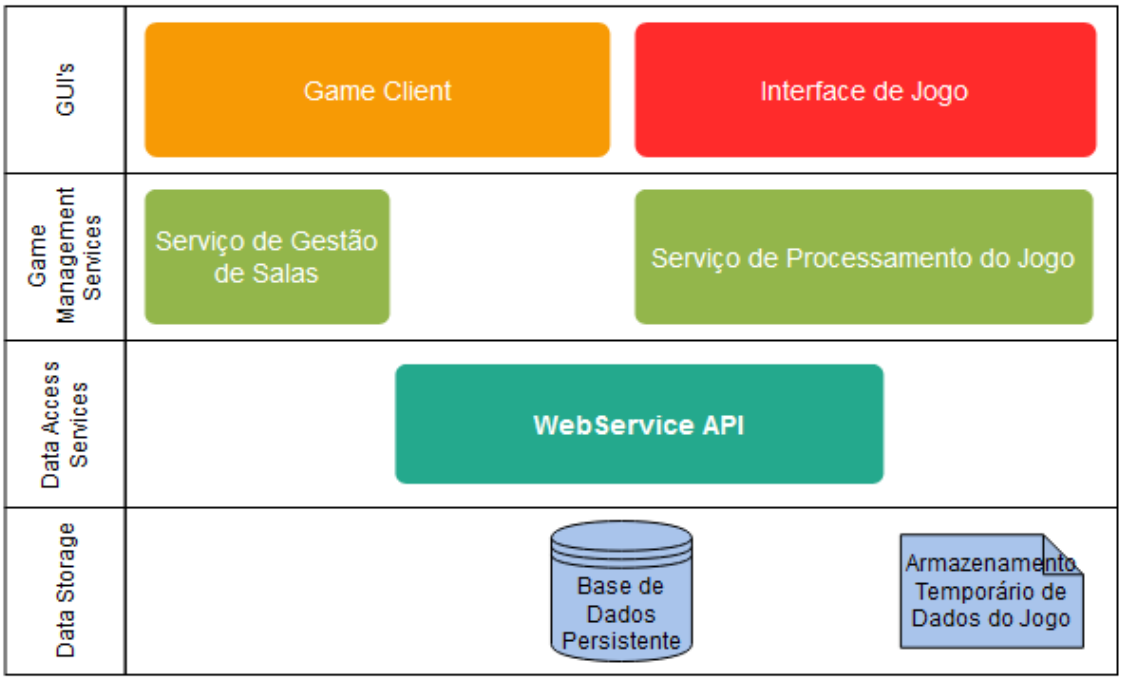


Figura 1 - Arquitetura Conceptual do Projeto

Com tudo isto, são de notar alguns dos *mindsets* seguidos durante a pesquisa, antes de se chegar à versão final:

- Inicialmente a ideia do "Game Client" não existia, ou seja. seria tudo desenvolvido dentro do motor do Unity. Esta seria uma ideia igualmente viável, no entanto decidimos seguir com a versão do browser por ser mais enriquecedor e multiplataforma. Um utilizador poderá assim aceder ao seu histórico de jogos através de qualquer máquina que contenha um Browser, como um smartphone, contudo não poderá entrar numa "sala" para dar início a um jogo.
- Também se pensou em desenvolver o sistema de salas através do serviço HTTP, mas isto impossibilita ou tornaria complicado a adição do sistema de chat dentro das salas, tal como a transição da sala para uma instância de jogo no servidor dedicado.

2.2. Autenticação

Para garantir privacidade e consistência em todo o projeto, existirá um processo de autenticação que permite ao utilizador manter a sessão iniciada entre qualquer componente, desde o Login no browser até ao início do jogo no Desktop.

Para isto, está planeado o uso de tokens JWT para manter uma sessão após o *sign-in*. No fundo, o token será verificado em cada Request realizado ao serviço web, tal como em cada nova conexão com o servidor de gestão de salas e processamento do jogo. Então:

- O serviço Web estará disponível para registo de novos utilizadores e realização de *sign-in*, onde será gerado um JWT token. Cada pedido HTTP após o *sign-in* será autenticado através do JWT token enviado pelo *header* de autenticação.
- Por sua vez, o servidor de gestão de salas e processamento do jogo espera por novas conexões. Quando um cliente se conecta é enviado o JWT token relativo à sua sessão na primeira mensagem. O Servidor então comunica com a API para confimar a autenticidade do token. Sendo válido, o servidor estabelece o protocolo de comunicação com o cliente, caso contrário termina a comunicação.

