INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Aplicação para o desenvolvimento de jogos sérios

PEDRO TIAGO DIONÍSIO ABREU
julho de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Aplicação para o desenvolvimento de jogos sérios

Pedro Abreu

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Filipe De Faria Pacheco

Porto, julho de 2019

«*Aos meus amigos e família, por todo o incentivo e apoio para que eu chegasse até esta etapa da minha vida.*»

Resumo

O presente documento relata o processo de investigação e desenvolvimento do projeto "Aplicação para o desenvolvimento de jogos sérios" no contexto da unidade curricular Tese / Dissertação / Estágio do Mestrado em Engenharia Informática do Instituto Superior de Engenharia do Porto.

Neste relatório é documentado todo o processo de desenvolvimento da aplicação, desde da conceção até aos testes finais, estando estas fases repartidas entre quatro secções. As secções são acompanhadas de imagens, tabelas e outro tipo de artefactos de forma a auxiliar o leitor na compreensão do problema. Na primeira secção é realizada uma breve introdução ao tema da tese. O corpo da tese descreve as soluções já existentes e a análise de valor da solução pretendida. A descrição técnica demonstra o processo de desenvolvimento e detalhes técnicos da solução desenvolvida. Por fim, na conclusão, é realizada uma apreciação final no último capítulo deste documento.

Palavras-chave: Desenvolvimento de sistemas gráficos, jogos sérios

Abstract

This document describes the development and research process of the project “Aplicação para o desenvolvimento de jogos sérios” for the class Tese / Dissertação / Estágio, for the conclusion of the master’s degree in software engineering on Instituto Superior de Engenharia do Porto.
This paper contains all the information gathered and produced, from early concepts to final tests. All documentation provided is split between four distinct sections. These sections contain images, tables and other types of content to help the reader understand the problem described. The first section acts as a brief introduction to document. The body of the document describes the state of the art and the value analysis of the developed solution. The third chapter contains all the technical documentation and the development process of the designed solution. Finally, the last chapter contains a global appreciation of all the research and development done.

Keywords: Graphical systems development, Serious games

Agradecimentos

Em primeiro lugar gostava de agradecer particularmente ao professor Filipe De Faria Pacheco pela disponibilidade e feedback obtido durante este projeto.

Agradeço também a todos os meus amigos e família por todo o apoio e incentivo.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

IDE	*Integrated Development Environment*
2D	*Two Dimensional*
3D	*Three Dimensional*
EDSAC	*Electronic delay storage automatic calculator*
IBM	*International Business Machines Corporation*
LINQ	*Language-Integrated Query*
PC	*Personal Computer*
QEF	*Quantitative Evaluation Framework*
RPG	*Role-playing Game*

1 Introdução

O presente documento foi realizado no âmbito da unidade curricular TMDEI, pertencente ao Mestrado em Engenharia Informática do Instituto Superior de Engenharia do Porto. Um dos principais objetivos desta unidade curricular consiste em consolidar e colocar em prática todos os conhecimentos e boas práticas adquiridas ao longo do curso, bem como introduzir aos alunos ao mercado de trabalho. Nas seguintes secções do presente capítulo é apresentado o tema e problema que esta dissertação pretende abordar.

1.1 Contexto

Desde o aparecimento dos primeiros computadores, o ser humano tem desenvolvido sistemas cada vez mais complexos na tentativa de resolver novos problemas ou necessidades existentes. De forma a acompanhar o crescimento e a difusão das tecnologias da informação, o desenvolvimento de sistemas de informação tornou-se uma tarefa cada vez mais complexa uma vez que os requisitos, necessidades e complexidade dos sistemas também evoluiu. (Mens & Eden, 2005)

Os videojogos são um tipo de software muito desenvolvido e utilizado nos dias de hoje, no entanto nem sempre foi assim. Os primeiros videojogos não passavam de provas de conceito, sendo que a componente de entretenimento era apenas um efeito secundário associado. O avanço tecnológico levou a um aumento da diversidade e da complexidade dos videojogos, estando estes, atualmente, bastante integrados no quotidiano das sociedades modernas. (The Strong, s.d.) O sucesso colossal desta forma de entretenimento levou a que muitas outras áreas distintas adotassem valores intrínsecos aos videojogos. De facto, 65% dos adultos que vivem nos Estados Unidos da América são utilizadores deste tipo de sistemas de entretenimento. (Venture Beat, 2019)

Os jogos sérios é uma das áreas que se destacou por adotar muitas características associadas aos videojogos modernos. Os jogos sérios são um tipo de jogos que se destacam dos videojogos convencionais. Apesar de terem uma componente lúdica, os principais objetivos

deste tipo de jogos passam por fatores com sensibilização, ensino, formação, terapia, *marketing* entre outros. (Growth Engineering, s.d.) Todavia, dada a complexidade no desenvolvimento deste tipo de sistemas e o facto que grande parte do mercado ainda permanece na área de entretenimento, este tipo de jogos continua a ser uma pequena percentagem da indústria de desenvolvimento de videojogos.

1.2 Problema

O desenvolvimento de videojogos, como qualquer desenvolvimento software comercial existente atualmente, é um processo longo, complexo e dispendioso, uma vez que envolve várias disciplinas, desde das áreas convencionais da engenharia informática, até a áreas mais criativas como a arte, literatura e música.

O desenvolvimento deste tipo de soluções não é um processo unificado, uma vez que dentro da indústria continuam a existir diferentes filosofias e métodos desenvolvimentos. Uma abordagem que tem ganho bastante sucesso nos últimos anos é o uso de motores de jogos para o desenvolvimento das soluções pretendidas. Os motores de jogos são um tipo de software que empregam várias ferramentas de alto nível para a manipulação de propriedades e entidades gráficas, o que permite acelerar consideravelmente o processo de desenvolvimento de aplicações gráficas, como os videojogos e simulações. As plataformas de desenvolvimento como os motores de jogos, continuam a ser softwares genéricos, que permitem a implementação de vários tipos e formatos de videojogos. Esta abordagem faz com que os motores de jogos sejam um tipo de sistema robusto e versátil. (Unreal Engine, s.d.), (Unity Technologies, 2018)

Em contrapartida, esta característica acaba por ser um dos maiores desafios, dado que o aumento da complexidade envolve a necessidade de formação especializada para a compreensão e bom funcionamento da ferramenta. Existe uma necessidade de uma ferramenta que permita o desenvolvimento de jogos sérios, mas que seja abstraída dos conceitos fundamentais do processo de desenvolvimento de software, de forma a abranger e alargar o maior número de utilizadores possíveis.

Contudo, mesmo com a ubiquidade do uso deste tipo de sistemas para o desenvolvimento de videojogos, o custo de desenvolvimento continua a ser cada vez mais elevado. (Venture Beat, 2018)

Em suma, as ferramentas atuais utilizadas no desenvolvimento de jogos sérios são extremamente complexas, o que dificultam o acesso a utilizadores sem formação na área do desenvolvimento de software.

Figura 1 - Gráfico com custo associado ao desenvolvimento de videojogos em milhões de dólares por ano (Venture Beat, 2018)

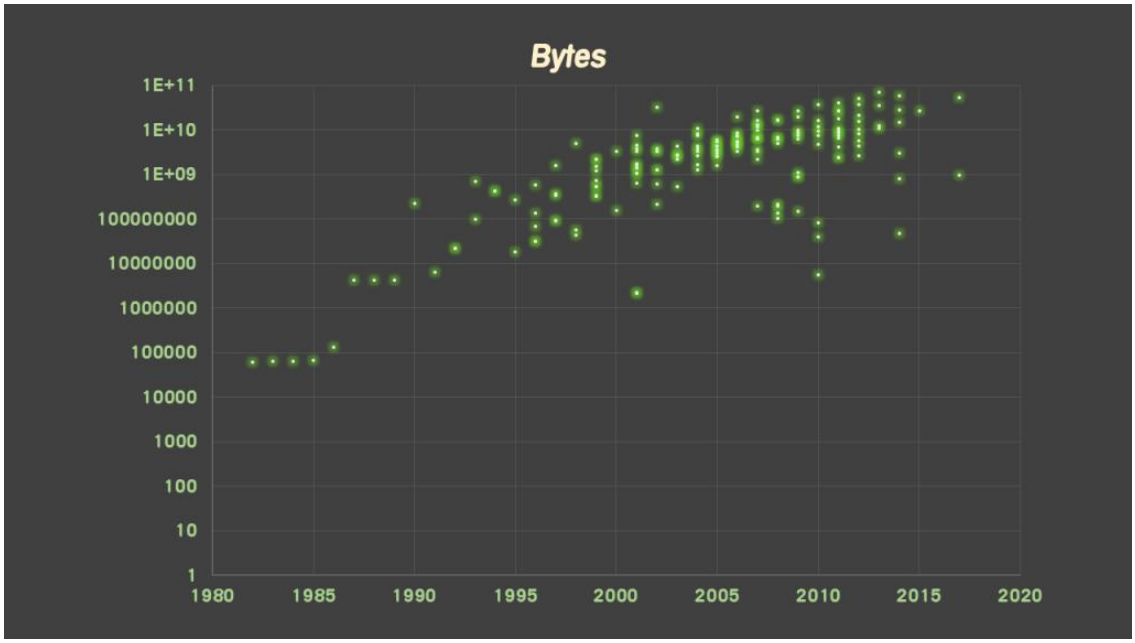

Figura 2 - Gráfico com o tamanho em bytes dos jogos desenvolvidos por ano (Venture Beat, 2018)

1.3 Conceito

O desenvolvimento de jogos sérios continua a ser uma tarefa semelhante ao desenvolvimento de videojogos. O desenvolvimento passa pelo o desenho e implementação de um novo sistema em torno das ferramentas de desenvolvimento de software existentes. Não existem muitas ferramentas fáceis de utilizar que permitam um elevado controlo e molde das mecânicas de jogo de forma a melhor se adaptarem às várias áreas de ensino que se pretendem apoiar. Por outras palavras, o desenvolvimento de jogos sérios deve ser focado em torno do objetivo e não da ferramenta, de modo a que seja mais acessível para um maior número de utilizadores.

Uma ferramenta que permita o desenvolvimento de qualquer tipo de jogo sério sem abstração de código seria a ferramenta ideal. No entanto também seria uma tarefa quase impossível de realizar uma vez que existiria também uma quantidade infinita de soluções possíveis de desenvolver. No entanto é possível criar uma família de tipos de jogos em torno de um conceito ou prática. Este conceito não é completamente novo. Existem algumas ferramentas no mercado que permitem o desenvolvimento de videojogos sem utilizar uma única linha de código por parte do utilizador. (RPG Maker, s.d.). Estas ferramentas, como por exemplo o RPG Maker, serão detalhadas no capítulo seguinte. Independentemente, estas ferramentas continuam a ser muito orientadas para o desenvolvimento de videojogos tradicionais.

O presente projeto pretende seguir um conceito semelhante. O propósito passa pelo desenvolvimento de uma ferramenta que permita o desenvolvimento de jogos sérios, focado num tipo de jogos.

Para este projeto foi escolhido os jogos de cartas para como peça central dos jogos a desenvolver. A escolha do conceito “carta” como entidade principal no desenvolvimento da ferramenta de auxílio para o desenvolvimento de jogos sérios deve-se aos seguintes fatores:

1. Os jogos de cartas sempre foram um tipo de jogos utilizados na vertente educacional, uma vez que estimulam a capacidade de resolver problemas, retenção de informação e criatividade. (Living made easy, 2018)
2. A carta é uma entidade extremamente versátil. É possível associar um conjunto de ideias e valores a um jogo de cartas através do uso de conceitos, imagens, números, símbolos nas cartas de jogo.
3. Popularidade dos jogos de cartas. Os jogos de cartas são um tipo de jogo extramente popular, quer em formato físico quer em formato digital.
4. Escassez de ferramentas de desenvolvimento semelhantes.

Com base nas características apresentadas, o objetivo principal deste projeto passa pela criação de uma **ferramenta de desenvolvimento de jogos sérios de cartas.**

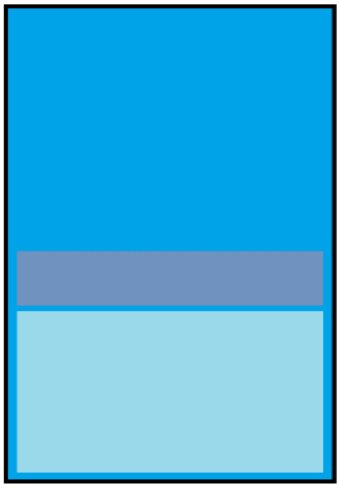

Figura 3 - Exemplo de carta em branco com vários campos editáveis (diferentes cores indicam diferentes zonas ou secções da carta)

1.4 Objetivos

Como foi referido anteriormente, as ferramentas utilizadas no quotidiano para o desenvolvimento de jogos sérios são as mesmas que são utilizadas para o desenvolvimento de videojogos.

De forma a dar resposta ao problema enunciado anteriormente, existe a necessidade de desenvolver uma solução que seja capaz de simplificar e acelerar o processo de desenvolvimento de jogos sérios. A abordagem que se pretende analisar consiste na combinação de vários elementos pertencentes à indústria dos videojogos e dos problemas salientados anteriormente.

A solução passa pelo desenvolvimento de uma ferramenta que auxilie o desenvolvimento de jogos sérios de cartas. A ferramenta deve ser capaz de manipular as características, mecânicas e propriedades associadas aos jogos de cartas, de modo a que seja possível criar jogos sérios de cartas com diferentes objetivos e finalidades. Ao simplificar o uso e customização das mecânicas e características base dos jogos de cartas, é possível criar uma ferramenta modular e apelativa para os utilizadores menos experientes.

As funcionalidades principais da ferramenta passam por manipular as regras, características e propriedades das cartas. Por exemplo, deve ser possível permitir gerir as regras do jogo como retirar cartas de um baralho, jogar as cartas, virar as cartas, retirar as cartas do campo, mas também editar o número de vezes que é possível realizar essa ação. Da mesma forma também deve ser possível editar o conteúdo de uma carta, como por exemplo associar-lhe um nome, um valor, uma descrição, um tipo ou uma imagem.

Estas mecânicas são comuns em diversos jogos de cartas existentes. A manipulação e alteração das características destas ações comuns permite o desenvolvimento de uma ferramenta modular e especializada para a conceção de jogos sérios orientados à volta dos jogos de cartas.

Existem dezenas de outras características e mecânicas, como por exemplo a interação e combinação de cartas. Estes tipos de características estão bastante presentes em versões de modernas de jogos de cartas digitais. Mas uma vez que este tipo de características e mecânicas são muito especificas e pertencem a videojogos de carácter lúdico, não foram consideradas.

Deste modo, salientam-se os principais objetivos de carácter tecnológico e académico que este documento pretende abordar:

- Estudo de arquiteturas de aplicações de sistemas gráficos
- Estudo de uma ferramenta utilizada no desenvolvimento de videojogos.
- Desenvolvimento de uma solução com intuito de acelerar e simplificar o processo de desenvolvimento de jogos sérios.
- Desenvolvimento de jogos sérios.

1.5 Abordagem e sistematização do processo

De forma a organizar o progresso de trabalho e seguir boas praticas de engenharia de software, um dos processos mais corretos a seguir inclui:

1. Pesquisa e análise bibliográfica. A análise focada nos conceitos predominantes ao problema como: Jogos sérios, jogos de cartas, desenvolvimento de jogos, arquitetura de aplicações de sistemas gráficos.
2. Analise de valor da solução pretendida
3. Especificação de requisitos e de casos de uso
4. Análise de domínio e de conceitos de negócio
5. Análise e design da arquitetura do sistema.
6. Implementação e teste do produto.
7. Avaliação e possíveis reiterações do produto.

1.6 Organização do relatório

O presente relatório está dividido em quatro secções distintas: Introdução, Corpo da Tese, Descrição Técnica e Conclusão

No primeiro capítulo, a introdução, é abordado o tema do projeto, bem como os principais e objetivos e problemas que este documento tenta resolver.

O corpo da tese aborda mais sucintamente o contexto da área do projeto. Esta secção contem também uma análise das soluções já existentes bem como uma análise de valor da solução pretendida.

No capítulo descrição técnica é descrito e documentado todo o processo de desenvolvimento e decisões tomadas em relação à solução desenvolvida.

O último capítulo deste documento, a conclusão, contem uma retrospetiva de todo o trabalho realizado tendo em conta os resultados obtidos, objetivos atingidos e limitações existentes.

2 Corpo da Tese

Este capítulo pretende enquadrar o projeto desenvolvido, com uma maior enfâse na análise do estado de soluções e na área de negócio em que este projeto se insere. O conteúdo deste capítulo serve também como um ponto de partida para a compreensão do problema inerente a este documento.

2.1 Contexto

O conceito videojogos têm crescido e evoluído drasticamente desde o aparecimento dos primeiros jogos nos meados do século anterior.

Os primeiros videojogos começaram a ser desenvolvidos na década de 50, em laboratórios, universidades e instalações militares capazes de alojar os primeiros computadores desenvolvidos. Os primeiros jogos consistiam em adaptações de jogos já existentes, como por exemplo: o jogo do galo, desenvolvido em 1952 no EDSAC, Blackjack, em 1954 no IBM-701, Damas, em 1956 no IBM-701, Xadrez, em 1957 no IBM-704. Nesta época, os computadores ocupavam salas inteiras e necessitavam de outro tipo de cuidados e técnicas para ser manuseados, dado que o acesso a estes estava muitas vezes restringido a equipas de investigação e a militares. Estes computadores e jogos não possuíam interfaces gráficas e robustez que os sistemas modernos possuem, não estando assim acessíveis ao público. Desta forma, os jogos eram considerados como provas de conceito ou objetos de avaliação científica. (The Strong, s.d.)

As primeiras consolas de videojogos começaram a surgir nos finais da década de 70, com o aparecimento de consolas como a Atari 2600, possibilitando pela primeira vez o acesso aos videojogos ao utilizador comum. Na década de 90, com os avanços tecnológicos e o crescimento da Internet, os videojogos sofreram uma grande transformação. Começam a surgir os primeiros jogos 3D bem como os primeiros jogos multijogador pela Internet.

Plataformas como PC, dispositivos móveis, consolas portáteis, tornaram-se cada vez mais populares, o que permitiu uma maior abrangência dos videojogos ao público. (The Strong, s.d.)

Figura 4 - Evolução dos gráficos da série popular de videojogos Super Mario (Wolf, 2011)

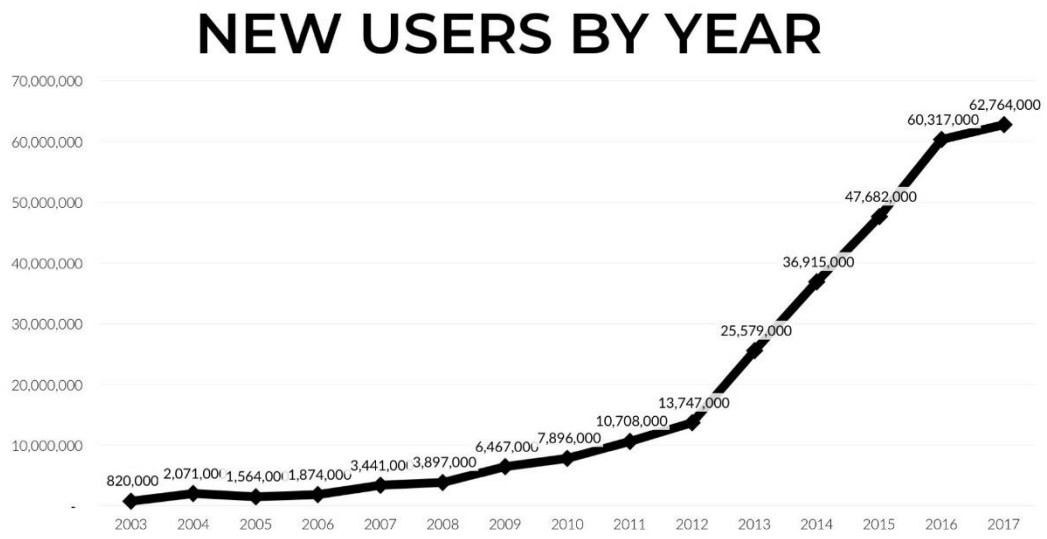

Figura 5 - Número de utilizadores registados na plataforma Steam por ano, uma das maiores plataformas de distribuição de videojogos (Galyonkin, 2018)

Atualmente a indústria dos videojogos vale mais de 136 mil milhões de dólares, sendo este um dos negócios mais lucrativos existentes. (Bloomberg, 2019) Apesar de o grande foco desta indústria ser o desenvolvimento de jogos lúdicos, começam a surgir novas tendências e ideias motivadas pelo o sucesso desta indústria.

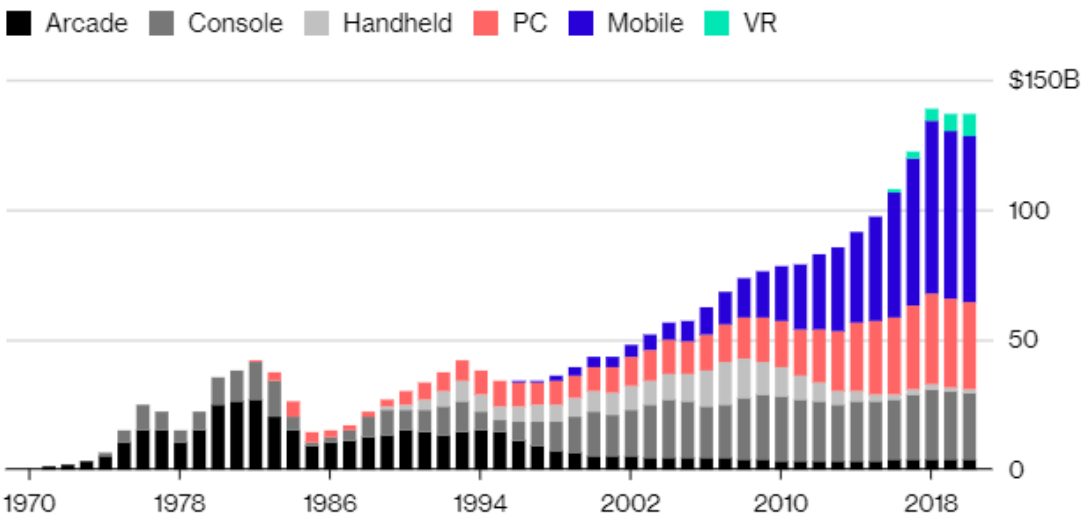

Figura 6 - Gráfico de barras acumulado com total de receitas geradas por parte dos videojogos por ano em milhar de milhão de dólares (Bloomberg, 2019)

Uma destas novas tendências é o uso e aplicação do conceito de gamificação.