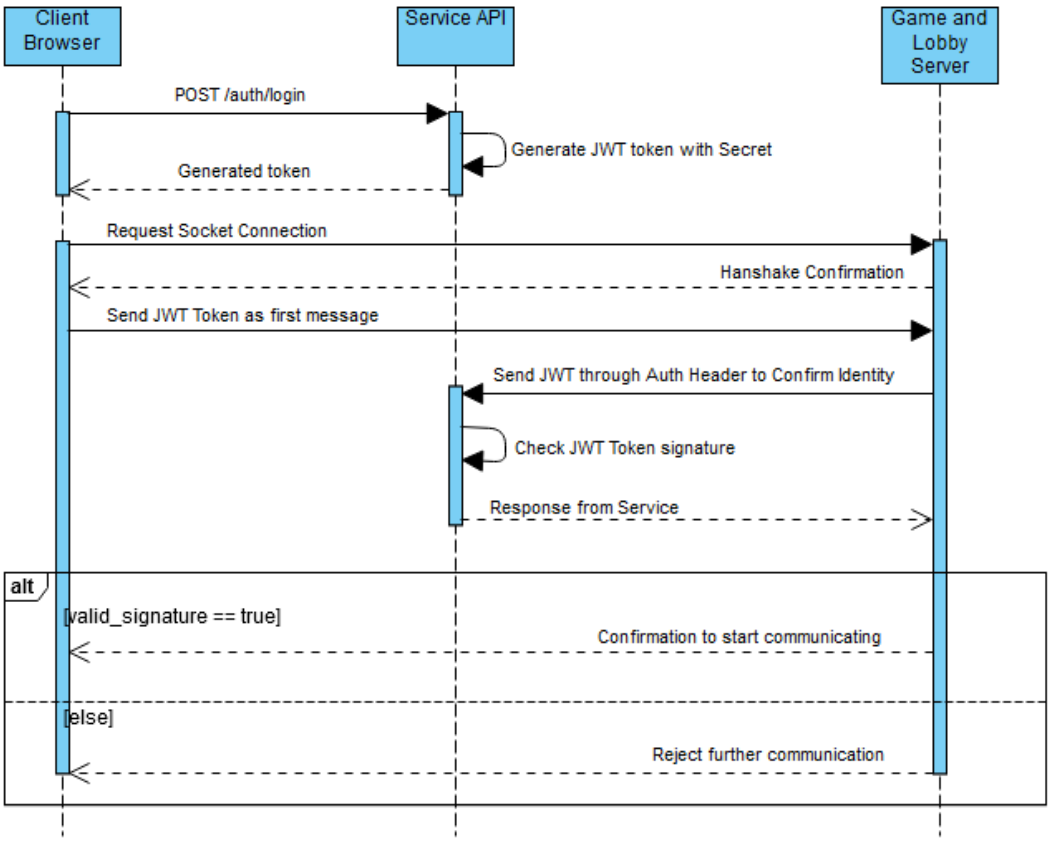


Figura 2 - Arquitetura de comunicação entre Sistemas

É importante notar que, no cliente, a interface de jogo estará completamente separada da interface de interação com as salas. No entanto, sendo que esses dois serviços estão implementados num mesmo servidor (devido à necessidade de partilha de dados), quando um cliente de jogo faz ligação ao servidor já estará identificado pela sala que deu início ao jogo. No entanto, poderá vir a ser necessário identificar a nova conexão de alguma forma adicional e não só através do IP da máquina, por exemplo.

Outras ideias pertinentes pensadas pelo grupo, mas que não conseguiram o *final cut* foram:

Foi considerada a alternativa de fazer a autenticação do cliente na API e nos servidores de jogo no *mesmo local*. No entanto, sendo que a implementação dos dois serviços difere bastante (HTTP requests e TCP/UDP sockets, respetivamente) ficou decidido que deverão estar separados. Conclui-se então que a autenticação é realizada sempre pelo serviço web.

2.3. Tecnologia

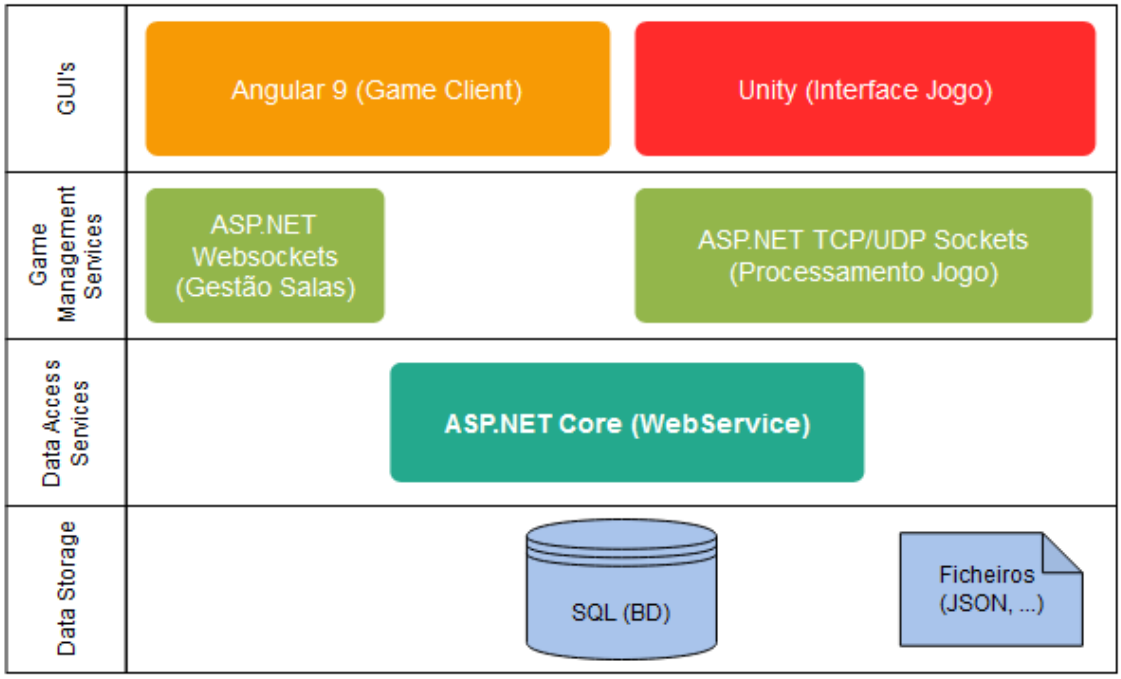


Figura 3 - Tecnologias adotadas para os respetivos componentes

Foi realizado um estudo com o intuito de confirmar que é possível iniciar a Interface de Jogo, desenvolvida em Unity, através do Browser, desenvolvido em Angular.

Resumidamente, será possível executar deste modo se registarmos um costum url protocol no windows. Podemos fazer este registo de forma manual, mas a ideia seria utilizar a aplicação externa "Inno" para gerar o instalador do jogo, nesta aplicação é possível alterar o script de instalação para que faça o registo do protocolo no Windows, no momento da instalação.

Relativamente à implementação nas tecnologias envolvidas é necessário da parte do browser criar um elemento html do tipo href em que o valor seria "(nomeDoProtocolo):(parâmetros)".

```
<a href="TestProtocolParams:GameCode=43foin5723t45cf6243d5bt34876524">Run Desktop App</a>
```

Figura 4 – Exemplo de como ficaria no html

Isto é um exemplo de teste no qual o parâmetro "GameCode" é estático, numa situação real seria um valor dinâmico que dependeria da sessão de jogo à qual se refere.

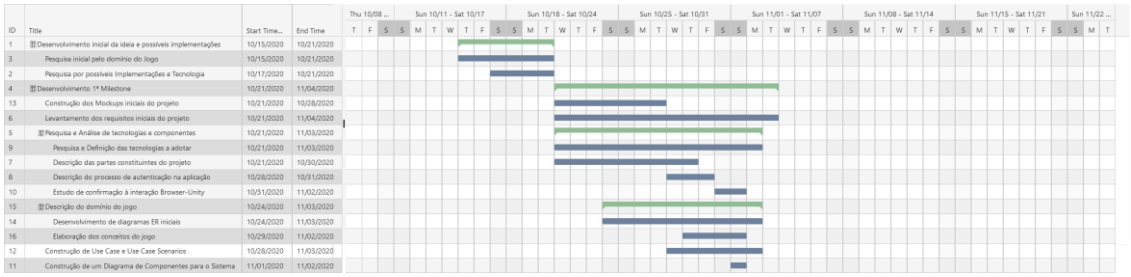
Do lado da aplicação desktop é preciso aceder aos argumentos da linha de comandos do processo atual para obter os parâmetros que o browser envia. Para testar isto foi feito uma aplicação em Unity básica cuja única função é mostrar os argumentos da linha de comandos.