A gamificação é uma aplicação direta de um conjunto de elementos e mecânicas normalmente associadas a jogos a ambientes externos. A gamificação tem uma definição generalizada e pouco concreta uma vez que não há consenso entre a definição dada por diferentes autores. Diferentes autores têm diferentes opiniões na distinção entre os elementos físicos e elementos digitais. Por exemplo, Gartner definiu o conceito de gamificação como *the use of game mechanics and experience design to digitally engage and motivate people to achieve their goals*. (Growth Engineering, s.d.)

A aplicação destes elementos normalmente associados a jogos (por exemplo medalhas, tabelas de classificação, sistemas de pontos, narrativas) a outras áreas tem contribuído para o aumento de eficiência, produtividade e lucro em inúmeras empresas.

Um exemplo clássico da aplicação de elementos de gamificação na indústria são os programas de fidelidade. Muitas empresas, desde gigantes tecnológicos até pequenos restaurantes e cafés, oferecem aos seus clientes um sistema de pontos baseado através da aquisição de novos produtos e/ou de partilha com outros potenciais clientes. Estes pontos normalmente podem ser utilizados para a aquisição de certos produtos, descontos, entregas ou outro tipo de prémios associados a essa determinada empresa. Este tipo de programas de fidelidade/recompensa é uma forma comum de manter os clientes interessados, uma vez que para as equipas de vendas e de marketing, é mais fácil manter clientes existentes do que adquirir novos (Technology Advice, 2019)

Outra nova tendência proveniente do crescimento e evolução dos videojogos são os jogos sérios. Os jogos sérios diferenciam-se dos jogos convencionais por serem um tipo de jogo cujo

o objetivo principal não passa pelo entretenimento. Estes tipos de jogos usam a componente de entretenimento inerente aos jogos como uma ferramenta educacional/formativa, meio publicitário ou como forma de consciencializar o público para uma determinada causa, como por exemplo saúde ou aquecimento global. Ao fazerem uso do aspeto apelativo e motivacional associado aos jogos lúdicos, os jogos sérios têm sido uma alternativa interessante pela forma como interagem com o público.

O desenvolvimento de videojogos mudou imenso desde o lançamento dos primeiros jogos nos meados do século passado. Tal como foi referido anteriormente, os jogos dessa época eram desenvolvidos para equipamentos específicos, com hardware considerado bastante rudimentar para os dias de hoje. As ferramentas de desenvolvimento eram praticamente escassas ou inexistentes. As equipas de desenvolvimento recorriam muitas das vezes a linguagens programação de baixo nível como por exemplo Assembly, para desenvolverem o código necessário para o funcionamento do jogo. Nesta época era necessário utilizar toda a eficiência e velocidade inerente às linguagens de baixo nível. Consequentemente, muitas das tarefas passavam por otimização e desenvolvimento de tarefas computacionais simples com intuito de criar uma base sólida para o jogo. (StackOverflow, 2011)

Figura 7 - RollerCoaster Tycoon, um jogo desenvolvido com Assembly na década de 90 (Sawyer, s.d.)

A evolução tecnológica levou ao aparecimento de novas técnicas, ferramentas e ambientes de desenvolvimento, todavia, o desenvolvimento de videojogos continua a ser uma tarefa extremamente complexa. As empresas e equipas de desenvolvimento continuam a enfrentar um grande número de desafios, de carácter tecnológico, económico e social, numa tentativa de criarem novos jogos de sucesso.

A especificação de requisitos, ou neste caso, a falta de especificação de requisitos é um dos problemas mais comuns associados a esta indústria. Em qualquer contexto de desenvolvimento de software, a especificação de requisitos é um segmento fundamental para a manutenção e longevidade do sistema desenvolvido. (Possible, 2014)

O desenvolvimento de videojogos diferencia-se do desenvolvimento de software comercial por ser um tipo de sistema que requer sempre o envolvimento de várias faculdades. No desenvolvimento de qualquer sistema há a necessidade de envolver várias disciplinas diferentes, consoante a especificidade e tipo de sistema. Persistência de dados, transferência e comunicação de dados, velocidade e eficiência de algoritmos, gestão de memória e recursos, interfaces e design, são alguns exemplos de diferentes disciplinas normalmente associadas ao desenvolvimento de software. O processo de desenvolvimento videojogos vai mais longe, no sentido de além de necessitar de quase todas as disciplinas referidas anteriormente, estas requerem outro tipo de necessidades ou novas faculdades. Os videojogos modernos são uma mistura de arte, música, narrativa e todos os outros componentes anteriormente identificados. A junção de todos estes fatores leva ao desenvolvimento de sistemas e soluções extremamente complexas, reforçando a relevância em especificar os requisitos, de modo a que a viabilidade do projeto seja garantida. A produção e desenvolvimento de todas as vertentes inerentes a este tipo de soluções não deve ser realizada em paralelo, uma vez que a consolidação de todas as estruturas possibilita uma melhor visão final do projeto e de todos os recursos envolvidos.

O desenvolvimento e difusão de motores de jogos e *frameworks* de desenvolvimento de jogos mudaram a forma como os videojogos são produzidos. Estas soluções caracterizam-se por serem um tipo de software que junta várias ferramentas e funcionalidades que permitem a manipulação de ficheiros de modelos 2D e 3D, áudio, fontes de luz e efeitos luminosos, áudio, animações, simulações, entre outros. Estas ferramentas permitem às equipas abstraírem-se de conceitos e tarefas primitivas, dando-lhes mais tempo para se focarem no desenvolvimento da lógica da solução em si. (Unreal Engine, s.d.) (Unity Technologies, 2018)

Consequentemente, o uso destas soluções para fins comerciais, teve um impacto profundo na indústria. O mercado de "motores de jogos" e *frameworks* baixou drasticamente a barreira de entrada para a indústria de desenvolvimento de videojogos. Movidos pela acessibilidade e pela robustez destas ferramentas, pequenos estúdios independentes, capazes de competir com os maiores produtores de videojogos, começaram a surgir um pouco por todo lado. Deste modo, o desenvolvimento de videojogos deixou-se de estar associado a grandes equipas de software.

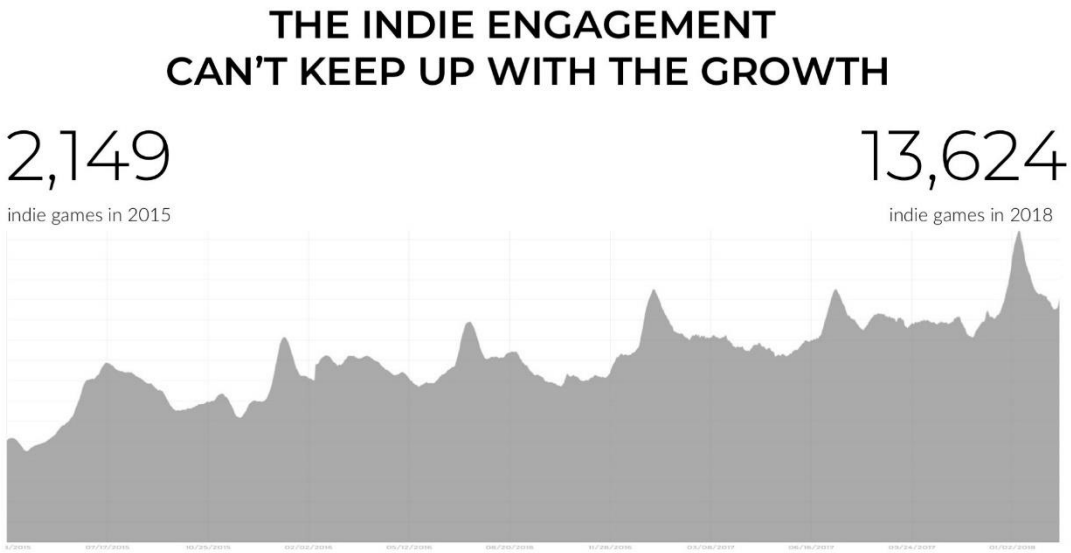

Figura 8 - Gráfico com o número de jogos lançados na plataforma Steam por estúdios independentes (Galyonkin, 2018)

O enorme acréscimo ao volume de jogos produzidos anualmente levou a uma saturação do mercado. De modo a garantirem o sucesso dos seus jogos, as empresas e equipas de desenvolvimento começaram-se a adaptar a esta nova competição no mercado. Muitas destas estratégias passaram pela criação de novos tipos de jogos, modos de jogos, gráficos e periféricos. Por lado, na tentação de obter cota de mercado, muitas empresas redesenharam o modo como o jogo é comercializado ou lançado. Muitos consumidores consideram que estas abordagens de comercialização levam a uma degradação da qualidade dos jogos desenvolvidos. (Coin-Drop, 2018)

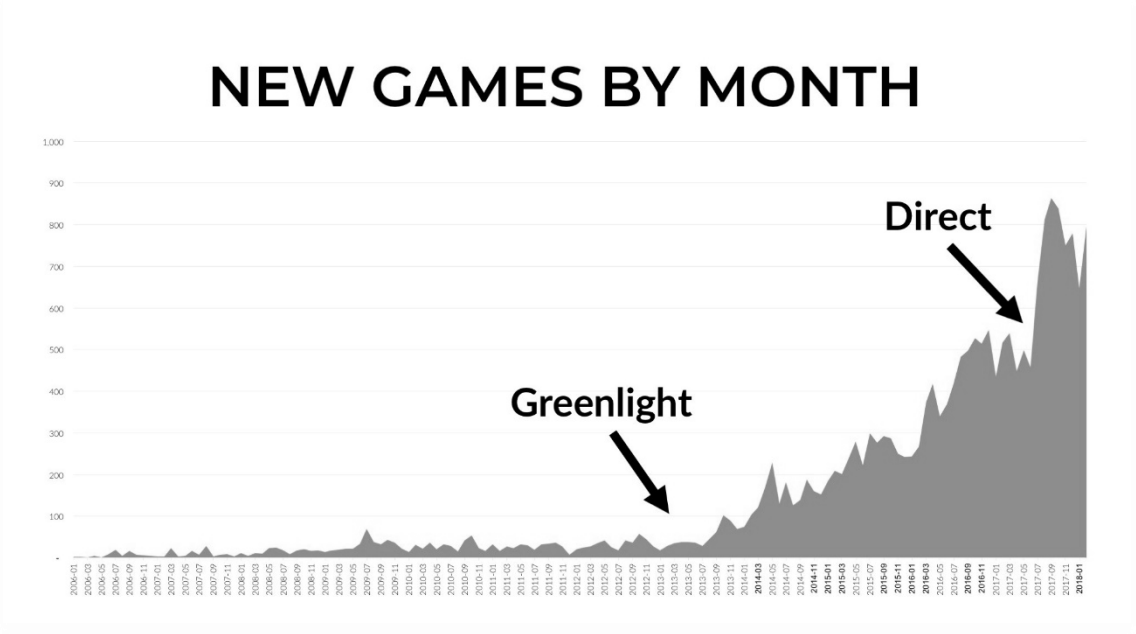

Figura 9 - Número de jogos lançados na plataforma steam por mês. Steam GreenLight e Steam Direct correspondem a lançamentos de novos programas de submissão de jogos para a plataforma. (Galyonkin, 2018)

Ano	Número de jogos lançados
2008	173
2009	271
2010	346
2011	369
2012	443
2013	576
2014	1412
2015	2575
2016	4586
2017	6767
2018	9075

Tabela 1 - Número de jogos lançados na plataforma Steam por ano (SteamSpy, s.d.)

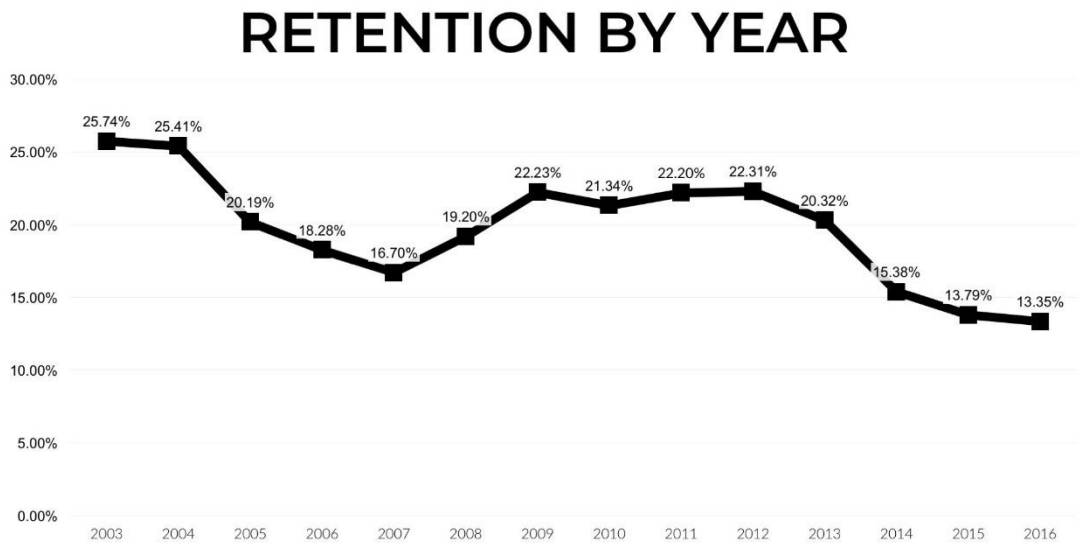

Figura 10 - Gráfico com a retenção de utilizadores na plataforma Steam por ano (Galyonkin, 2018)

O desenvolvimento de videojogos continua a ser visto como uma prática paralela em relação ao desenvolvimento de software convencional. A maior parte das instituições de ensino não possuem o programa indicado para as necessidades do mercado, levando à escassez de profissionais qualificados para esta tarefa. Do mesmo modo, a falta de programas deste tipo de ensino leva muitos a tornarem-se autodidatas. A ausência um estilo e padronização de arquiteturas deste tipo de sistema leva a uma degradação da qualidade do software bem como um aumento de custos por parte das empresas.

2.2 Estado de arte

2.2.1 Soluções existentes

Atualmente existe um conjunto diverso de aplicações e serviços nos quais é possível desenvolver videojogos e jogos sérios. Todavia, tal como foi referido na secção anterior, muitas destas ferramentas requerem conhecimentos especializado. Isto cria uma barreira de entrada elevada para quem quer começar a trabalhar neste tipo de projetos.

Segue-se uma breve descrição para cada uma das tecnologias consideradas e uma tabela de comparativa.

2.2.1.1 Unity

O Unity é um motor de jogos multiplataforma desenvolvido pela Unity Technologies. Reconhecido como um dos maiores motores de jogos disponíveis atualmente, o Unity oferece uma vasta gama de ferramentas e tecnologias para auxiliar o desenvolvimento de software na plataforma, como por exemplo: editor de jogo avançado que permite desenho, design e implementação do produto, suporte para jogos 2D e 3D, algoritmos de pesquisa que se adaptam ao meio, ferramentas para a criação e edição de interfaces gráficas de forma rápida e intuitiva, simulação, edição de áudio, suporte para ferramentas externas, entre outros. O C# é principal linguagem utilizada no desenvolvimento de jogos na plataforma, tendo outras linguagens como o Boo e UnityScript (uma linguagem altamente baseada em Javascript) sido deprecadas nas versões mais recentes da plataforma.

Inicialmente desenvolvido como um motor de jogo exclusivo para OS X, atualmente suporta mais de 20 plataformas diferentes, desde de ambientes *desktop*, dispositivos móveis, dispositivos de realidade virtual e realidade aumentada, consolas de videojogos e ambientes Web.

O Unity é também responsável por alguns dos maiores jogos atualmente disponíveis no mercado como por exemplo: *Hearthstone, Cities: Skylines, Kerbal Space Program, Cuphead*, *Pokémon Go*, entre outros. (Unity Technologies, 2018)

Figura 11 - Ambiente de desenvolvimento da plataforma Unity (Unity Technologies, 2018)

2.2.1.2 GameMaker

O GameMaker é um motor de jogos multiplataforma desenvolvido pela YoYo Games. O GameMaker caracteriza por ser um motor de jogo bastante popular para o desenvolvimento de jogos 2D, devido a sua poderosa ferramenta de *drag-and-drop*. Esta ferramenta permite aos utilizadores menos experientes tirarem partido de maior parte das funcionalidades da linguagem proprietária do GameMaker, sem o esforço de ter de aprender a linguagem ou os conceitos tecnológicos. Os resultados produzidos por esta ferramenta são convertidos internamente em código pelo o motor de jogo, o que torna esta ferramenta ainda mais apelativa quando há a necessidade de desenvolver o produto num curto espaço de tempo. O GameMaker também possui ferramentas normalmente associadas motores de jogo, como por exemplo, editor de entidades do jogo, editor de código da linguagem GameMaker, editor de terreno, suporte para animações, *debugging*, controlo de versões, entre outros.

A versão mais recente do GameMaker suporta o desenvolvimento de jogos 3D e várias plataformas como por exemplo ambientes *desktop*, dispositivos móveis, consolas de videojogos, e ambientes Web. (YoYo Games, s.d.)

Figura 12 - Ambiente de desenvolvimento da plataforma GameMaker (Steam, s.d.)

2.2.1.3 Unreal Engine

O Unreal Engine é um motor de jogos multiplataforma desenvolvido pela Epic Games. O Unreal Engine é considerado um dos maiores motores de jogos disponíveis atualmente, graças a sua vasta oferta de ferramentas que permitem uma maior facilidade no processo desenvolvimento de jogos O Unreal Engine tem à sua disposição várias ferramentas para auxiliar o desenvolvimento de jogos e simulações como por exemplo, suporte para jogos multijogador, efeitos luminosos e partículas, editor de materiais, editor de mapas e terrenos, editor de código, algoritmos de comportamento de entidades baseados em inteligência artificial, manipulação de áudio, entre outros. O C++ é a principal linguagem utilizada neste sistema, uma vez que é a linguagem utilizada no desenvolvimento da plataforma e dos jogos, contudo o Unreal Engine permite o desenvolvimento de jogos por meio da ferramenta Blueprint, que permite manipular o comportamento de diversas entidades, edição de interfaces gráficas, ajuste de controlos e muito mais sem a escrita de código, o que dá uma maior acessibilidade a programadores com menos experiência

O Unreal Engine foi inicialmente criado para suportar o desenvolvimento do jogo Unreal, mas atualmente é capaz de suportar diversas plataformas como por exemplo ambientes *desktop*, dispositivos móveis, dispositivos de realidade virtual e realidade aumentada, consolas de videojogos, e ambientes Web. (Unreal Engine, s.d.)

Figura 13 - Ambiente de desenvolvimento da plataforma Unreal Engine (Unreal Engine, s.d.)

2.2.1.4 RPG Maker MV

O RPG Maker MV é um motor de jogos desenvolvido pela Degica. O RPG Maker MV caracteriza-se por ser um motor de jogos utilizado para o desenvolvimento de jogos do tipo RPG.

Esta ferramenta permite o desenvolvimento completo de jogos do tipo de RPG sem o uso de uma linha de código. Isto deve-se ao facto de possuir diversas ferramentas de desenvolvimento de alto nível e conteúdo pré-gerado, que permitem uma abstração total do código, como por exemplo o editor de terrenos, editor de cenários de batalha, e gestor de plugins que permitem a criação de novas personagens e mapas capazes de se ajustar ao jogo desenvolvido. (RPG Maker, s.d.)

Apesar de não permitir criar todo o tipo de jogos, como as ferramentas anteriormente enunciadas, é de salientar esta solução, uma vez que tem em comum muitas características com o projeto deste documento.

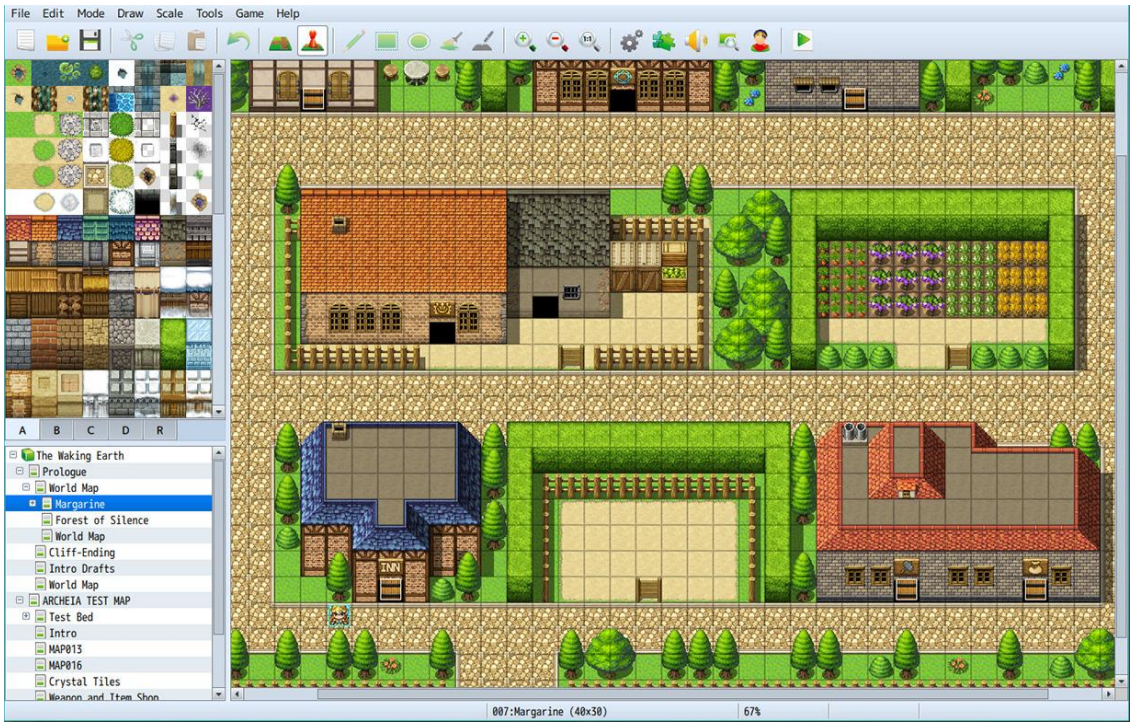

Figura 14 - Ambiente de desenvolvimento do RPG Maker MV (Steam, s.d.)

Na seguinte tabela é possível comparar as principais características das ferramentas anteriormente mencionadas

	Unity	**GameMaker**	**Unreal Engine**	**RPG Maker MV**
Complexidade da aplicação	Exige conhecimentos relacionados com o desenvolvimento de software	A ferramenta de *drag-and-drop* permite o desenvolvimento de lógica de jogo	A ferramenta Blueprint permite o desenvolvimento de lógica de jogo	Os jogos podem ser criados sem a utilização de linguagens de programação
Linguagem de programação utilizada	C#	Linguagem proprietária	C++	Javascript
Ferramentas disponíveis	Ferramentas para a criação e edição de interfaces gráficas de forma rápida e intuitiva, simulação, edição de áudio, suporte para ferramentas externas, entre outros	Ferramentas para o desenvolvimento e edição de entidades, terrenos, animações, entre outros.	Ferramentas para o desenvolvimento de jogos multijogador, efeitos luminosos, partículas, materiais, mapas, terrenos, áudio e comportamento de entidades, entre outros.	Editor de terrenos, gestor de plugins, editor de cenários de batalhas, entre outros

Modelo de negócio	Preço depende do tamanho da equipa e projeto	Preço depende do tamanho da equipa e projeto	5% da receita gerada a partir dos 3000€	Custo de 73,99€

Tabela 2 - Comparação entre as diversas plataformas de desenvolvimento

2.2.2 Jogos de cartas

Como já foi referido anteriormente neste documento, os jogos de cartas são um tipo de jogo extramente popular, com um número estimado de mais de 2 mil milhões de dólares em vendas para o ano 2020. (SuperData, s.d.) De facto, o número de jogadores deste tipo de jogos tem aumentado, tendo o número de jogadores em formato digital ultrapassado o número de jogadores de formato físico nos últimos anos. A popularidade e o crescimento deste tipo de jogos devem-se ao facto deste tipo de jogos serem bastante curtos e fáceis de utilizar em dispositivos móveis e consolas portáteis. No fundo, estas características permitem a que um jogador possa jogar em qualquer lugar durante pequenos períodos de tempo, ou até mesmo longas sessões de jogo. (Softpedia News, 2014).

Outro fator deve-se à popularidade de plataformas de transmissão em direto de jogos como o Youtube e Twitch, que permite que os jogadores possam ver outros jogadores a competir em torneios oficiais ou jogos casuais, o que contribui para uma maior interação com as comunidades deste tipo de jogos.

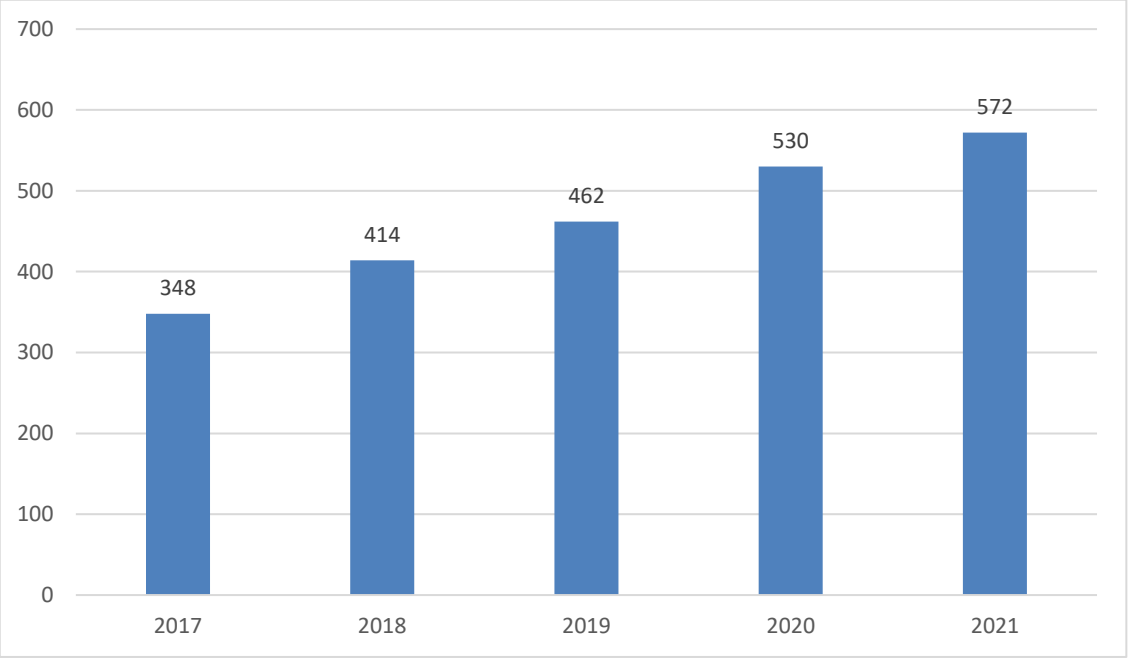

Figura 15 - Gráfico de barras com previsão para o total de vendas em jogos de cartas colecionáveis digitais em milhões de dólares na América do Norte (SuperData, s.d.)

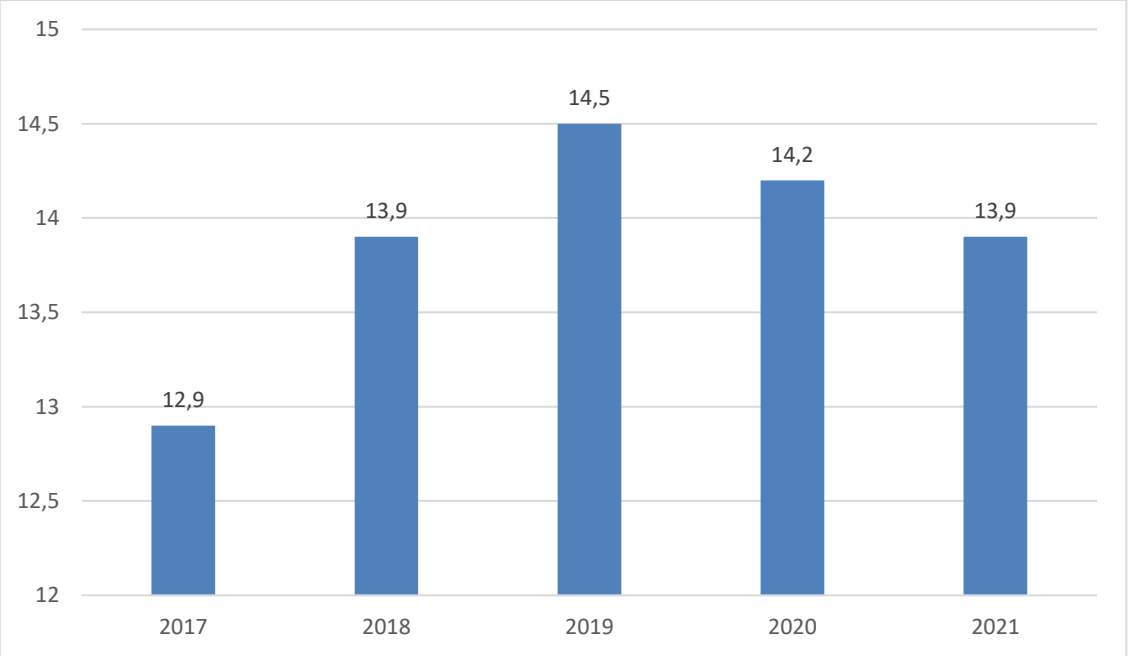

Figura 16 - Gráfico de barras com previsão para o total de milhões de utilizadores ativos em jogos de cartas colecionáveis digitai na América do Norte (SuperData, s.d.)

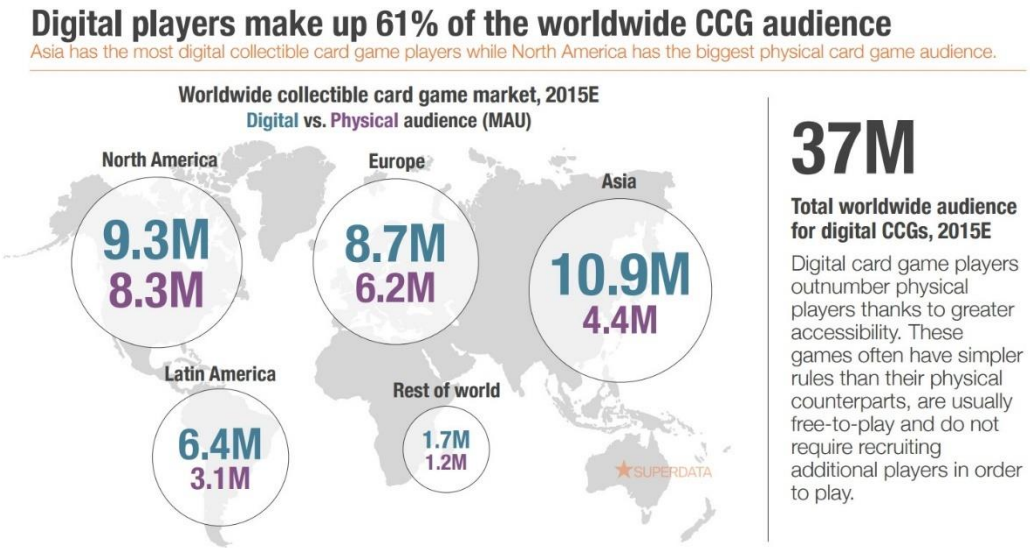

Figura 17 - Distribuição do número de jogadores de jogos de cartas em 2015 (Venture Beat, 2015)

Os jogos de cartas são diversos e extramente extensos em termos de conteúdo, alguns tendo até vários milhares de cartas de jogo. No entanto, muitos destes jogos partilham várias características e mecânicas de jogo. De forma a melhor compreender as dimensões e mecânicas deste tipo de jogos, foram analisados vários formatos de jogos de cartas de caracter digital e físico.

A tabela seguinte contem uma comparação entre várias propriedades e mecânicas dos jogos de cartas analisados.

	Nº de jogadores	**Formato**	**Criação de baralhos**	**Permite jogar as cartas para o campo**	**Permite retirar cartas do baralho**	**Permite virar cartas no campo**	**Baralho partilhado entre os jogadores?**	**Campo partilhado entre os jogadores?**
Poker (Texas Hold 'Em)	2 ou mais	Físico e digital	Não	Não	Não	Sim	Sim	Sim
BlackJack	2 ou mais	Físico e digital	Não	Não	Sim	Sim	Sim	Sim
Sueca	4	Físico e digital	Não	Sim	Não	Não	Sim	Não
Hearthstone	2	Digital	Sim	Sim	Sim	Não	Não	Não*
Magic the Gathering	2 ou mais	Físico e digital	Sim	Sim	Sim	Sim	Não	Não*
Pokemon Trading Card Game	2	Físico e digital	Sim	Sim	Sim	Sim	Não	Não*
Yu-gi-oh	2	Físico e digital	Sim	Sim	Sim	Sim	Não	Não*

Tabela 3 - Tabela comparativa de alguns jogos de cartas

Algumas destas propriedades e características são difíceis de comparar e analisar na tabela anterior. Por exemplo, nos jogos em que é possível criar um baralho personalizado, é comum existir um sistema complexo em que os atributos de jogo são um conjunto de pontos pertencentes aos jogadores, sendo que o objetivo é e retirar estes mesmos pontos ao adversário. Consequentemente, o conceito de o campo ser partilhado não é absoluto. Os jogadores não conseguem interagir com o campo do adversário, mas, uma vez que existem milhares de cartas disponíveis com diversos e diferentes efeitos, é possível influenciar e alterar o estado do campo do adversário, ainda que indiretamente.

Figura 18 - Hearthstone, um dos jogos de cartas colecionáveis mais populares. (Hearthstone, s.d.)

2.3 Visão da Solução

Deste modo, tendo por base no estudo do estado atual das soluções e nas limitações existentes, a solução a desenvolvida visa a permitir o desenvolvimento de jogos sérios de cartas.

A aplicação deve ser capaz de permitir a edição de mecânicas, e propriedades e características inerentes aos jogos de cartas. Para este projeto foram consideradas as seguintes mecânicas de jogo:

- Retirar carta do baralho
- Jogar carta
- Virar carta
- Retirar carta do campo

Do mesmo modo, também deve ser possível editar as propriedades e características da carta. Os elementos considerados são:

- Imagem da carta
- Nome da carta
- Valor da carta
- Tipo da carta (como por exemplo um naipe)

Apesar do principal objetivo de a aplicação ser o desenvolvimento de jogos sérios de cartas, também há a possibilidade de ser utilizada para a prototipagem rápida de jogos de cartas digitais.

Outro aspeto fundamental a considerar é a interface com o utilizador. Uma vez que esta ferramenta tem o objetivo de realizar uma abstração completa do código, devem ser criadas ferramentas para a criação e edição de cartas bem como os baralhos de jogo.

Por fim, a aplicação deve também permitir criar algum tipo de persistência de dados do jogo criado.

2.4 Análise de Valor

Um pouco por todo o mundo, milhares de instituições empresariais tentam lançar novos produtos ou ideias para o mercado, numa tentativa de prosperarem neste meio competitivo. O desenvolvimento de um novo produto é um processo complexo e heterogéneo. Pesquisas, estudo de mercado, uso de novos materiais ou tecnologias, produção em massa e provas de conceito são alguns dos métodos normalmente empregues pelas instituições. Todos os dias, novas ideias surgem e alcançam o sucesso. Mas nem todas chegam a ter sucesso. Por cada ideia ou novo produto que surge, centenas de outras falham.

O crescimento industrial e evolução tecnológica alcançaram uma nova era, culminando num estado de saturação. Um mercado inundado e saturado demonstra que a inovação é cada mais importante para o sucesso. O sucesso parte muitas vezes pela mudança e pela criação de algo novo. Por muito que uma ideia seja inovadora e que resolva um problema, o sucesso continua a ter uma componente subjetiva por parte mercado. O valor que uma ideia trás é ditada pelo consumidor final. Se o consumidor não está disposto a pagar pelo produto, então esta não tem valor para a sociedade.

Numa tentativa de unificar o processo inovativo, e assim aumentar as probabilidades de sucesso neste mercado competitivo, começaram a surgir novas ideias e metodologias de desenvolvimento de produto. As abordagens mais tradicionais tendem em dividir o processo de inovação em três áreas de estudo. A área de comercialização, a área de desenvolvimento do novo produto e a área do *Fuzzy Front End*. A área do *Fuzzy Front End* é a área mais abstrata dos três, uma vez que é nesta secção que se dá o processo que envolve o desenvolvimento de uma nova ideia. O processo criativo é diferente de organização para organização, bem como de para cada ser individual. Enquanto as outras áreas possuem metodologias e padrões definidos, a subjetividade do *Fuzzy Front End* implica que não haja linearidade no processo. A falta de meios funcionais e concretos causa instabilidade na fase inicial do processo, o que acaba por agravar a sustentabilidade e sucesso posterior do projeto.

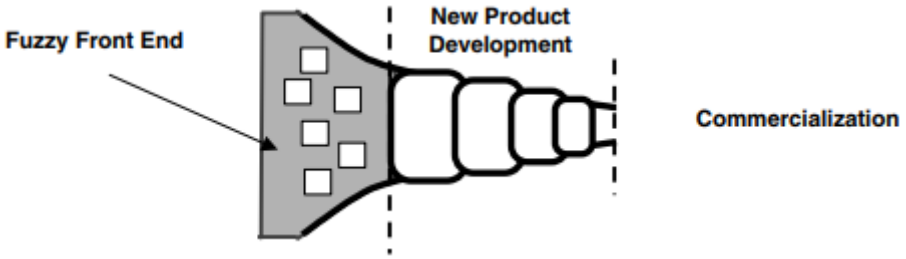

Figura 19 - Processo de Inovação (Peter A.Koen, 2004)

De modo a resolver este problema, Peter A.Koen desenvolveu uma nova abordagem, juntamente com outros 7 autores, no livro *The PDMA ToolBook for New Product Development,* denominada de *New Concept Development Model*. O *New Concept Development Model* surgiu como uma forma de centralizar e unificar esta parte do processo de inovação. Ao agregar e combinar diversas experiências e ideias de profissionais líderes na indústria foi possível alcançar um processo unificado e mais eficiente, que prometia uma abordagem mais linear e comum para o desenvolvimento inovativo. (Peter A.Koen, 2004)

O *New Concept Development Model* é dividido em três partes principais:

1. Área central que define cinco elementos chave que definem o processo de inovação:
    - Identificação de oportunidade
    - Analise de oportunidade
    - Criação de novas ideias
    - Seleção de ideias
    - Desenvolvimento de conceito
2. A força motriz que impulsiona os cinco elementos chave. A cultura e gestão da organizacional são fulcrais para a funcionamento correto destes cinco elementos.
3. Os fatores externos, que influenciam o processo como um todo:
    - Mundo
    - Fatores científicos
    - Estratégias de negócio
    - Capacidades da organização

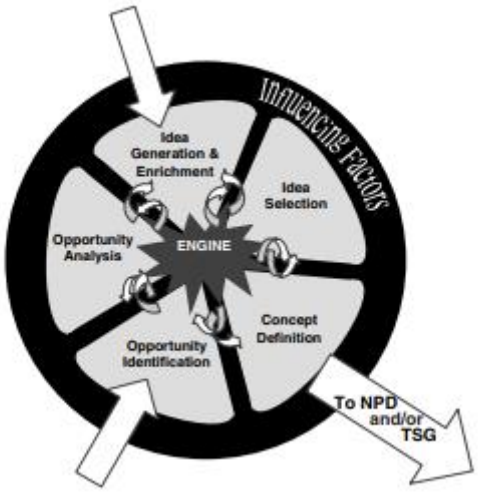

Figura 20 - New Concept Development Model (Peter A.Koen, 2004)

2.4.1 Identificação de Oportunidade

A identificação de oportunidade pode surgir de diferentes fatores. As derivações mais comuns tendem surgir por evolução cultural ou como resposta a uma necessidade existente. Outras instâncias de identificação de oportunidade podem surgir através de melhorias a um produto ou serviço já existente. Pequenos ajustes ou até mesmo mudanças radicais na eficiência ou no preço representam alterações que podem ser uteis na racionalização de uma nova oportunidade de negócio. A adaptação como resposta a uma ameaça competitiva também é outro fator comum na identificação de uma nova oportunidade.

A identificação de oportunidade que o presente documento pretende demostrar baseia-se na combinação dos seguintes fatores atuais:

- Os motores de jogos continuam a ser uma ferramenta extremamente complexa e difícil de usar.
- Os jogos de cartas são utilizados como uma forma alternativa de ensino.
- Os jogos de cartas digitais estão em forte expansão e desenvolvimento.

2.4.2 Análise de Oportunidade

A transformação de uma identificação de uma oportunidade numa ideia de negócio necessita de uma análise capaz de suportar as necessidades e requisitos existentes. Esta análise normalmente é suportada por estudos de mercado, debates de benefícios e cortes a realizar, estudos tecnológicos e científicos, entre muitos outros.

A extensão e complexidade da análise reflete os esforços realizados pelos intervenientes envolvidos no desenvolvimento, da cultura empresarial.

No contexto deste projeto, é particularmente interessante analisar abordagens semelhantes. Um produto que se destaca pelas semelhanças na abordagem pretendida é o RPG Maker. O RPG Maker é um sistema de software que permite o desenvolvimento de jogos do tipo RPG. A característica mais interessante do RPG Maker é o facto que este software permite o desenvolvimento completo deste tipo de jogos sem a necessidade de escrever uma única linha de código. O sucesso desta ferramenta permitiu que as equipas desenvolvimentos conseguissem lançar comercialmente os seus jogos em plataformas de venda de videojogos como por exemplo a Steam. (Eurogamer, 2018)

As semelhanças entre a solução desenvolvida e o a solução anteriormente apresentada são um bom ponto de partida para a demonstração da viabilidade do projeto.

2.4.3 Criação de novas ideias

Esta fase é caracterizada pelo desenvolvimento e maturação da ideia inicialmente identificada. Durante esta fase do processo, é comum as ideias sofreram várias iterações que culminam em diferentes abordagens, estudos e provas de conceito.

Com base na análise realizada anteriormente e dos pressupostos do projeto, é de salientar as seguintes ideias:

- Uma ferramenta com as mecânicas e características mais comuns nos jogos de cartas e possibilitar a edição sequencial destas mesmas características.
- Possibilidade de escrita e edição código, de modo a permitir uma maior liberdade e dinâmica dentro da ferramenta.

2.4.4 Seleção de ideias

A escolha de uma ou várias ideias pode ser realizada de diferentes abordagens. Deste de escolha de ideias que possibilitem uma maior segurança na entrada do mercado, até mesmo escolha individual preferida.

A primeira opção é mais interessante, uma vez que é a possibilidade que permite baixar a barreira de entrada a utilizadores menos experientes e é uma abordagem alternativa ao processo de desenvolvimento convencional.

A segunda opção acaba por encaixar na perspetiva de solução de motor de jogos, o que acaba por não acrescentar valor.

2.4.5 Desenvolvimento do conceito

O último elemento deste modelo envolve o desenvolvimento de uma prova de conceito baseada nos necessidades e possibilidades de todos os intervenientes do projeto.

O desenvolvimento do projeto é explicado com detalhe no capítulo seguinte.

2.5 Definição de valor

2.5.1 Valor

Valor é um termo subjetivo e difícil de avaliar. A definição de valor varia de acordo com a posição do sujeito que avalia o produto ou serviço. Para o produtor ou comerciante, a definição de valor pode variar consoante os custos de produção ou opinião do consumidor. Por outro lado, para o consumidor final, a definição de valor pode passar por pelo preço de aquisição do produto/serviço, qualidade do produto/serviço (outro atributo que é subjetivo e depende da opinião individual de cada sujeito), utilidade do produto, e até mesmo pelo tipo de produto.