Figura 5 - Este é argumento que a aplicação do Unity recebe enviado a partir da web

Depois de receber este argumento só temos de extrair a informação necessária, neste caso seria o GameCode.

3. Tarefas desenvolvidas

3.1. Planeamento da Primeira Entrega



ID	Title	Start Time	End Time
1	Desenvolvimento inicial da ideia e possíveis implementações	10/15/2020	10/21/2020
3	Pesquisa inicial pelo domínio do Jogo	10/15/2020	10/21/2020
2	Pesquisa por possíveis Implementações e Tecnologia	10/17/2020	10/21/2020
4	Desenvolvimento 1ª Milestone	10/21/2020	11/04/2020
13	Construção dos Mockups iniciais do projeto	10/21/2020	10/28/2020
6	Levantamento dos requisitos iniciais do projeto	10/21/2020	11/04/2020
5	Pesquisa e Análise de tecnologias e componentes	10/21/2020	11/03/2020
9	Pesquisa e Definição das tecnologias a adotar	10/21/2020	11/03/2020
7	Descrição das partes constituintes do projeto	10/21/2020	10/30/2020
8	Descrição do processo de autenticação na aplicação	10/28/2020	10/31/2020
10	Estudo de confirmação à interação Browser-Unity	10/31/2020	11/02/2020
15	Descrição do domínio do jogo	10/24/2020	11/03/2020
14	Desenvolvimento de diagramas ER iniciais	10/24/2020	11/03/2020
16	Elaboração dos conceitos do jogo	10/29/2020	11/02/2020
12	Construção de Use Case e Use Case Scenarios	10/28/2020	11/03/2020
11	Construção de um Diagrama de Componentes para o Sistema	11/01/2020	11/02/2020

Figura 6 - Diagrama planeamento da primeira entrega

3.2. Criação de Diagramas

Foram elaborados alguns diagramas, com o presente propósito de explicar, uma forma simples as diversas componentes do projeto.

Os diagramas ER criados procuram explicar as diversas relações entre as entidades que irão permanentes e estarão guardadas de forma a suportar a lógica de funcionamento dos diversos sistemas e aquelas que descrevem a situação em que estes se encontram naquele momento. Ou seja, são estes os dados em específico que os componentes necessitam de forma a poder funcionar e modela de forma bastante inicial a forma como certas componentes de cada um dos sistemas funcionam.

3.3. Criação de Requisitos e Use-Cases

O principal esforço desta entrega consistiu precisamente na construção de Requisitos funcionais. Os requisitos desenvolvidos consistem no entendimento presente acerca daquilo que planeamos para o jogo e também para o que cada utilizador poderá fazer e consultar. Com isto, os requisitos apresentam a forma como pretendemos que o jogo funcione de uma maneira mais prática, isto é, além de meros conceitos, apresentando um conjunto de características e condições que são de fácil entendimento e possível transcrição para a programação do próprio jogo e também de todas as componentes do cliente, tais como *matchmaking*, entre outros. No entanto, estes requisitos não são necessariamente os finais. Uma vez que estamos a trabalhar utilizando metodologias ágeis, estes servem como *kickstart* para o entendimento geral do projeto, mas são passíveis a mudança e a incremento de novos requisitos, tendo em conta o desenvolvimento do projeto e gestão de mudança que poderá ter de ser efetuada.

Em relação a Use-Cases e a Scenarios, pretende-se mais uma vez mostrar funcionalidades do projeto mas desta vez da perspetiva do utilizador. Ao longo do correr do projeto prevê-se

que os use-cases scenarios sejam alterados principalmente em relação ao passos a percorrer para executar a funcionalidade, visto que nesta fase do projeto ainda não temos uma ideia muito solidificada relativamente à interação do jogador com o jogo.

3.4. Desenvolvimento de *Mockups*

Como parte preliminar do projeto, nesta entrega foram desenvolvidos alguns *mockups*, não muito extensivos, que determinam já a aparência de algumas das funções base do jogo. Servem precisamente para criar uma ideia essencial de como o projeto se deverá apresentar, mas de forma a deixar espaço para desenvolvimentos futuros, isto é, não demasiado finalizados nem demasiado completos, sendo que a cada Sprint, serão incrementados e melhorados para uma forma mais definitiva, o que significa que nesta fase, o essencial foi criar a base para o desenvolvimento de *mockups* futuros.

Para já, foram desenvolvidos *mockups* referentes aos principais menus de uma perspetiva básica, assim como referentes a *matchmaking* e uma potencial base para desenvolvimentos de *mockups* mais maduros para a Game UI.