2.5.2 Valor na ótica de consumidor

O valor para o consumidor é muitas vezes associado à definição genérica de valor. Na realidade, o valor para consumidor passa pela perceção e avaliação individual do sujeito em relação ao que é transmitido, em forma de produto ou serviço, por uma empresa. Esta avaliação passa pelo o discernimento dos benefícios e sacrifícios associados à aquisição e uso do produto ou serviço. No artigo *Conceptualising 'Value for the Customer'*, T.Woodall vai mais longe e define o valor para o consumidor como *Value for the customer (VC) is any demand-side, personal perception of advantage arising out of a customer's association with an organisation's offering, and can occur as reduction in sacrifice; presence of benefit (perceived*

as either attributes or outcomes); the resultant of any weighed combination of sacrifice and benefit (determined and expressed either rationally or intuitively); or an aggregation , over time, of any or all of these. (Woodall, 2003)

Num modelo proposto por Shillito e DeMarle, a definição de valor é repartida em quatro definições distintas que são influenciadas pelo contexto social e pelos valores subjetivos do indivíduo. As quatro definições são:

- ***Exchange value***: A definição de valor é orientada em torno do produto ou serviço, e é caracterizada pela natureza do objeto e pelo mercado em que se insere. No entanto, o sujeito pode reavaliar o valor que é oferecido.
- ***Intrinsic value***: A definição de valor é orientada em torno do produto ou serviço. Neste caso o valor atribuído é caracterizado antes ou durante a experiência de utilização.
- ***Use value***: A definição de valor é orientada à volta do sujeito que utiliza o produto ou serviço. O valor é caracterizado durante ou depois da utilização.
- ***Utilitarian value***: A definição de valor é orientada à volta do sujeito que utiliza o produto ou serviço. O valor é caracterizado pelos sacrifícios que o consumidor tem de fazer durante o uso do produto

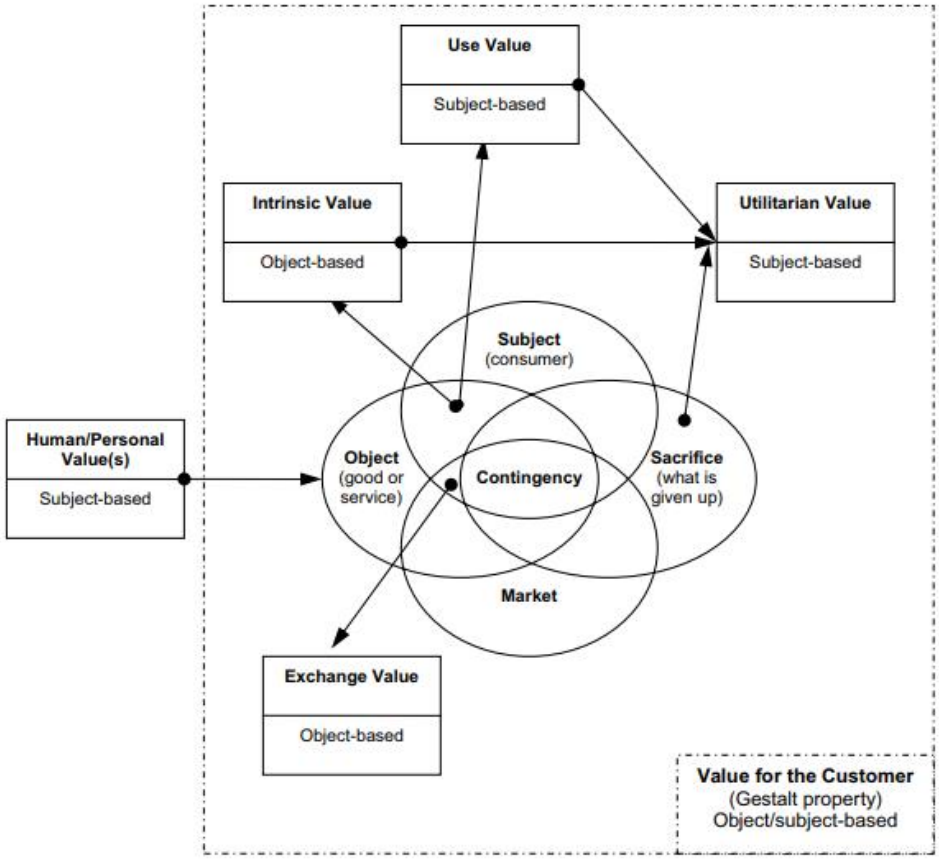

Figura 21 - Modelo preliminar proposto por Shillito e DeMarle (Woodall, 2003)

2.5.3 Valor percebido

O valor percebido relaciona-se com a perspetiva de valor do consumidor. O valor percebido baseia-se num balanço entre a utilidade entre um dado produto ou serviço e os custos associados. A utilidade pode ser vista como um conjunto de fatores que influenciam a opinião do utilizador relativamente a um determinado produto ou serviço. Os fatores mais comuns passam pela estética, disponibilidade, prestígio e funcionalidades.

2.5.4 Perspetiva longitudinal de valor

No artigo *Conceptualising 'Value for the Customer'*, T.Woodall propôs uma nova abordagem que relaciona a experiência do utilizador com a forma de como valor é atribuído. Nesta nova perspetiva, T.Woodall defende que o valor para o consumidor é constituído por quatro fases temporais, que contribuem como um todo. As quatro fases são:

- Pré-aquisição
- Momento da transação
- Pós-aquisição
- Pós-uso

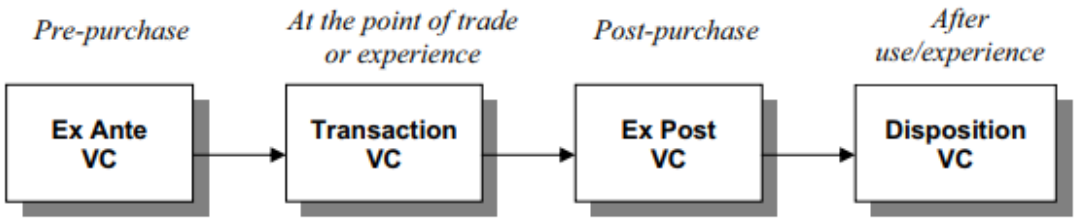

Figura 22 - As 4 fases da perspetiva longitudinal de Woodall (Woodall, 2003)

No contexto deste documento, esta abordagem pode ser relacionada com os benefícios e sacrifícios associados ao consumidor final.

	Pré-aquisição	**Momento de transação**	**Pós-aquisição**	**Pós-uso**
Benefícios	Desejo de uma aplicação para o desenvolvimento de jogos sérios simples e intuitiva.	Aquisição da aplicação.	Facilidade na criação e edição dos atributos e mecânicas base dos jogos sérios de cartas	Biblioteca de jogos sérios desenvolvidos
Sacrifícios	Necessidade de compreender o funcionamento da ferramenta.	Necessidade de compreender o funcionamento da ferramenta	Desenvolvimento da lógica dos jogos sérios	Substituição por novas tecnologias ou ferramentas

Tabela 4 - Valor da solução sob o ponto de vista da perspetiva longitudinal de Woodall

2.6 Proposta de Valor

Em qualquer negócio, é necessário transmitir ao consumidor os benefícios do produto ou serviço que está a ser comercializado. A proposta de valor é um dos meios mais comuns estabelecer a mensagem a ser transmitida, uma vez que define as vantagens inerentes ao produto ou serviço.

No contexto no caso de estudo deste documento, a proposta de valor passa pelo desenvolvimento de uma aplicação que facilita o desenvolvimento de jogos de cartas para que possam ser usados no contexto de educação. Ao abstrair o conceito de programação de

aplicações e permitir a edição de mecânicas e regras de jogo, o consumidor tem mais tempo para se dedicar o desenvolvimento das cartas e dos valores que pretende transmitir, sem passar pela tarefa monótona de desenvolver as tarefas básicas associadas ao processo de desenvolvimento de aplicações.

2.7 Modelo Canvas

O modelo Canvas sumariza o processo que uma organização se relaciona com o mundo e com a sua área de negócio. O modelo Canvas está divido em nove secções que representam diferentes objetivos e atributos de negócio.

Parceiros Chave: Entidades ou organizações externas que são fundamentais para o sucesso do desenvolvimento ou comercialização do produto.

Atividades Chave: Atividades necessárias para o sucesso do desenvolvimento ou comercialização do produto.

Recursos Chave: Recursos fundamentais para o sucesso do desenvolvimento ou comercialização do produto.

Proposta de Valor: Características do produto que se destacam da competição

Relações com os Clientes: Estratégia ou maneira de como a organização interage com os clientes.

Segmentos de Cliente: Esta secção identifica os diferentes tipos de clientes.

Canais de distribuição: Esta secção identifica a estratégia de entrega do produto ou serviço.

Custos: Identifica as principais fontes de custos para o desenvolvimento e/ou comercialização do produto.

Fontes de Receita: Identifica as principais fontes de rendimento do produto.

A seguinte tabela demonstra o modelo Canvas criado para a solução desenvolvida.

<table>
<tr><td rowspan="2">Parceiros chave:

Instituições de ensino

Programadores de e empresas sistemas gráficos</td><td>Atividades Chave:

Desenvolvimento de ferramenta de desenvolvimento de jogos sérios</td><td rowspan="2">Proposta de Valor:

Ferramenta de desenvolvimento de jogos de cartas interativa</td><td>Relações com os Clientes:

Feedback</td><td rowspan="2">Segmentos de Cliente:

Instituições de ensino

Programadores e empresas de sistemas gráficos</td></tr>
<tr><td>Recursos Chave:

Mecânicas e características base de outros jogos de cartas</td><td>Canais de Distribuição:

Não aplicável nesta fase</td></tr>
<tr><td colspan="3">Custos:

Desenvolvimento da plataforma (esforço humano, custos de ferramentas de desenvolvimento)

Custos de aquisição de software</td><td colspan="2">Fontes de Receita:

Não aplicável nesta fase</td></tr>
</table>

Tabela 5 - Modelo Canvas

2.8 Analisar o valor de negócio

Tal como já foi referido anteriormente, a definição de valor depende bastante do sujeito que está a avaliar bem como do meio ou produto em que se insere. Isto torna-se um problema quando existe a necessidade de avaliar o valor intrínseco do nosso negócio. Uma abordagem interessante, introduzida por V.Allen, consiste avaliar o negócio como uma rede de relações, baseado na crescente popularidade do conceito de *networking*.

Esta nova abordagem, chamada de *Value Networks*, define que todas as relações, interações e intervenientes devem ser mapeados numa rede de ligações. Ao definir um mapa de todas as ligações existentes num negócio, é possível ter uma visão mais significativa de como todos os intervenientes interagem para o sucesso do negócio. Este modelo destaca-se por não se cingir apenas às trocas ou interações entre bens materiais. De facto, este modelo permite mapear

conceitos abstratos e intangíveis como por exemplo a promoção de um produto, *feedback* de clientes, colaborações, entre outros. (Allee, 2008)

No contexto do presente documento, foi elaborado o seguinte diagrama de *Value Networks*.

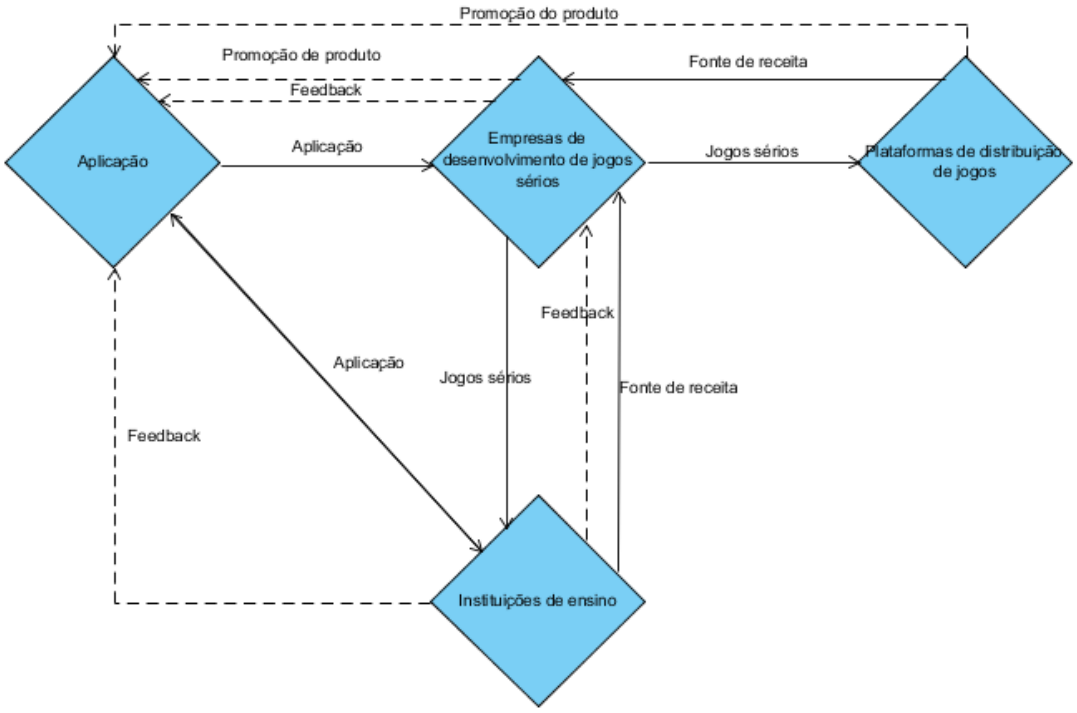

Figura 23 -Diagrama de *Value Network* baseado no modelo de V.Allee

Com base no diagrama apresentado, é de salientar o feedback e promoção do produto desenvolvido como consequências da interação entre as diversas entidades. Estas duas características são de extrema importância uma vez que permitem a melhorar o produto bem como desenvolver novos formas de capitalização.

2.9 Método de Análise Hierárquica

Um dos métodos mais utilizados na toma de decisões que envolvem múltiplos critérios e alternativas é o método de análise hierárquica. O método consiste em dividir o problema em vários níveis hierárquicos, consoante a natureza dos fatores intervenientes, como por exemplo objetivo, critérios de avaliação e soluções. A divisão em vários níveis hierárquicos permite uma melhor compreensão e definição dos fatores intervenientes na escolha final.

No contexto do documento, foram escolhidos três critérios:

- Interface e interação: Este atributo relaciona-se com interação da ferramenta com o utilizador.
- Funcionalidades: Avalia a necessidade de existir funcionalidades avançadas (como por exemplo edição de código), de modo a permitir um maior controlo da ferramenta
- Modularidade: Avalia a necessidade e possibilidade de edição dos vários atributos e mecânicas relacionados com as mecânicas dos jogos de cartas.

E duas alternativas:

- Solução 1: Ferramenta com as mecânicas e características mais comuns nos jogos de cartas e possibilitar a edição sequencial destas mesmas características.
- Solução 2: Possibilidade de escrita e edição código, de modo a permitir uma maior liberdade e dinâmica dentro da ferramenta.

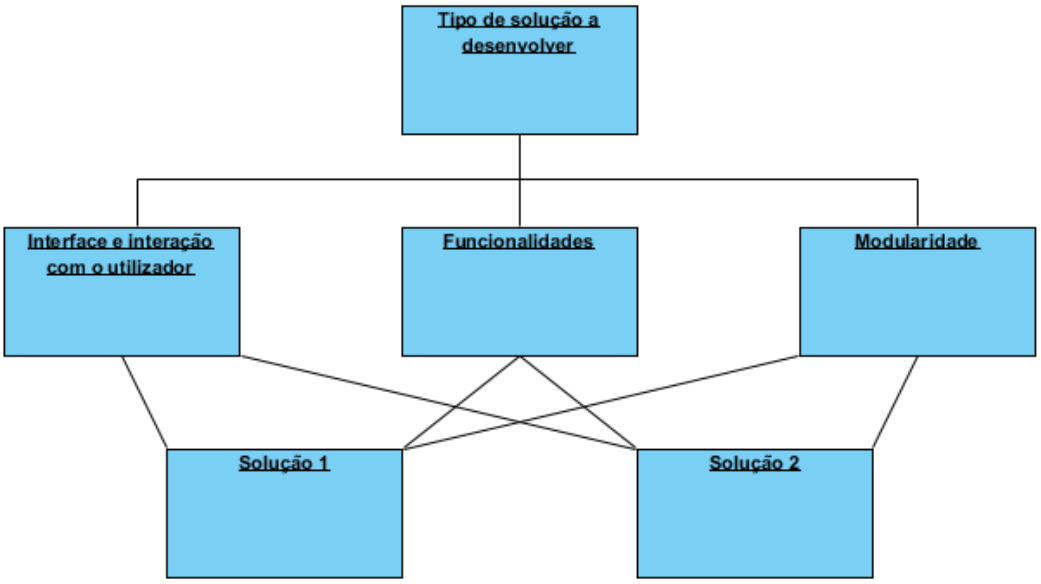

Tabela 6 – Divisão do problema em hierarquias

	Interação	Funcionalidades	Modularidade
Interação	1	3	1/3
Funcionalidades	1/3	1	1/4
Modularidade	3	4	1

Tabela 7 – Tabela de comparação

Interação	Funcionalidades	Modularidade
0,2684	0,1172	0,6144

Tabela 8 – Vetor de prioridades

Razão das consistências = 0,0706 < 0,1

Logo, os valores das prioridades relativas são consistentes

	Interação	Funcionalidades	Modularidade
Solução 1	1/2	1/4	4/5
Solução 2	1/2	3/4	1/5

Tabela 9 – Tabela de comparação paritária para os três critérios

	Interação	Funcionalidades	Modularidade	Total
Solução 1	0,1342	0,0293	0,4915	0,6550
Solução 2	0,1342	0,0879	0,1229	0,3450

Tabela 10 – Tabela de prioridades compostas para cada alternativa

Com base nos valores obtidos do método de análise hierárquica, a solução 1 é a mais favorável, uma vez que favorece a simplicidade da modularidade da edição dos atributos e mecânicas inerentes aos jogos de cartas.

3 Descrição técnica

Este capítulo descreve a componente técnica e tecnológica realizada ao longo da unidade curricular. O presente capítulo relata o trabalho desenvolvido na área da Engenharia de Requisitos, Análise, Design, Aplicação, Testes e Avaliação.

3.1 Engenharia de Requisitos

A secção Engenharia de Requisitos consiste na análise e documentação dos requisitos e restrições necessários para o desenvolvimento de um sistema de software. A Engenharia de Requisitos deve ser a primeira tarefa a realizar, uma vez que todas as outras fases envolvidas dependem de esta. Neste documento, a Engenharia de Requisitos está divida em requisitos funcionais, requisitos não funcionais e casos de uso.

3.1.1 Requisitos funcionais

A secção requisitos funcionais enumera os principais requisitos funcionais do sistema. Os requisitos funcionais representam as funcionalidades principais que o sistema deve ter.

Os principais requisitos funcionais do sistema são:

- Criação e edição de jogos de cartas, bem como a manipulação de todos os atributos e regras de jogo.
- Criação de baralhos e cartas.
- Permitir jogar os jogos desenvolvidos.

3.1.2 Requisitos não funcionais

Esta secção enumera os principais requisitos não funcionais do sistema. Os requisitos não funcionais do sistema distinguem-se dos requisitos funcionais, uma vez que representam características ou comportamentos que o sistema deve ter. O modelo FURPS+ define que os requisitos não funcionais devem ser agrupados consoante a sua natureza.

Tipo de Requisito	**Descrição**
Funcionalidades	Especifica as funcionalidades que não se relacionam com os casos de uso, nomeadamente: Auditoria, Reporte, Interoperabilidade e Segurança
Usabilidade	Avalia a interface com o utilizador. Possui diversas subcategorias, entre elas: prevenção de erros, estética, design, ajudas, documentação, design e padrões.
Fiabilidade/Confiabilidade	Refere-se a integridade, conformidade e interoperabilidade do software. Os requisitos a serem considerados são: frequência e gravidade de falha, possibilidade de recuperação, possibilidade de previsão, exatidão, tempo médio entre falhas.
Desempenho	Avalia os requisitos de desempenho do software, nomeadamente: tempo de resposta, consumo de memória, utilização de CPU, capacidade de carga e disponibilidade da aplicação
Suportabilidade	Os requisitos de suportabilidade agrupam várias características como: testabilidade, adaptabilidade, manutenibilidade, computabilidade, configurabilidade, instabilidade, escalabilidade, entre outros

Restrições: Design	Especifica ou restringe o processo de design do sistema. Exemplos podem incluir: padrões de software, processo de desenvolvimento de software, uso de ferramentas de desenvolvimento, biblioteca de classes, etc …
Restrições: Implementação	Especifica ou restringe o código ou a construção de um sistema, tais como: normas de implementação, linguagens de implementação, políticas de integridade de base de dados, limites de recursos, sistema operativo.
Restrições: Interface	Especifica ou restringe das funcionalidades inerentes a interface entre o sistema e outros.
Restrições físicas	Especifica uma limitação ou requisito do hardware utilizado, por exemplo: material, forma, tamanho ou peso.

Tabela 11 - Tabela de requisitos não funcionais

Os principais requisitos não funcionais do sistema desenvolvido podem ser consultados na seguinte tabela:

Tipo de Requisito	**Requisito**
Usabilidade	A interface deve ser simples e intuitiva

Suportabilidade	A aplicação deve permitir várias configurações de jogos
Restrições: Implementação	O sistema deve ser desenvolvido num motor de jogos ou API com suporte a desenvolvimento de sistemas gráficos. A aplicação deve suportar sistema operativo Windows e Linux
Restrições: Interface	A interface deve implementar um sistema de *drag-and-drop* para o desenvolvimento de cartas e baralhos

Tabela 12 - Tabela de requisitos não funcionais do sistema desenvolvido

3.1.3 Casos de Uso

A presente secção descreve os casos de uso (ou funcionalidades) determinados com base nos requisitos anteriores. O caso de uso é uma funcionalidade do sistema que pode ser realizada por um ou maiores intervenientes (atores). O ator é a entidade que interage com o sistema desenvolvido, como por exemplo um utilizador ou serviço.

3.1.3.1 Caso de uso 1 – Criar novo jogo

O utilizador inicia no sistema o processo de criar um novo jogo. O sistema solicita a introdução dos parâmetros e regras do jogo. O utilizador insere os dados. O sistema valida. O sistema guarda a informação e mostra uma notificação de sucesso da operação.

3.1.3.2 Caso de uso 2 – Editar jogo

O utilizador inicia no sistema o processo de editar um jogo. O sistema solicita a escolha de um jogo. O utilizador escolhe. O sistema solicita a introdução dos parâmetros e regras do jogo. O utilizador insere os dados. O sistema valida. O sistema guarda a informação e mostra uma notificação de sucesso da operação.

3.1.3.3 Caso de uso 3 – Criar nova carta de jogo.

O utilizador inicia no sistema o processo de criar uma nova carta de jogo. O sistema solicita a introdução dos atributos e imagem da carta. O utilizador insere os dados. O sistema valida. O sistema guarda a informação e mostra uma notificação do sucesso da operação.

3.1.3.4 Caso de uso 4 – Editar carta de jogo

O utilizador inicia no sistema o processo de editar uma carta de jogo. O sistema solicita a escolha da carta. O utilizador escolhe. O sistema solicita a introdução dos atributos e imagem da carta. O utilizador insere os dados. O sistema valida. O sistema guarda a informação e mostra uma notificação do sucesso da operação.

3.1.3.5 Caso de uso 5 – Remover carta de jogo

O utilizador inicia no sistema o processo de editar uma carta de jogo. O sistema solicita a escolha da carta. O utilizador escolhe. O sistema valida. O sistema guarda a informação e mostra uma notificação do sucesso da operação.

3.1.3.6 Caso de uso 6 – Editar baralho de jogo

O utilizador inicia no sistema o processo de editar um baralho de jogo. O sistema solicita o baralho a ser editado. O utilizador escolhe. O sistema solicita a escolha das cartas que compõem o baralho de jogo. O utilizador escolhe as cartas. O sistema valida. O sistema guarda a informação e mostra a notificação do sucesso da operação.

3.1.3.7 Caso de uso 7 – Jogar o jogo desenvolvido.

O jogador solicita no sistema o processo de jogar o jogo desenvolvido. O sistema inicia a partida.

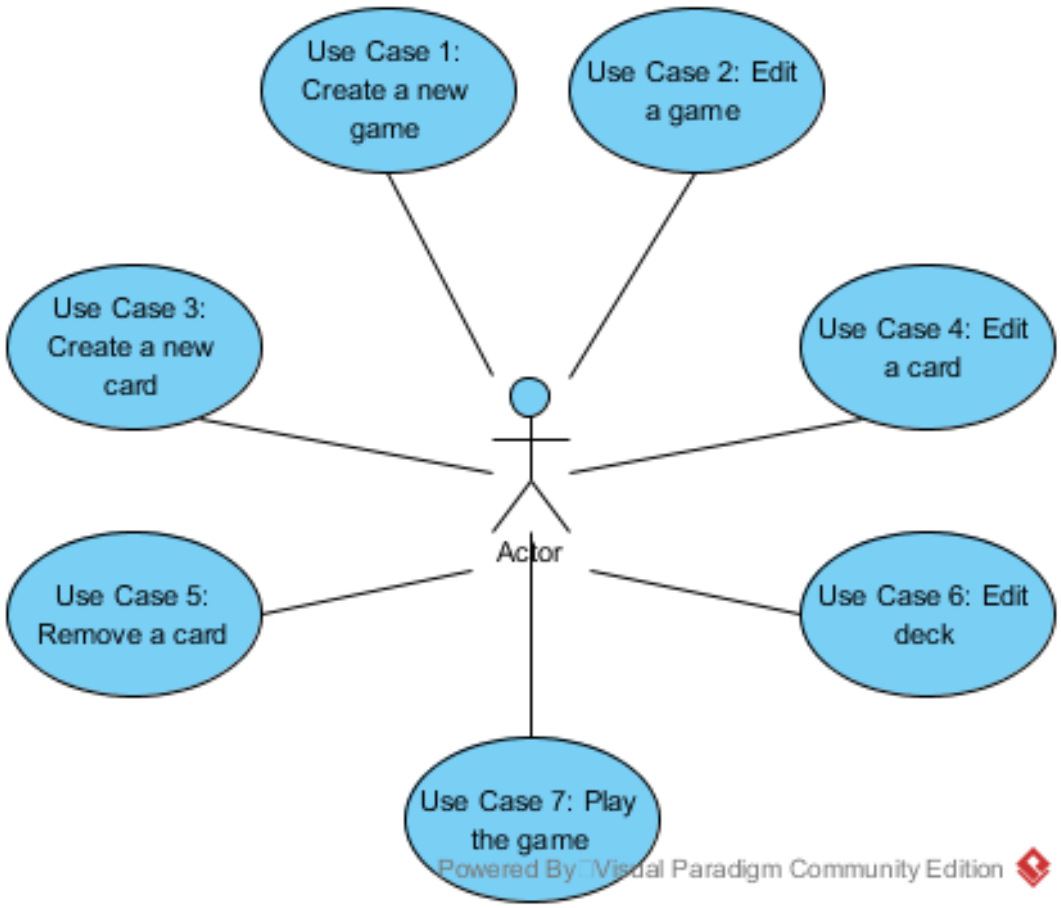

Figura 24 - Diagrama de casos de uso

3.2 Análise

A análise consiste em criar um modelo dos requisitos funcionais do sistema que seja abstrato da implementação do sistema. Os principais objetivos desta secção compreendem a descrição do domínio numa perspetiva de classificação de entidades bem como diminuir a separação que existe entre os requisitos e design da solução.

3.2.1 Modelo de domínio

O modelo de domínio é um artefacto produzido no contexto na disciplina de Modelação de Negócio. O modelo de domínio representa os principais conceitos, atributos e relações entre as classes conceptuais do domínio do sistema. As classes conceptuais representadas no modelo de domínio não representam necessariamente classes de código, uma vez que o modelo de domínio se insere na fase de Análise. O seguinte diagrama reflete as diversas classes conceptuais, relações e atributos inerentes ao problema.

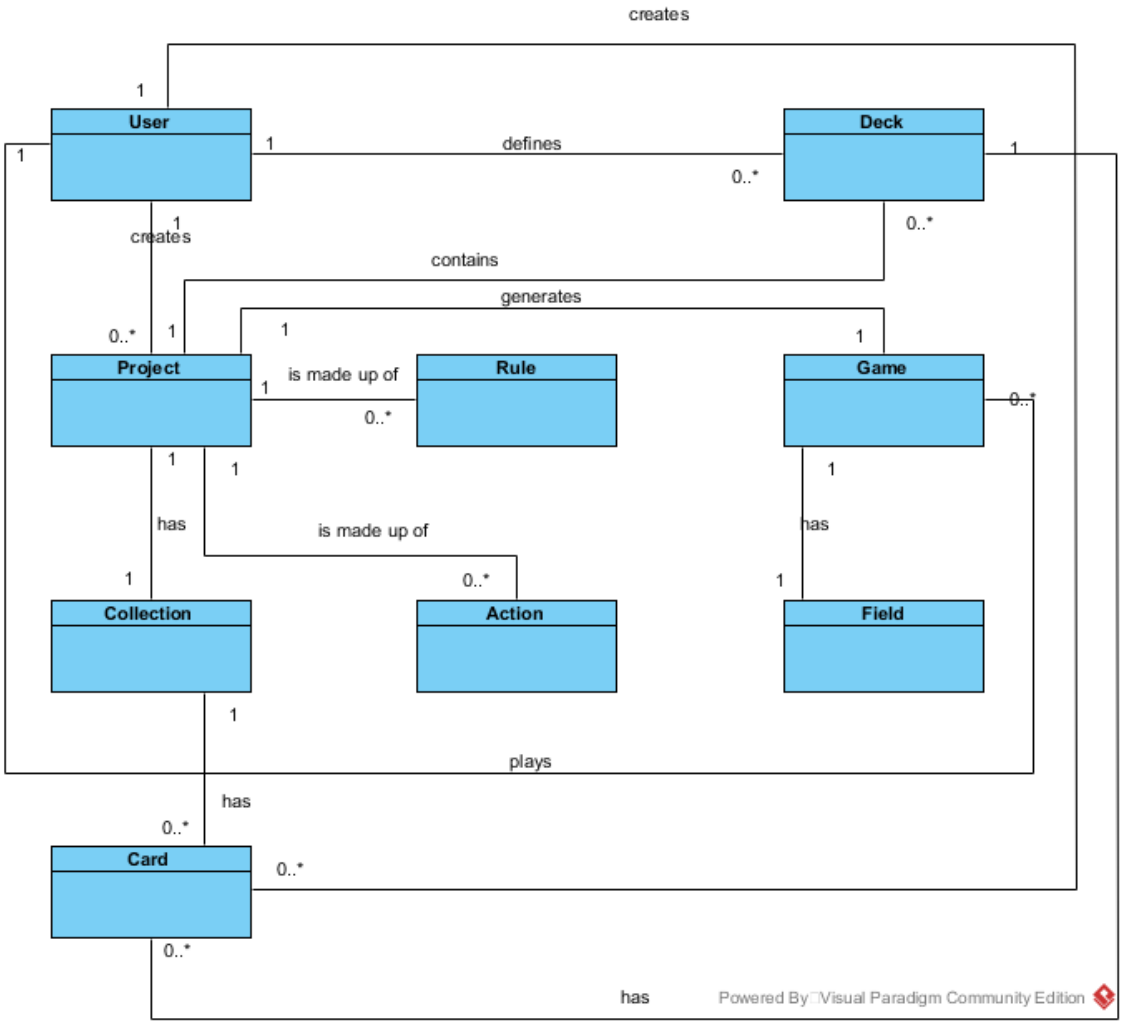

Figura 25 - Diagrama de modelo de domínio da aplicação

3.2.1.1 Utilizador

O utilizador tem como responsabilidade fazer uso de todas as funcionalidades existentes.

3.2.1.2 Projeto

A entidade projeto contem todas as definições das regras, características, baralhos e cartas do jogo. O projeto é uma entidade maleável que o permite ao utilizador fazer gestão de todos os recursos inerentes ao jogo.

3.2.1.3 Regras de Jogo

As regras de jogo correspondem a definições e características da dinâmica de jogo. Por exemplo as regras de jogo podem ser o número de cartas jogadas por ronda ou o número de cartas inicial.

3.2.1.4 Ações

As ações são semelhantes às regras de jogo, mas como o nome indica, referem-se às ações pode fazer durante o jogo. Por exemplo a ação pode ser mudar de turno, jogar uma carta, virar uma carta, retirar uma carta do baralho, entre outros

3.2.1.5 Jogo

O jogo corresponde ao produto desenvolvido. Ao contrário do projeto, em que as regras e características de jogo não passam de definições. o jogo é constituído por um conjunto de entidades funcionais que interagem entre si de modo a permitir um fluxo de jogo continuo.

3.2.1.6 Carta

A carta é a entidade principal do jogo, uma vez que os jogos desenvolvidos são desenvolvidos à volta deste conceito. Uma carta é caracterizada por um nome, um valor, um tipo, uma imagem de carta, e duas imagens padrão para a frente e verso da mesma.

3.2.1.7 Coleção

A coleção corresponde ao conjunto de todas as cartas existentes no jogo. É a partir da coleção que o utilizador pode gestão das cartas existentes, bem como do conteúdo de cada baralho.

3.2.1.8 Baralho

O baralho refere-se ao conjunto de cartas a ser utilizado durante o jogo. O baralho pode ser constituído por todas as cartas existentes ou parte delas. Cada jogador pode ter um baralho, ou o baralho pode ser partilhado entre os jogadores.

3.2.1.9 Campo de jogo

O campo de jogo é responsável por manter representação de todas as entidades intrínsecas ao uso do jogo, correspondendo assim, à área de jogo.

3.3 Design

3.3.1 Arquitetura

A definição da arquitetura de um sistema é sempre uma questão importante a considerar, uma vez que esta escolha influência todo o processo de desenvolvimento do software em questão. A escolha de uma arquitetura deve refletir o tipo de solução bem como o tempo de vida expectável do sistema. Os custos de desenvolvimento de software são um dos muitos fatores que influenciam a longevidade de um sistema. Avaliar os custos de desenvolvimento de um sistema é uma tarefa complexa, dada a necessidade de considerar vários fatores envolventes. Alguns dos fatores mais comuns são o tempo de desenvolvimento, complexidade do sistema, escalabilidade do sistema e integração com outros serviços. Uma maior necessidade de implementar estes requisitos reflete uma maior complexidade do sistema. Consequentemente, uma maior complexidade do sistema envolve maiores gastos na manutenção do sistema desenvolvido. Este é um problema comum na indústria de desenvolvimento de software. De facto, mais de metade dos custos associados ao desenvolvimento de software residem na área da manutenção. Os custos na fase de manutenção de produto podem ser parcialmente mitigados com uma definição de uma arquitetura coerente, que facilite a edição e correção do sistema numa fase posterior. De modo a permitir uma maior longevidade e manutenibilidade do sistema, o presente documento tenta seguir uma arquitetura que permita responder ao problema enunciado.

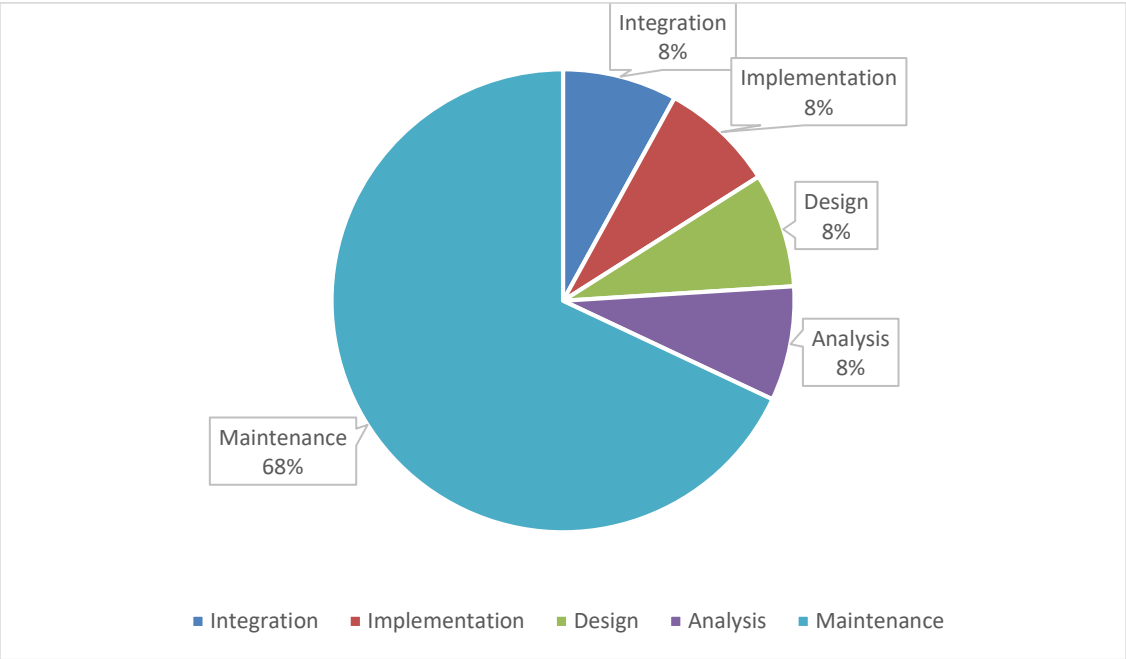

Figura 26 - Gráfico circular com as percentagens dos custos associados ao processo de desenvolvimento de software em ambiente empresarial (teacheramccarthymasco, 2012)

Para o presente projeto, foi considerada uma arquitetura baseada em componentes. A arquitetura baseada em componentes permite dividir o sistema em pequenos componentes individuais. Esta abordagem permite segregar as responsabilidades de cada componente, oferecendo assim um maior nível de abstração para com o sistema. O principal objetivo desta arquitetura passa pela reutilização de código. Ao dividir sistema em pequenos problemas individuais, com comportamentos específicos e bem definidos, os componentes podem ser facilmente reutilizados ou trocados por outros.

As principais características dos componentes são:

- **Modularidade**: Os componentes podem ser facilmente substituídos por outros e são não dependem de outros para o seu funcionamento.
- **Extensibilidade**: Um componente pode ser estendido de modo a permitir novos comportamentos ou funcionalidades.
- **Reutilização**: Os componentes podem ser reutilizados em diferentes contextos e aplicações.
- **Encapsulamento**: Cada componente implementa uma serie de interfaces comuns que permitem outros componentes fazerem uso das suas funcionalidades sem expor os detalhes de implementação.

A arquitetura baseada em componentes é uma arquitetura simples que permite uma divisão das responsabilidades em pequenas entidades individuais. No entanto esta abordagem não é suficiente para explicar a interação entre os componentes. É necessário introduzir outros padrões para unificar a solução desejada.

A arquitetura em camadas ou *Layered Architecture* é uma arquitetura caracterizada pela divisão das responsabilidades do sistema em diferentes áreas (camadas). Cada camada tem apenas uma responsabilidade, estando assim separada das restantes. Um dos grandes benefícios deste tipo de arquitetura é a modularidade do software. Este tipo de arquitetura permite que o software seja facilmente alterado, o que contribui para a sua manutenibilidade e testabilidade do mesmo. Apesar de exigir um custo maior no início do seu desenvolvimento, este tipo de arquitetura destaca-se pelas suas vantagens a longo prazo.

A arquitetura em camadas apresenta também alguns problemas. A divisão convencional em várias camadas não é adequada para refletir a necessidade de interação do utilizador com o sistema. Da mesma forma, este tipo de arquitetura requer a utilização de vários processos que passam por várias camadas para realizar tarefas simples, o que não são adequadas a projetos de sistemas gráficos, onde existe a necessidade de uma maior eficiência de código.

Ambas as soluções apresentam problemas, no entanto, é possível utilizar características de ambas as abordagens de forma a obter uma arquitetura mais robusta. Deste modo, foram definidas várias camadas de modo a permitir uma divisão do software em responsabilidades. Definir componentes também permite reutilizar certas entidades do jogo, o que contribui para uma maior estabilidade e coesão do mesmo.

Para o projeto foram definidas as seguintes camadas:

Camada de apresentação – A camada de apresentação representa a componente que interage com o utilizador. Esta camada é responsável por capturar os *inputs* do utilizador, mostrar informação e manipular certos tipos de informação.

Camada de aplicação - A camada de aplicação é responsável por ser a ponte entre a interface e o modelo de dados, bem, responsável por coordenar todas as ações despoletadas pelo utilizador.

Camada de domínio – A camada de domínio representa o modelo de dados da aplicação. Esta camada representa a informação do negócio, mas não é responsável pela manipulação ou comportamento da mesma. Um exemplo de uma entidade que pertence a esta camada é a entidade Carta. A Carta é apenas responsável por guardar a informação relativa a esta entidade, estando assim segregada do modo de como é aplicada no jogo.

Camada de serviços – Responsável por funcionalidades inerentes da aplicação, mas que não representam nenhum conceito do negócio (por exemplo manipulação ficheiros, gestão de cenários).

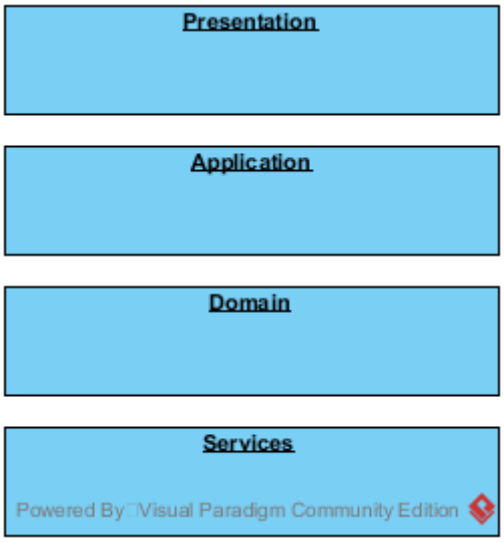

Figura 27 - Diagrama com as diferentes camadas da aplicação

Em relação à arquitetura baseada em componentes, neste projeto, um componente é uma representação gráfica de uma entidade do modelo de domínio. De forma a ser coerente com a divisão de responsabilidades em camada, cada componente contem uma referência para a entidade que está a representar. Quando uma ação é despoletada, continua a ser responsabilidade da camada de aplicação de produzir o fluxo esperado.

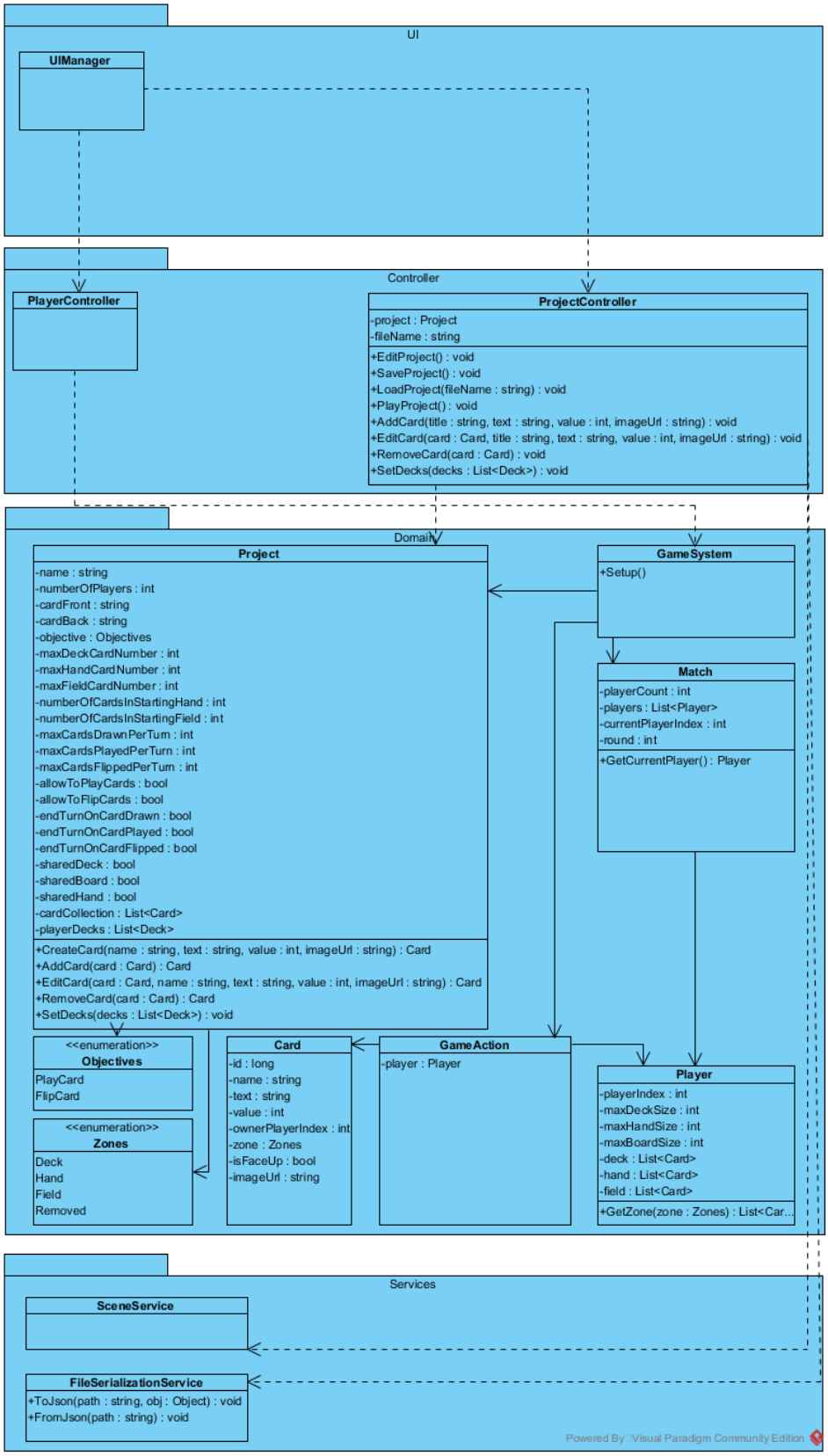

Figura 28 - Diagrama de Classes do projeto (Classes, propriedades e operações relacionados com a plataforma de desenvolvimento foram omitidas)

3.4 Aplicação

A presente secção está dividida em funcionalidades de modo a demonstrar os detalhes de implementação e modo de operação do sistema desenvolvido

3.4.1 Funcionamento da aplicação

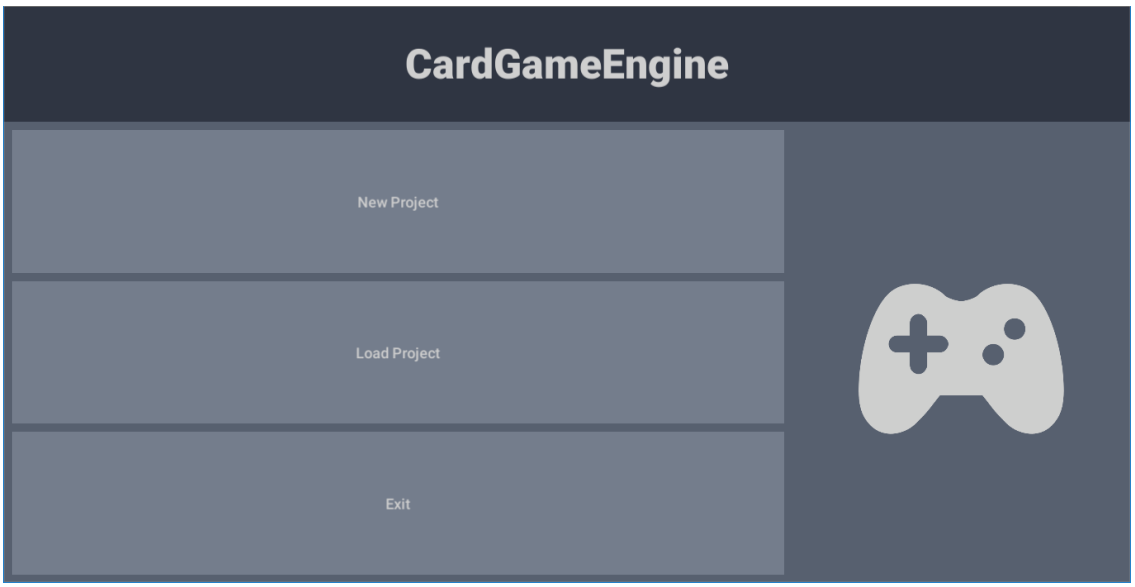

Figura 29 - Menu principal da aplicação

3.4.1.1 Tecnologias

Para criar a ferramenta proposta, foi escolhida a plataforma de desenvolvimento Unity. Tal como foi referido anteriormente, a plataforma Unity é considerada uma das mais conceituadas plataforma de desenvolvimento de software orientado para sistemas gráficos, uma vez que contem diversas tecnologias e ferramentas incorporadas que auxiliam o desenvolvimento deste tipo de sistemas. (Unity Technologies, 2018)

A linguagem de programação utilizada nesta plataforma de desenvolvimento é outra razão para a escolha do Unity. O C# é a principal linguagem de programação utilizada no desenvolvimento de software no ambiente do Unity. Esta linguagem de programação tem ganho cada vez mais popularidade na indústria dado que possui uma vasta gama de funcionalidades consideradas modernas. Algumas das principais características que tornam esta linguagem apelativa para o desenvolvimento de software são:

- Extensa documentação existente
- Gestão de memória automática
- Gestão de bibliotecas e recursos

- LINQ: *Language-Integrated Query*
- Programação assíncrona.
- Genéricos
- Multiplataforma (utilizada em Web, *Desktop*, Jogos…)
- Biblioteca nativa com funções comuns (por exemplo manipulação de *strings*)

3.4.1.2 Definição do jogo

Para desenvolver um jogo de cartas na plataforma desenvolvida, é necessário definir um conjunto de dados que caracterizam o próprio jogo. No contexto deste projeto, existe uma entidade, denominada de projeto, que agrega todos os estes valores. Esta abordagem permite que todos os valores sejam facilmente guardados e manipulados por parte da aplicação, durante a fase de conceção do jogo. Para esta iteração do produto, foram consideradas as seguintes características e propriedades.

- Nome do jogo/projeto.
- Número de jogadores.
- Objetivo principal do jogo, ou seja, como os pontos são acumulados (por exemplo, se os pontos são acumulados ao jogar as cartas, como no caso da bisca, ou se são acumulados ao virar pares de cartas).
- Número máximo de cartas por baralho.
- Número máximo de cartas na mão.
- Número máximo de cartas no campo.
- Número de cartas inicial na mão.
- Número de cartas inicial no campo.
- Número de cartas que é retirado do baralho por ronda.
- Número de cartas que é permitido jogar por ronda.
- Número de cartas que é permitido virar por ronda.
- Se o jogo permite retirar cartas do baralho.
- Se o jogo permite jogar cartas a partir da mão.
- Se o jogo permite virar cartas.
- Se o jogo deve mudar automaticamente de ronda no caso de uma carta ser retirada do baralho.
- Se o jogo deve mudar automaticamente de ronda no caso de uma carta ser jogada.
- Se o jogo deve mudar automaticamente de ronda no caso de uma carta ser virada.
- Se o jogo deve ter um único baralho, partilhado por todos os jogadores.
- Se o jogo deve ter uma única mão de jogo, partilhada por todos os jogadores.
- Se o jogo deve ter um único campo de jogo, partilhado por todos os jogadores.
- Uma imagem que serve como padrão da frente da carta de jogo.
- Uma imagem que serve como padrão do verso da carta de jogo.
- A coleção de cartas criadas.
- Os baralhos de jogo.

3.4.1.3 Persistência dos dados

O sistema desenvolvido permite guardar os jogos desenvolvidos em armazenamento local. Os jogos são guardados num ficheiro em formato JSON, o que permite ao utilizador editar os seus jogos dentro da aplicação ou com um editor de texto. Este ficheiro contem uma representação das características e propriedades mencionadas no ponto anterior.

```
{
    "name": "",
    "numberOfPlayers": 0,
    "cardFront": "",
    "cardBack": "",
    "objective": 0,
    "maxDeckCardNumber": 0,
    "maxHandCardNumber": 0,
    "maxFieldCardNumber": 0,
    "numberOfCardsInStartingHand": 0,
    "numberOfSummonsInTheField": 0,
    "maxCardsDrawnPerTurn": 0,
    "maxCardsPlayedPerTurn": 0,
    "maxCardsFlippedPerTurn": 0,
    "allowToDrawCards": false,
    "allowToPlayCards": false,
    "allowToFlipCards": false,
    "endTurnOnCardDrawn": false,
    "endTurnOnCardPlayed": false,
    "endTurnOnCardFlipped": false,
    "sharedDeck": false,
    "sharedBoard": false,
    "sharedHand": false,
    "cardCollection": [
        {
            "id": 0,
            "name": "",
            "text": "",
            "value": 0,
            "imageUrl": ""
        }
    ],
    "playerDecks": [
        {
            "deckList": [
                {
                    "id": 0,
                    "name": "",
                    "text": "",
                    "value": 0,
                    "imageUrl": ""
                },
            ]
        }
    ]
}
```


Figura 30 - Formato do ficheiro JSON utilizado na persistência dos dados da aplicação

3.4.1.4 Gestão de cenários

O projeto desenvolvido faz uso do sistema de cenários incorporado do Unity. Os cenários operam de forma independente, como se fossem uma aplicação diferente. Cada cenário contem os seus respetivos menus, ambientes e componentes.

O presente projeto contem dois cenários. O cenário de desenvolvimento do jogo e a cenário onde é possível jogar o jogo desenvolvido. Tal como o nome indica, o cenário de desenvolvimento do jogo é responsável por todos os dados introduzidos pelo utilizador relacionados com a customização e definição do jogo. Como já foi referido anteriormente, isto facilita a agregação de todos os dados numa entidade única.

Para o cenário do jogo, o sistema mantem uma referência dos dados do jogo do cenário anterior. Quando o cenário de jogo é carregado, o jogo é construído com base nos dados existentes nesta referência.

3.4.1.5 Adversário

De forma a simular os adversários que poderão existir nos jogos desenvolvidos, o sistema implementa um conjunto de algoritmos que permitem interagir com as mecânicas e regras existentes. Desta forma os adversários gerados adaptam-se ao jogo em si, quer seja um jogo de memória ou jogo de cartas tradicional.

3.4.2 Aplicação e funcionalidades

3.4.2.1 Criar um novo jogo

O primeiro passo consiste em criar um novo projeto/jogo. O projeto contem todas as definições de das regras, características, e conteúdo do jogo. O utilizador deve introduzir os valores necessários de forma a criar o seu próprio jogo.

A criação e edição de cartas, bem como dos baralhos de jogo, não se enquadram neste de caso de uso, uma vez que a manipulação destas entidades é um processo mais complexo, que merece o seu próprio caso de uso. De qualquer foram, é de salientar estas propriedades, uma vez que pertencem ao conjunto de dados agregados pelo jogo.

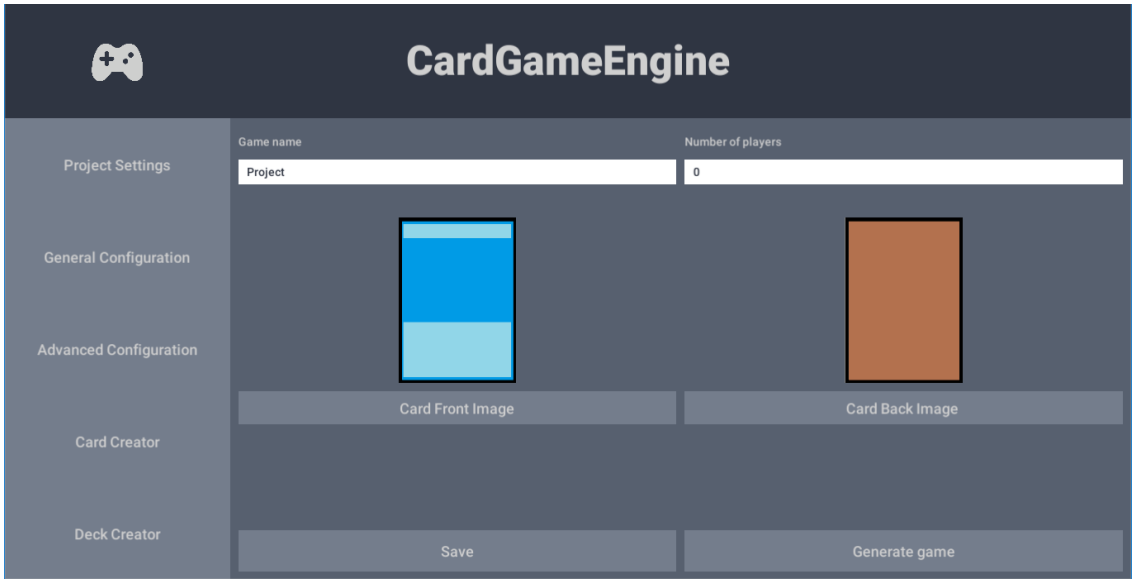

Figura 31 - Interface gráfica: Criar um novo jogo

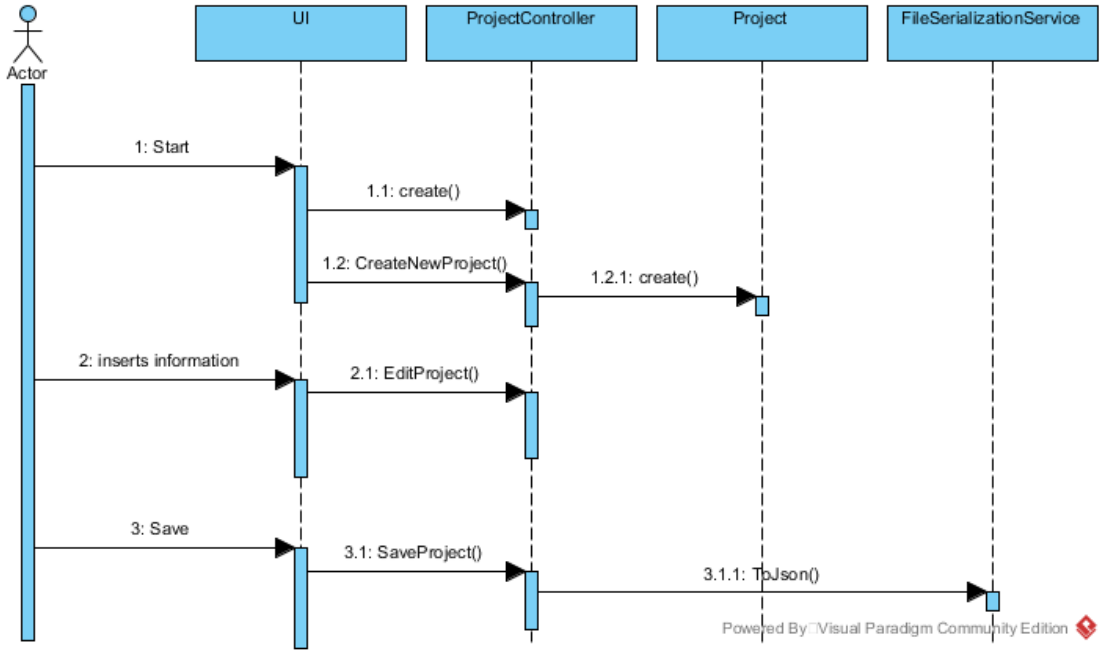

Figura 32 - Diagrama de sequência: Criar um novo jogo

3.4.2.2 Editar jogo

A ação de editar um jogo existente segue o mesmo princípio do caso de uso anterior. Tal, como foi referido no caso de uso anterior, a entidade projeto contem todas as definições de das regras, características, e conteúdo do jogo, de modo a que o utilizador possa facilmente edita-las. A única diferença deve-se ao facto de o utilizador necessitar de carregar para memoria um ficheiro com o conteúdo do jogo.

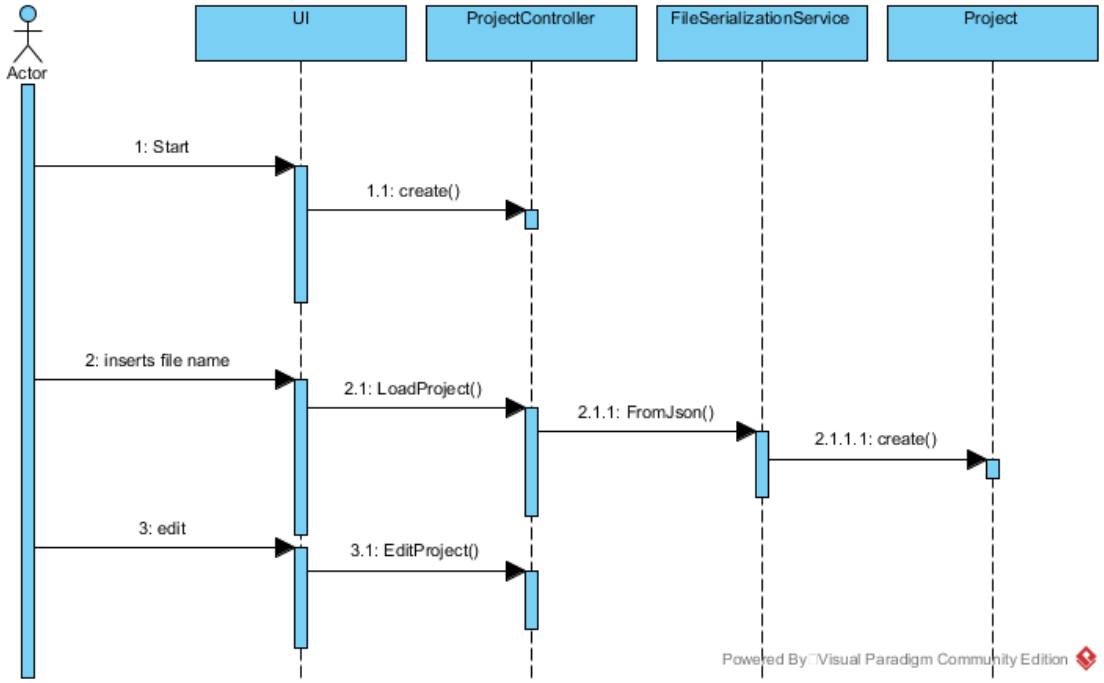

Figura 33 - Diagrama de sequência: Editar um jogo

3.4.2.3 Criar uma nova carta de jogo

Como em qualquer jogo de cartas existente, o processo de criação e design de cartas é um aspeto fundamental para o bom funcionamento do jogo. Deste modo, foi desenvolvido um sistema que permite o desenvolvimento rápido de cartas de jogo.

Para além de ter um projeto base já criado, o utilizador deve inserir a informação relativa à carta. Tal como já foi referido anteriormente, uma carta é constituída por:

- Nome
- Valor
- Tipo
- Imagem de carta
- Imagem padrão para a frente da carta
- Imagem padrão para o verso da carta

As imagens padrão fazem parte da constituição da carta, no entanto estas são globais para todas as cartas, ou seja, a alteração da definição de um dos padrões da carta implica uma mudança em todas as cartas.

A interface desenvolvida é simples, intuitiva e permite ver as modificações que acontecem na carta à medida que ela é criada. No mesmo ecrã existe uma área onde são mostradas todas as cartas já existentes, de forma a auxiliar o utilizador no processo.

No momento em que uma carta é guardada, a carta é adicionada à coleção, e é criada uma referência na interface que permite visualização ou edição da mesma

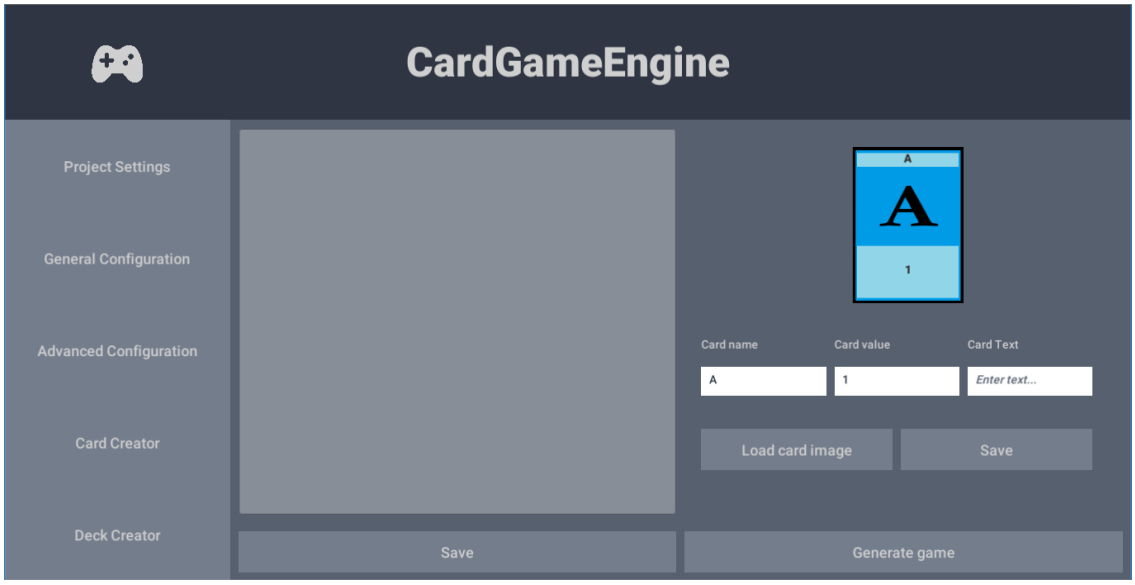

Figura 34 - Interface gráfica: Criar uma nova carta de jogo

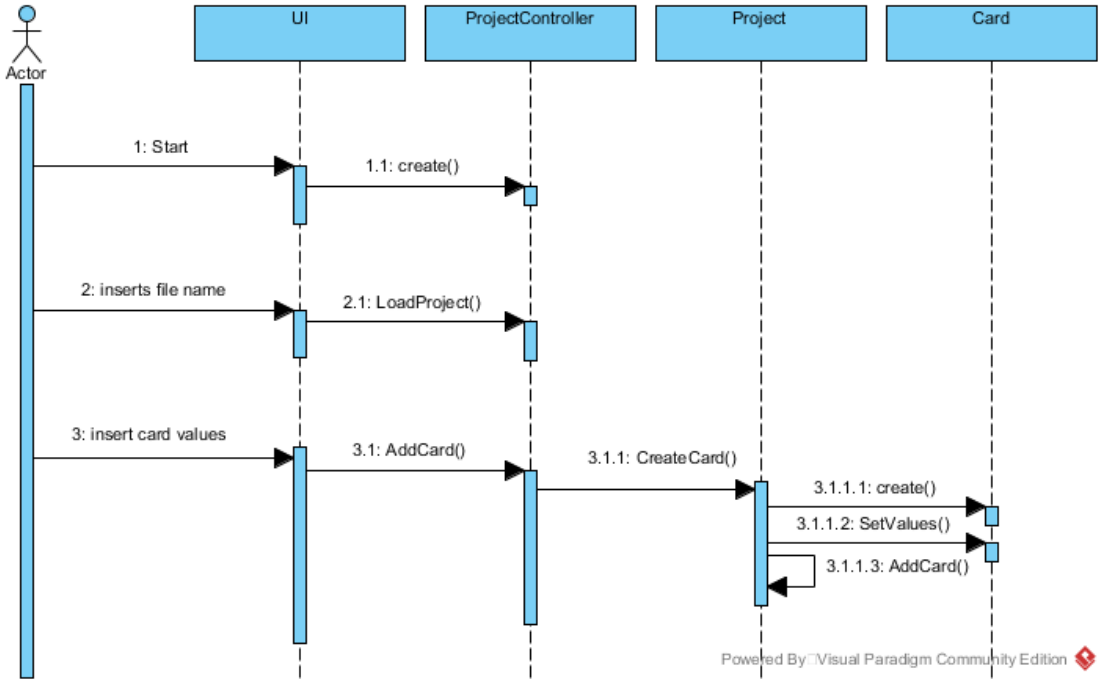

Figura 35 - Diagrama de sequência: Criar uma nova carta de jogo

3.4.2.4 Editar uma carta de jogo

Editar uma carta de jogo é idêntico ao caso de uso anterior. O utilizador deve selecionar uma carta existente da coleção e alterar todos os parâmetros que necessitam de ser editados.

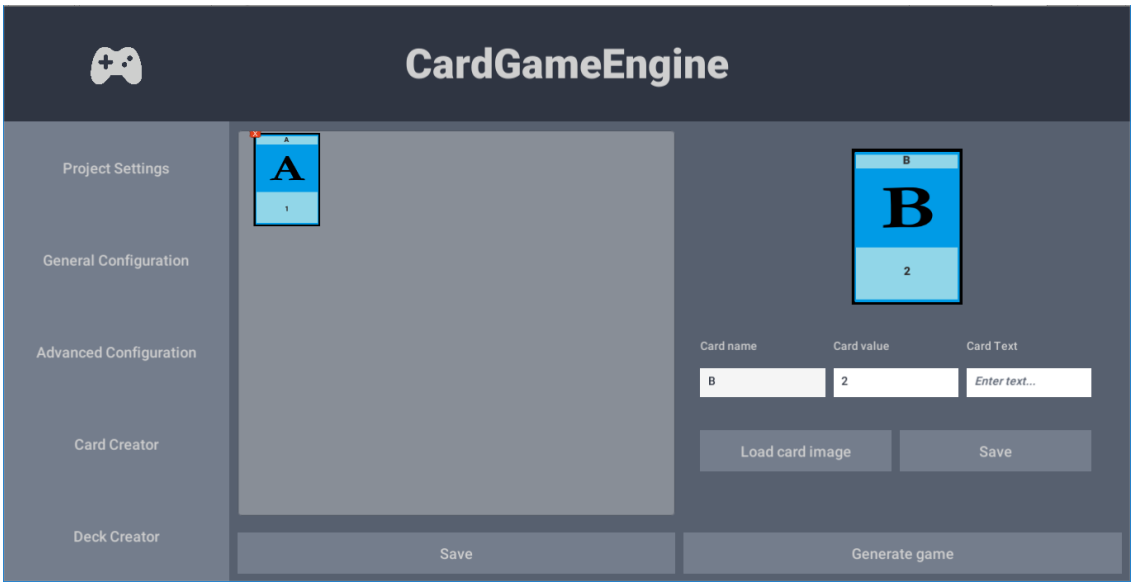

Figura 36 - Interface gráfica: Editar uma carta de jogo

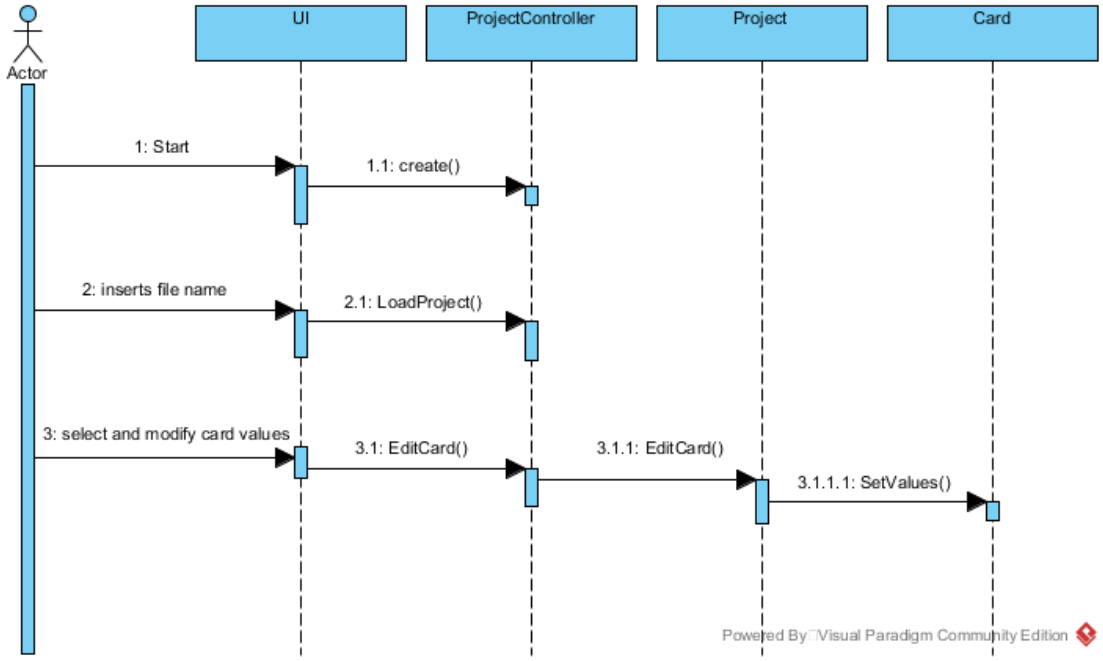

Figura 37 - Diagrama de sequência: Editar uma carta de jogo

3.4.2.5 Remover carta de jogo

O utilizador pode também, apagar as cartas que foram adicionadas previamente, uma vez que o processo de criação de cartas é propenso a erros e a mudanças. Para apagar uma carta da coleção o utilizador tem de carregar no botão vermelhão assinalado com uma cruz.

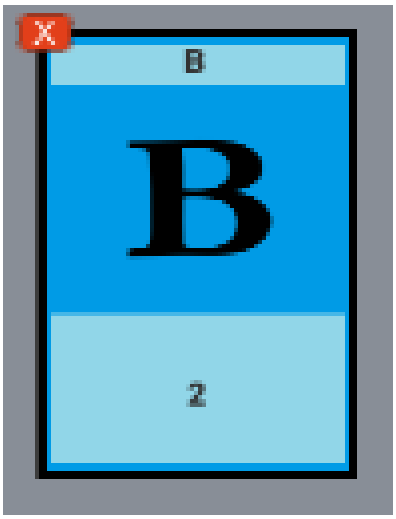

Figura 38 – Interface gráfica: Remover uma carta de jogo

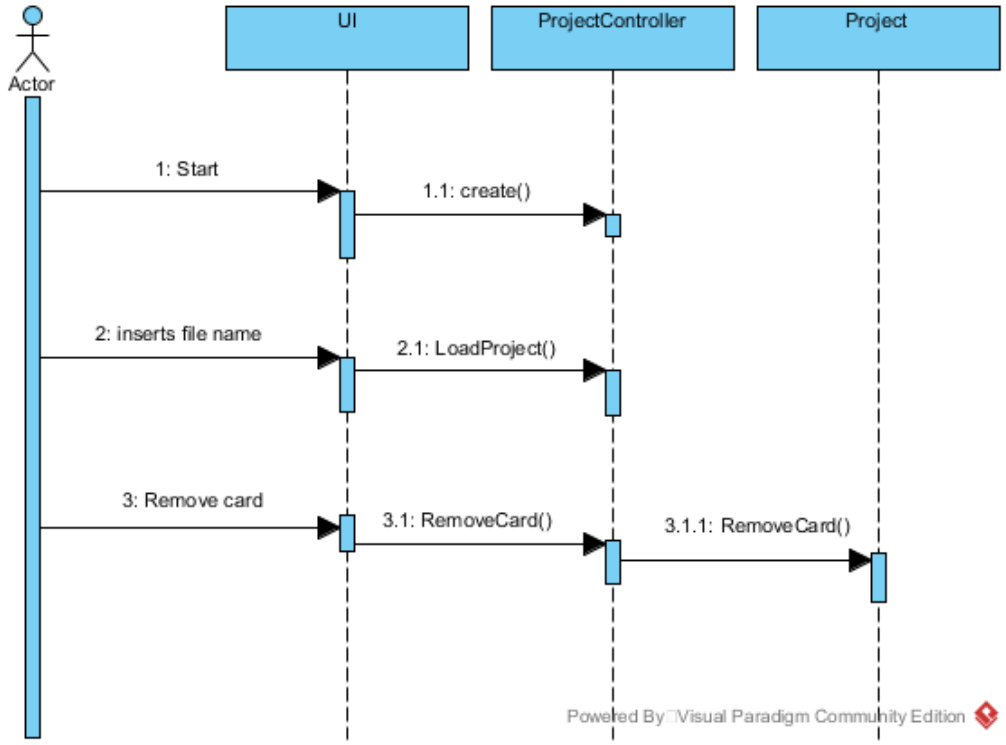

Figura 39 - Diagrama de sequência: Remover uma carta de jogo

3.4.2.6 Editar baralhos de jogo

Para poder jogar com as cartas, é necessário configurar os baralhos de cartas. A interface implementa um sistema de *drag-and-drop*, que permite ao utilizador editar os baralhos com bastante facilidade. Deste modo o utilizador apenas tem de arrastar uma carta da coleção para área designada do baralho do respetivo jogador.

O lado esquerdo contem espaços para as cartas a incluir dos baralhos, enquanto que o lado direito contem todas as cartas criadas. O número de baralhos que é possível criar, bem como o número de cartas que é possível de adicionar ao baralho está dependente da configuração que o utilizador escolheu durante a criação do projeto.

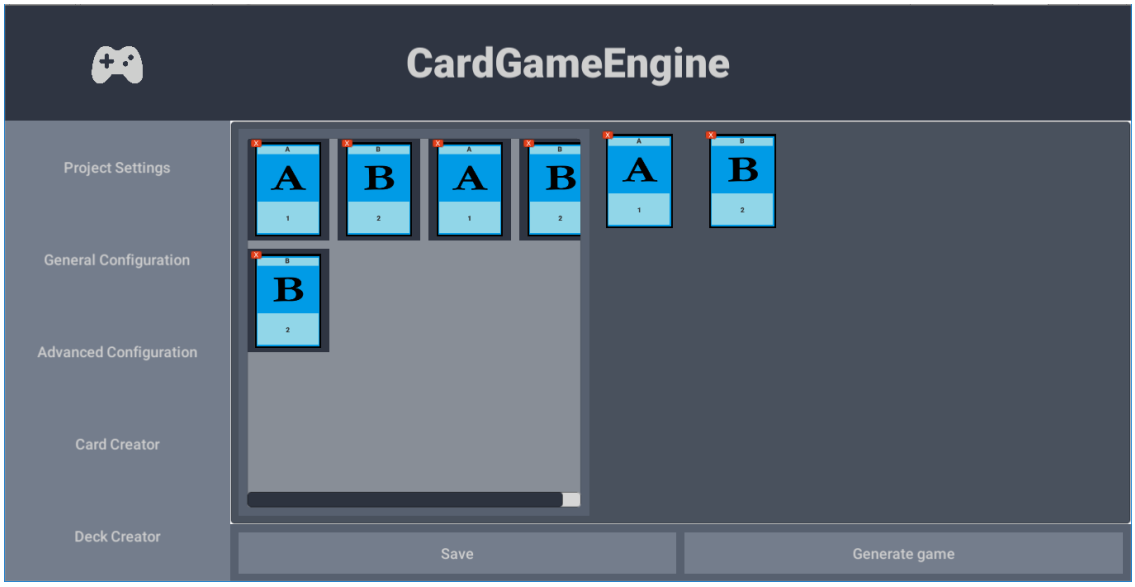

Figura 40 - Interface gráfica: Editar baralhos de jogo

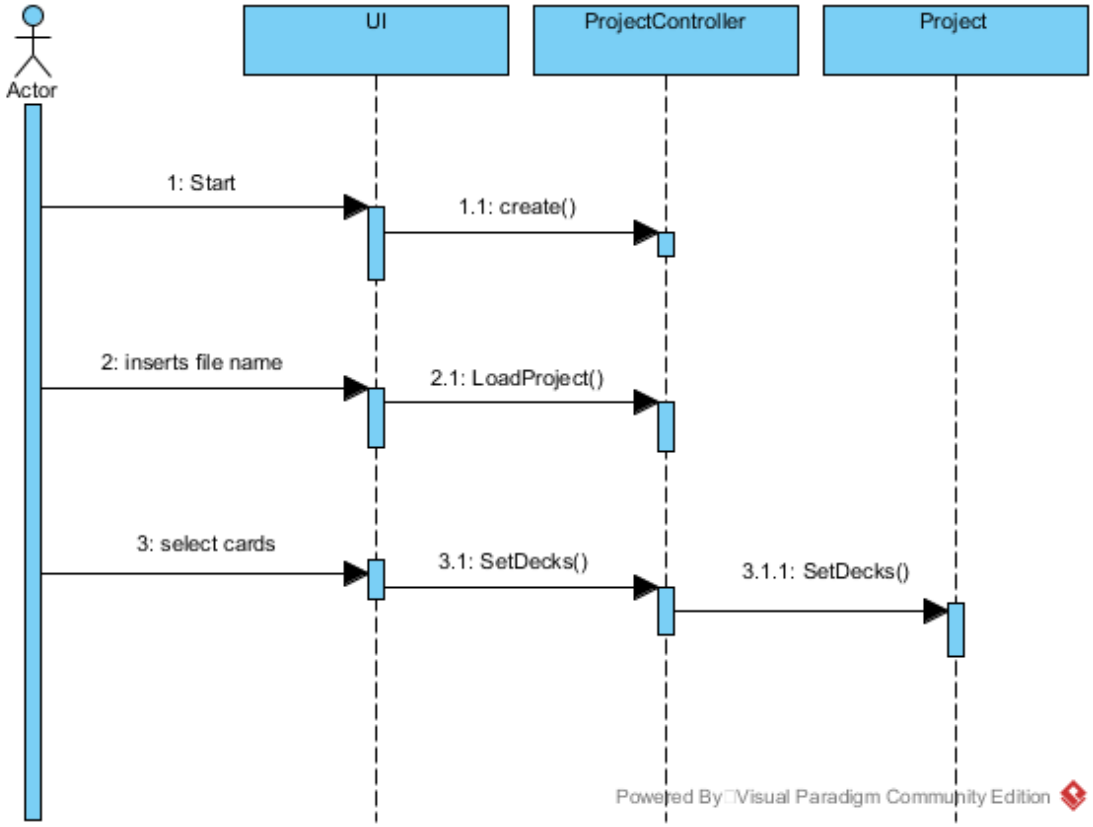

Figura 41 - Diagrama de sequência: Editar baralhos de jogo

3.4.2.7 Jogar o jogo desenvolvido

Jogar e experimentar o jogo desenvolvido é principal objetivo da aplicação desenvolvida. Como foi referido no ponto anterior, após o utilizador configurar ou carregar o conteúdo de um projeto é possível visualizar e jogar uma representação gráfica do jogo. Este cenário do projeto é constituído por um controlador cuja responsabilidade é gerir e delegar todas ações presentes no jogo.

Diferentes configurações geram diferentes entidades no cenário de jogo. Por exemplo, um jogo de memória é caracterizado por ter todas as cartas viradas para baixo no início do jogo.

O sistema contem várias animações de ações que contribuem para uma melhor interação com o utilizador. Nesta fase, o sistema contem as seguintes animações de jogo:

- Mudar de turno de jogo
- Retirar uma carta do baralho
- Construir mão de jogo
- Ver uma carta na mão
- Jogar uma carta na mão
- Rodar uma carta de jogo

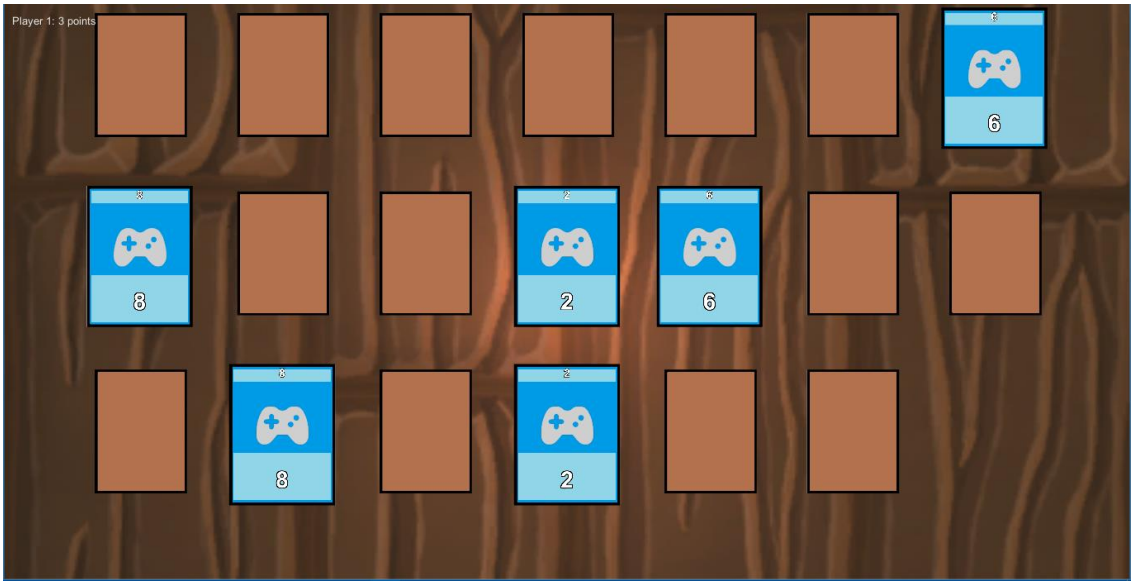

Figura 42 – Interface gráfica: Jogar o jogo desenvolvido – Exemplo de jogo de memória

Outros tipos de jogos poderão incluir diferentes mecânicas e entidades. Por exemplo, um jogo pode incluir baralhos, mão de jogo, cartas em campo, entre outros elementos.

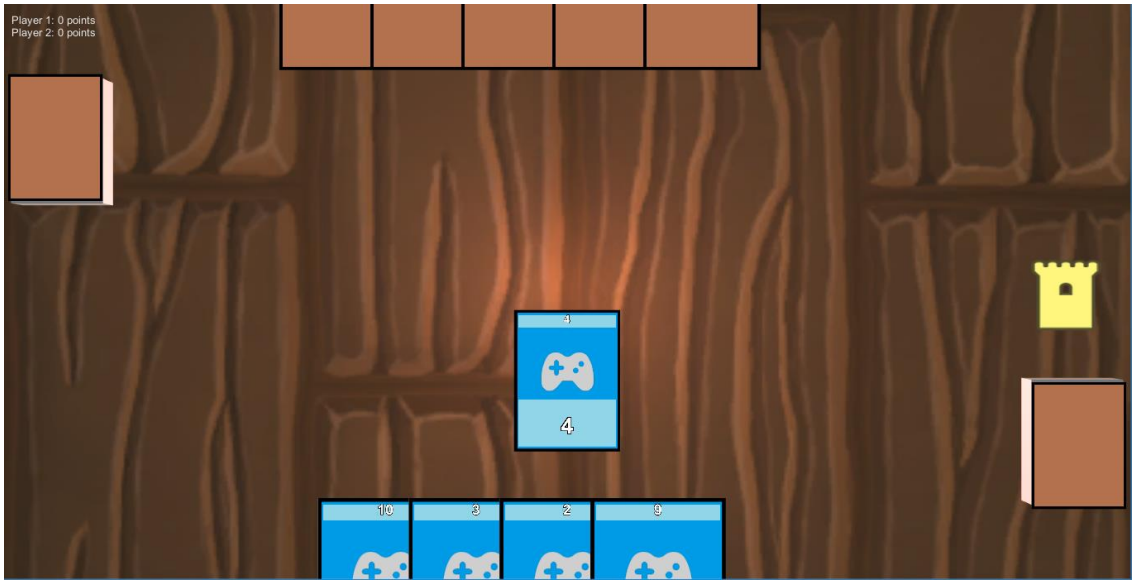

Figura 43 – Interface gráfica: Jogar o jogo desenvolvido – Exemplo de jogo de cartas com baralho e mão de jogo

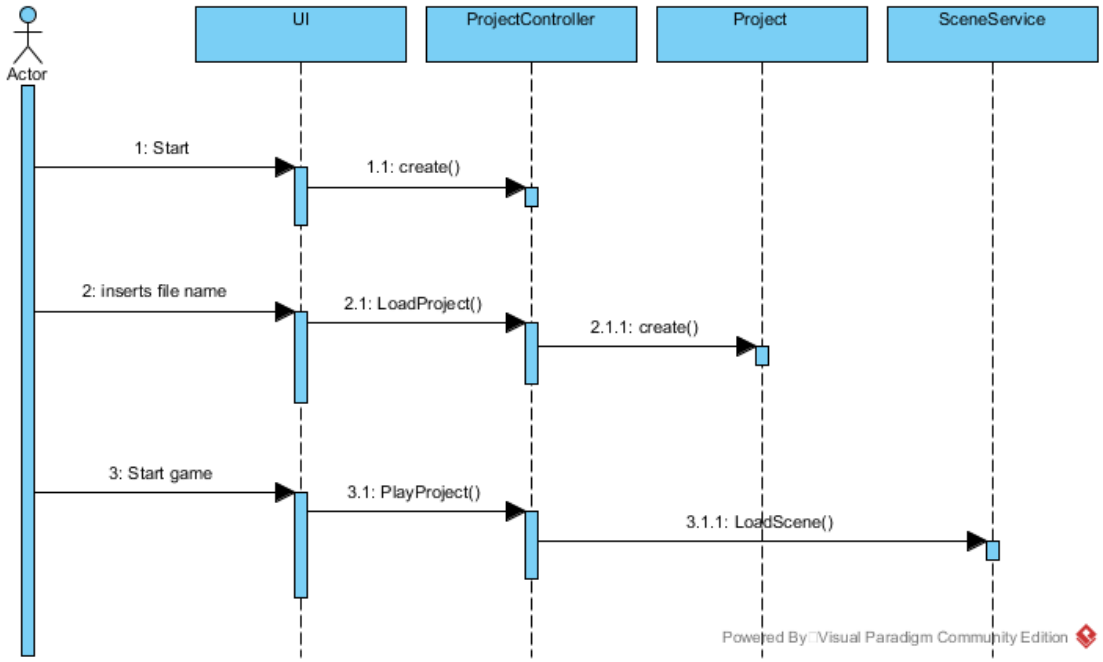

Figura 44 – Diagrama de sequência: Jogar o jogo desenvolvido

3.5 Testes e Avaliação

Esta secção aborda os testes realizados de modo a verificar a fiabilidade e qualidade do produto desenvolvido

3.5.1 Testes Unitários

Os testes unitários são um tipo de teste utilizado para determinada a fiabilidade do código da aplicação. O principal propósito deste tipo de teste é verificar a integridade de métodos e funções base do código de um modo isolado.

Para realizar este tipo de testes foi utilizada a biblioteca MSTest. Estes tipos de testes foram aplicados às classes da camada de domínio e aplicação do projeto.

```
[TestClass]
    public class ProjectTests
    {
        public readonly string TEST_TITLE       = "testTitle";
        public readonly string TEST_TEXT        = "testText";
        public readonly int    TEST_VALUE       = 0;
        public readonly string TEST_IMAGE_URL   = "testUrl";

        public Project testProject;
        public Card card;

        [TestInitialize()]
        public void Initialize()
        {
            testProject = new Project();
            card = testProject.CreateCard(TEST_TITLE, TEST_TEXT, 0,
TEST_IMAGE_URL);
        }
```

Figura 45 – Exemplo de teste unitários realizados para a classe Project

```
        [TestMethod]
        public void TestCreateCard()
        {
            Assert.IsNotNull(card);
            Assert.IsNotNull(card.id);
            Assert.AreEqual(card.name,TEST_TITLE);
            Assert.AreEqual(card.text, TEST_TEXT);
            Assert.AreEqual(card.value, TEST_VALUE);
            Assert.AreEqual(card.imageUrl, TEST_IMAGE_URL);
        }

        [TestMethod]
        public void TestAddCard()
        {
            testProject.AddCard(card);
            Assert.AreEqual(testProject.cardCollection.Contains(card),
true);
        }

        [TestMethod]
        public void TestEditCard()
        {
            string newTestTitle     = "newTestTitle";
            string newTestText      = "newTestText";
            int    newTestValue     = 1;
            string newTestImageUrl  = "newTestUrl";
            testProject.EditCard(card, newTestTitle, newTestText,
newTestValue, newTestImageUrl);
            Assert.AreEqual(card.name, newTestTitle);
            Assert.AreEqual(card.text, newTestText);
            Assert.AreEqual(card.value, newTestValue);
            Assert.AreEqual(card.imageUrl, newTestImageUrl);
        }

        [TestMethod]
        public void RemoveCard()
        {
            testProject.RemoveCard(card);
            Assert.AreEqual(testProject.cardCollection.Contains(card),
false);
        }
```

Figura 46 – Exemplo de testes realizados para a classe Project

3.5.2 Avaliação de qualidade

De forma a avaliar o projeto desenvolvido, foi elaborado um plano de avaliação baseado no modelo QEF. O modelo QEF ou Quantitative Evaluation Framework permite realizar uma avaliação quantitativa e qualitativa de software educativo. O modelo QEF propõem uma divisão dos requisitos de avaliação em diferentes dimensões, consoante a sua natureza. Cada dimensão é dividida em fatores, que se focam em aspetos mais concretos a avaliar. Cada fator é caracterizado por uma lista de requisitos não funcionais que o compõem.

Quanto à componente quantitativa, a cada dimensão, fator e requisito é associado um valor percentual que avalia o cumprimento da tarefa. O valor associado à dimensão é influenciado pelo valor do fator, que por sua vez é influenciado pelo valor do requisito. O resultado final é uma percentagem que reflete os resultados atingidos pelo software desenvolvido.

É importante referir que o projeto desenvolvido em si não é um software educacional. No entanto os resultados esperados (jogos sérios) enquadram-se na componente de software educacional. Desta forma o modelo QEF desenvolvido é baseado nestas duas entidades.

Para o presente projeto foram escolhidas três dimensões: **Funcionalidades**, **Área pedagógica** e **Usabilidade**. Estas três áreas foram escolhidas como dimensões a avaliar uma vez que permitem abordar os principais objetivos propostos para este projeto.

- A dimensão **Funcionalidades** permite avaliar de forma objetiva e tecnológica o produto desenvolvido
- A **Área pedagógica** enquadra-se nos fatores de ensino que o projeto pretende abordar
- A **Usabilidade** permite avaliar a interação com o sistema em si. Para o utilizador comum a interface dita a qualidade de um sistema.

Outra dimensão que foi considerada, mas que não foi utilizada na fase final foi a perspetiva técnica. A perspetiva técnica é uma dimensão bastante utilizada neste modelo de avaliação, no entanto, não foi considerada uma vez que os requisitos não se enquadram com os objetivos principais do documento, uma vez que a necessidade de escalar o produto ainda não chegou a uma maturação necessária para avaliar esses critérios

As métricas de avaliação foram classificadas segundo a seguinte natureza:

1. **Funcionalidades**
    1.1. Desenvolvimento do jogo
        1.1.1. UC1 - Criar novo jogo
        1.1.2. UC2 - Editar jogo
        1.1.3. UC3 - Criar nova carta de jogo
        1.1.4. UC4 - Editar carta de jogo
        1.1.5. UC5 - Remover carta de jogo
        1.1.6. UC6 - Editar baralhos de jogo
    1.2. Jogo
        1.2.1. UC7 – Jogar jogo desenvolvido
2. **Área Pedagógica**
    2.1. Aprendizagem
        2.1.1. O contexto de aprendizagem é apropriado
        2.1.2. O produto adapta-se a várias a diferentes metodologias pedagógicas
        2.1.3. Terminologia utilizada é correta e válida
    2.2. Avaliação
        2.2.1. A pontuação do jogo relaciona-se com os métodos de avaliação
3. **Usabilidade**
    3.1. Menu de jogo
        3.1.1. A interface é simples e intuitiva.
        3.1.2. A interface está bem estruturada
        3.1.3. O utilizador consegue facilmente entrar e sair da aplicação
    3.2. Jogo
        3.2.1. O jogo contem áudio que caracteriza a experiência de jogo
        3.2.2. O jogo é dotado de imagens e gráficos que caracterizam a experiência de jogo
        3.2.3. Permite alterar a dificuldade de jogo
        3.2.4. A velocidade das interações entre o jogador e o sistema é adequada
        3.2.5. A interação com o jogo é intuitiva
        3.2.6. A informação relativa ao jogo é visível
        3.2.7. As ações de jogo contêm animações visuais adequadas

Por sua vez, cada fator foi caracterizado consoante o grau de satisfação do requisito que o compõe

Factor 1.1 Game Development		Wfk - Fulfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
1.1.1 UC1 - Create a new game	The user can create and develop a new game	Not implemented		File manipulation		Implemented
1.1.2 UC2 - Edit a game	The user can edit and develop an existing game	Not implemented		File manipulation		Implemented
1.1.3 UC3 - Create a new card	The user can create and add a new card to the card collection	Not implemented		File manipulation		Implemented
1.1.4 UC4 - Edit a new card	The user can edit an existing card	Not implemented		File manipulation		Implemented
1.1.5 UC5 - Remove a card	The user can remove an existing card from the collection	Not implemented		File manipulation		Implemented
1.1.6 UC6 - Edit deck	The user can edit the decks	Not implemented		File manipulation		Interactive drag-and-drop system

Figura 47 - Grau de avaliação das métricas relacionadas com o fator: Desenvolvimento do jogo

Factor 1.2 Gameplay		Wfk - Fulfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
1.2.1 UC7 - Play the game	The user can play its game	Not implemented	Scene loading	Basic gameplay	Implemented with animations	Supports multiple types of games

Figura 48 - Grau de avaliação das métricas relacionadas com o fator: Jogo

Factor 2.1 Learning		Wfk - Fulfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
2.1.1 Learning context is appropriate	Learning context is appropriate	No				The games can be modified to meet the current needs
2.1.2 The system can be integrated in different pedagogical methodologies	The system can support multiple pedagogical methodologies	No				The games can be modified to meet the current needs
2.1.3 The technical and scientific terminology of the field being taught is correct and valid	The system allows valid and correct scientific terminology	No				Yes

Figura 49 - Grau de avaliação das métricas relacionadas com o fator: Aprendizagem

Factor 2.2 Assessment		Wfk - Fulfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
2.2.1 Learner assessment relates with game scoring	Learning assessment relates with game scoring	No		Somewhat related		Complex and multivariable assessment

Figura 50 - Grau de avaliação das métricas relacionadas com o fator: Avaliação

Factor 3.1 Menu Navigation		Wfk - Fulfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
3.1.1 User interface is simple and intuitive	Application is easy, simple and intuitive to use	Hard to navigate				Easy, simple and intuitive
3.1.2 User interface is coherently structured	All the menus have a similar structure.	Bad structure				Adequate structure
3.1.3 The player can easily start and quit the game	The player can easily and start and quit the game	No				Yes

Figura 51 - Grau de avaliação das métricas relacionadas com o fator: Menu de jogo

Factor 3.2 Gameplay		Wfk - Fullfilment (%)				
Requirement	**Metric Evaluation**	**0**	**25**	**50**	**75**	**100**
3.2.1 Audio usage enhances game play	Quality of the audio	No sounds		Adequate		Superb sound track
3.2.2 Graphics and pictures enhance game play	Quality of the graphics and pictures	No design		Adequate		Superb graphic design
3.2.3 Allow to change difficulty	The player can change the difficulty of the game	No				In-game difficulty setting
3.2.4 Speed of communication between the player and the game is adequate	Pace rate of the game	Awful				Adequate

3.2.5 The interaction with the game is intuitive	Important player actions have feedback to inform the player	No				Yes
3.2.6 Important information is visible and highlighted	Important information is visible and highlighted	No				Yes
3.2.7 Game actions contain adequate visual animations	Quality of the game animations	No animations		Adequate		Superb animations

Figura 52- Grau de avaliação das métricas relacionadas com o fator: Jogo

De forma a avaliar o projeto desenvolvido, foi elaborado o seguinte questionário para recolher informação sobre o mesmo. O questionário contem perguntas de carácter qualitativo de forma a melhor se enquadrarem com o modelo QEF desenvolvido. As perguntas foram aplicadas às dimensões **Área Pedagógica** e **Usabilidade** uma vez que são as áreas mais subjetivas de avaliar.

Dimensão	Fator	Questão	Valores possíveis
Área Pedagógica	Aprendizagem	Contexto de aprendizagem é apropriado?	Sim; Não
Área Pedagógica	Aprendizagem	O sistema suporta múltiplas metodologias pedagógicas?	Sim; Não
Área Pedagógica	Aprendizagem	O sistema permite o uso de terminologia científica correta e válida	Sim; Não
Área Pedagógica	Avaliação	A pontuação do jogo relaciona-se com os métodos de avaliação?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Menu de jogo	A aplicação é simples e intuitiva de usar?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Menu de jogo	A estrutura dos menus é idêntica	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Menu de jogo	O jogador consegue facilmente entrar e sair dos jogos?	Sim; Não
Usabilidade	Jogo	Qualidade do áudio	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Jogo	Qualidade das imagens e gráficos?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Jogo	O jogador consegue facilmente alterar a dificuldade do jogo?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota

Usabilidade	Jogo	A velocidade das interações entre o jogador e o sistema é adequada?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Jogo	A interação com o jogo é intuitiva?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Jogo	A informação relativa ao jogo é visível?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota
Usabilidade	Jogo	Qualidade das animações?	[1,2,3,4,5] em que 1 é a pior nota e 5 a melhor nota

Tabela 13 – Questionário com base no Modelo QEF desenvolvido

Do questionário resultaram 11 avaliações. O questionário focou-se principalmente num público constituído por colegas de curso e utilizadores de jogos de cartas. Os resultados foram convertidos e arredondados de forma a enquadrarem-se no estudo do grau de avaliação das diferentes métricas.

O sistema também foi alvo de outro tipo de comentários construtivos para novas funcionalidades que não contemplam o questionário desenvolvido.

- Mais imagens-padrão para frente e verso de cartas
- Permitir edição do campo de jogo
- Suporte para mais tipos de jogos
- Suporte multijogador

O resultado final da avaliação baseado no modelo QEF situa o valor de qualidade q à volta dos 90%, um valor bastante otimista para esta fase do projeto.

q	D	Qi	Dimension	Qj	Wij (Factor Weight j in Dim i) [0,1]	Factor	rwjk (requirement weight k in Factor j) {2, 4, 6, 8, 10}	Requirement	wfk % requirement fulfillment k) [0,100]
90%	0,30	100	1. Functionality	100	0,857	1.1 Game Development	10	1.1.1 UC1 - Create a new game	100
							10	1.1.2 UC2 - Edit a game	100
							10	1.1.3 UC3 - Create a new card	100
							10	1.1.4 UC4 - Edit a new card	100
							10	1.1.5 UC5 - Remove a card	100
							10	1.1.6 UC6 - Edit deck	100
				100	0,143	1.2 Gameplay	10	1.2.1 UC7 - Play the game	100
		87,5	2. Pedagogical	100	0,75	2.1 Learning	10	2.1.1 Learning context is appropriate	100
							10	2.1.2 The system can be integrated in different pedagogical methodologies	100
							10	2.1.3 The technical and scientific terminology of the field being taught is correct and valid	100
				50	0,25	2.2 Assessment	10	2.2.1 Learner assessment relates with game scoring	50
		72,5	3. Usability	100	0,30	3.1 Menu Navigation	10	3.1.1 User interface is simple and intuitive	100
							10	3.1.2 User interface is coherently structured	100
							10	3.1.3 The player can easily start and quit the game	100
				60,7143	0,70	3.2 Gameplay	10	3.2.1 Audio usage enhances game play	0
							10	3.2.2 Graphics and pictures enhance game play	50
							10	3.2.3 Allow to change difficulty	50
							10	3.2.4 Speed of communication between the player and the game is adequate	100
							10	3.2.5 The interaction with the game is intuitive	75
							10	3.2.6 Important information is visible and highlighted	100
							10	3.2.7 Game actions contain adequate visual animations	50

Figura 53 – Modelo QEF

4 Conclusão

Neste último capítulo é resumido o projeto realizado, com uma maior enfâse nos objetivos e tarefas realizadas, dificuldades encontradas, limitações do projeto e trabalhos futuros. Por fim este documento termina com uma nota pessoal do trabalho realizado.

4.1 Objetivos realizados

Em termos de funcionalidades, foram desenvolvidas todas as funcionalidades proposta para uma primeira iteração deste produto.

4.2 Limitações e trabalho futuro

A aplicação deve ser considerada como uma primeira iteração ou protótipo de um produto futuro a desenvolver. Aplicações semelhantes, desenvolvidas em contexto comercial, são mantidas por equipas constituídas por centenas de colaboradores de diversas áreas de estudo como engenharia, programação, design de interface, arte, inteligência artificial, entre muitos outros. É impensável comparar o trabalho realizado em contexto académico por uma pessoa com o trabalho com as empresas na vanguarda desta área.

Em relação a trabalhos futuros, é sempre possível expandir, melhorar ou modificar a aplicação, deste modo, destacam-se as principais funcionalidades a introduzir em posteriores versões:

- Suporte para mais mecânicas e regras de jogo, normalmente associadas a jogos de cartas digitais (por exemplo iteração entre cartas)
- Suporte para edição de mais propriedades gráficas do jogo (edição do campo de jogo, orientação das cartas, baralhos, entre outros.)

- Suporte multijogador (jogador contra jogador)
- Melhorar o algoritmo quando o jogo é do tipo jogador contra computador. O algoritmo atual é bastante simples e apenas serve para simular um utilizador. Uma alternativa mais interessante seria usar uma abordagem determinística como por exemplo uma implementação do algoritmo *minmax* de forma a encontrar as melhores jogadas possíveis. Outra abordagem interessante seria, por exemplo, utilizar *machine learning* para ensinar um agente a jogar os diversos tipos de jogos de cartas. Ambas as abordagens aqui descritas são extremamente complexas, que mereciam o seu próprio projeto individual. A implementação destas duas abordagens consumiria demasiados recursos no desenvolvimento deste primeiro protótipo e os benefícios provenientes não seriam tão fulcrais para esta fase de desenvolvimento.

4.3 Apreciação final

O projeto reflete o trabalho realizado por um estudante de engenharia informática no último ano do mestrado.

O projeto integra várias áreas como engenharia de software, programação em C#, sistemas gráficos, desenvolvimento de aplicações com motores de jogos, entre outros.

Em suma, tendo em conta a proposta do projeto, documentação produzida, produto desenvolvido, condições disponibilizadas e o feedback obtido considero que o projeto cumpriu os objetivos pretendidos.

Referências

Allee, V. (2008). Value network analysis and value conversion of tangible. *Journal of Intellectual Capital*.

Apiumhub. (10 de April de 2016). *The importance of a good software architecture*. Obtido de Apiumhub: https://apiumhub.com/tech-blog-barcelona/importance-good-software-architecture/

Bloomberg. (23 de January de 2019). *Peak Video Game? Top Analyst Sees Industry Slumping in 2019*. Obtido de Bloomberg: https://www.bloomberg.com/news/articles/2019-01-23/peak-video-game-top-analyst-sees-industry-slumping-in-2019

Coin-Drop. (30 de March de 2018). *The Trend to Publish Unfinished Games: A Blessing For Developers, Or a Curse for Players?* Obtido de Coin-Drop: http://coin-drop.com/trend-publish-unfinished-games/

Epic Games. (s.d.). *Make something Unreal with the most powerful creation engine*. Obtido de Epic Games: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

Eurogamer. (1 de September de 2018). *Are RPG Maker games as bad as people think?* Obtido de Eurogamer: https://www.eurogamer.net/articles/2018-09-01-are-rpg-maker-games-as-bad-as-people-think

Galyonkin, S. (8 de April de 2018). *Steam in 2017*. Obtido de Medium: https://galyonk.in/steam-in-2017-129c0e6be260

Gamasutra. (28 de March de 2014). *Is it cheating to use a game engine*. Obtido de Gamasutra: https://www.gamasutra.com/blogs/AlanThorn/20140328/214178/Is_it_cheating_to_use_a_game_engine.php

Gamesbrief. (26 de November de 2012). *Why are card battle games so popular?* Obtido de Gamesbrief: https://www.gamesbrief.com/2012/11/why-are-card-battle-games-so-popular-gamesbriefers/

GamesIndustry.biz. (15 de December de 2017). *The great challenge of contemporary video game production*. Obtido de GamesIndustry.biz: https://www.gamesindustry.biz/articles/2017-12-15-the-great-challenge-of-contemporary-video-game-production

Growth Engineering. (s.d.). *What are serious games*. Obtido de Growth Engineering: https://www.growthengineering.co.uk/what-are-serious-games/

Growth Engineering. (s.d.). *What is the definition of gamification?* Obtido de Growth Engineering: https://www.growthengineering.co.uk/definition-of-gamification/

Hacker Noon. (27 de August de 2017). *You're Not Paying the True Cost of Software Development, and It's Killing Your Business*. Obtido de Hacker Noon: https://hackernoon.com/youre-not-paying-the-true-cost-of-software-development-and-it-s-killing-your-business-8d44b54be055

Investopedia. (30 de March de 2018). *Perceived Value*. Obtido de Investopedia: https://www.investopedia.com/terms/p/perceived-value.asp

Itch.io. (s.d.). *Most used Engines*. Obtido de Itch.io: https://itch.io/game-development/engines/most-projects

Living made easy. (30 de January de 2018). *Benefits of playing card games*. Obtido de Living made easy: https://www.livingmadeeasy.org.uk/scenario.php?csid=421

Mark Richards. (s.d.). *Software Architecture Patterns by Mark Richards*. Obtido de O'Reilly: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

Mens, T., & Eden, A. H. (11 de April de 2005). *On the Evolution Complexity of Design Patterns*. Obtido de Science Direct: https://www.sciencedirect.com/science/article/pii/S1571066105001465

Microsoft. (20 de julho de 2015). *Introduction to the C# Language and the .NET Framework*. Obtido de Microsoft Docs: Introduction to the C# Language and the .NET Framework

Microsoft. (8 de outubro de 2016). *A Tour of the C# Language*. Obtido de Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/index

Microsoft. (4 de January de 2019). *Get started with unit testing*. Obtido de Visual Studio Docs: https://docs.microsoft.com/en-us/visualstudio/test/getting-started-with-unit-testing

Moodle Isep. (s.d.). *Aplicações Gráficas Avançadas*. Obtido de Moodle Isep: https://moodle.isep.ipp.pt/course/view.php?id=7078

Newzoo. (s.d.). *Most Popular Core PC Games | Global*. Obtido de Newzoo: https://newzoo.com/insights/rankings/top-20-core-pc-games/

Parham, J. (21 de August de 2017). *Make a CCG – Intro*. Obtido de The Liquid Fire: http://theliquidfire.com/2017/08/21/make-a-ccg-intro/

Peter A.Koen, G. M. (2004). *The PDMA ToolBook for New Product Development.* John Wiley & Sons.

Possible. (16 de December de 2014). *7 difficulties in game development*. Obtido de Possible: https://www.possible.com/games/blog-7-difficulties-in-game-development

RPG Maker. (s.d.). *RPG Maker*. Obtido de http://www.rpgmakerweb.com/products/programs/rpg-maker-mv

Sawyer, C. (s.d.). *...about Chris Sawyer & Game Development*. Obtido de ChrisSawyerGames: http://www.chrissawyergames.com/faq3.htm

Softpedia News. (4 de April de 2014). *Why Card Games Are So Popular in Japan and How the West Is Catching Up*. Obtido de Softpedia News: https://news.softpedia.com/news/Why-Card-Games-Are-So-Popular-in-Japan-and-How-the-West-Is-Catching-Up-435737.shtml

Stackify. (27 de March de 2017). *Why .NET Core and C# are the Next Big Thing*. Obtido de Stackify: https://stackify.com/net-core-csharp-next-programming-language/

StackOverflow. (5 de February de 2011). *Why were old games programmed in assembly when higher level languages existed?* Obtido de StackOverflow: https://stackoverflow.com/questions/4904707/why-were-old-games-programmed-in-assembly-when-higher-level-languages-existed

Steam. (s.d.). *GameMaker Studio 2 Desktop*. Obtido de Steam: https://store.steampowered.com/app/585410/GameMaker_Studio_2_Desktop/

Steam. (s.d.). *RPG Maker MV*. Obtido de Steam: https://store.steampowered.com/app/363890/RPG_Maker_MV/

SteamSpy. (s.d.). *Summary Overall*. Obtido de SteamSpy: https://steamspy.com/year/

Studytonight. (s.d.). *Introduction To Game Engineering*. Obtido de Studytonight: https://www.studytonight.com/3d-game-engineering-with-unity/introduction

SuperData. (s.d.). *Digital Collectible Card Games Report*. Obtido de SuperData: https://www.superdataresearch.com/market-data/digital-card-games/

teacheramccarthymasco. (14 de Setembro de 2012). *Fundamentals of Java: AP Computer Science Essencials, 4th Edition*. Obtido de SlideShare: https://www.slideshare.net/teacheramccarthymasco/chapter-01-14289550

Techaeris. (16 de January de 2018). *Techaeris*. Obtido de These are a few common challenges that gaming developers are facing: https://techaeris.com/2018/01/16/challenges-gaming-developers-facing/

Technology Advice. (15 de February de 2019). *TechnologyAdvice Gamification Software Buyer's Guide*. Obtido de Technology Advice: https://technologyadvice.com/gamification/

The Strong. (s.d.). *Video Game History Timeline*. Obtido de The Strong National Museum of Play: https://www.museumofplay.org/about/icheg/video-game-history/timeline

Tutorials Point. (s.d.). *Component-Based Architecture*. Obtido de Tutorials Point: https://www.tutorialspoint.com/software_architecture_design/component_based_architecture.htm

Twitch Metrics. (July de 2019). *The Most Watched Games on Twitch, July 2019*. Obtido de Twitch Metrics: https://www.twitchmetrics.net/games/viewership

Unity Technologies. (2018). *The world's leading real-time engine*. Obtido de Unity: https://unity3d.com/unity

Unreal Engine. (s.d.). *What is Unreal Engine 4*. Obtido de Unreal Engine: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

Venture Beat. (11 de August de 2015). *Hearthstone leads digital card games as they become 'a dominant category,' SuperData finds*. Obtido de Venture Beat: https://venturebeat.com/2015/08/11/hearthstone-and-other-digital-collectible-card-games-becoming-a-dominant-category-superdata-finds/

Venture Beat. (23 de January de 2018). *The cost of games*. Obtido de Venture Beat: https://venturebeat.com/2018/01/23/the-cost-of-games/

Venture Beat. (9 de May de 2019). *65% of American adults play video games*. Obtido de Venture Beat: https://venturebeat.com/2019/05/09/65-of-american-adults-play-video-games/

Visual Paradigm. (s.d.). *Why UML Modeling?* Obtido de Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/why-uml-modeling/

Warburton, R. (2015). *Object-Oriented vs Functional Programming - Bridging the Divide Between Opposing Paragdims.* O'Reilly Media, Inc.

Wintellect. (14 de April de 2014). *Model-View-ViewModel (MVVM) Explained*. Obtido de Wintellect: https://www.wintellect.com/model-view-viewmodel-mvvm-explained/

Wolf, M. (5 de February de 2011). *Super Mario Evolution Timeline*. Obtido de Yum Yum Matt: https://yumyummatt.wordpress.com/2011/02/05/super-mario-timeline/

Woodall, T. (2003). Conceptualising 'Value for the Customer'. *Academy of Marketing Science*.

YoYo Games. (s.d.). *Yoyo Games*. Obtido de Yoyo Games: https://www.yoyogames.com/