UNIVERSIDADE ABERTA
www.uab.pt

Projeto Final em Engenharia Informática

The Immersive Learning Brain

RELATÓRIO FINAL

Ari Silva – aluno 1802549

Prof. Leonel Morgado (UAb & INESC TEC)
Dr. Fernando Cassola (FEUP & INESC TEC)
UAb, junho de 2022

Índice

Lista de Figuras

Lista de Diagramas

Lista de Tabelas

Introdução

Em 2014, foi criada uma organização sem fins lucrativos, a Immersive Learning Research Network (iLRN), que tem por missão principal juntar as comunidades académicas e profissionais desta área, levando-as a unir esforços para desenvolver uma base rigorosa e sólida de evidências acerca de, no fundo, de "o que é que funciona" com aprendizagem imersiva, realidade imersiva (IR) e realidade estendida (XR). A iLRN organiza todos os anos conferências mundiais onde a comunidade tem a oportunidade de apresentar e partilhar as suas investigações e avanços na área, bem como quais têm sido os desafios encontrados. O que se pretende é, no fundo, um momento de comunicação, partilha e discussão entre toda a comunidade, para perceber de que forma a aprendizagem imersiva pode contribuir para resolver problemas práticos do mundo.

Desde logo, um dos grandes desafios que a rede tem sentido vem da forte interdisciplinaridade do campo de investigação, que é muito disperso e vai desde a medicina, à construção civil, psicologia, ensino, entre muitas outras áreas. A rede iLRN sentiu que estava na altura de, num esforço pioneiro para a sistematização do conhecimento no campo, lançar uma iniciativa, a *Knowledge Tree*, que pretende unificar a diversidade de perspetivas e abordagens à investigação em aprendizagem imersiva [iLRN 2022].

Como resposta direta a esta iniciativa da rede, a *Knowledge Tree*, o meu orientador, Prof. Leonel Morgado, em coautoria com outros investigadores, levou a cabo um trabalho de levantamento, uma revisão sistemática de revisões da literatura entre os anos 2000 e 2019, tendo analisado centenas de extratos de artigos publicados e as suas referências bibliográficas, que sustentam a ocorrência de usos de ambientes imersivos de aprendizagem, fornecendo uma visão geral acerca desses usos e do contexto em que ocorrem.

Este projeto de fim de curso vem na sequência desse esforço de levantamento sistemático nas revisões da literatura. Pretende-se desenvolver uma ferramenta de análise que permita aos investigadores avaliar os resultados práticos desse estudo (e suas eventuais atualizações) de forma visualmente imersiva e interativa. O que se pretende não é uma simples apresentação de resultados aos utilizadores, mas antes uma forma de eles próprios poderem desenvolver o espírito crítico acerca da investigação publicada, e que assim daí surjam novas questões que permitam evoluir os modelos apresentados.

ORGANIZAÇÃO DO RELATÓRIO

Este documento corresponde ao relatório final do projeto desenvolvido. Nas próximas secções deste documento, começa-se por identificar casos de uso a serem implementados no projeto, realiza-se um levantamento e avaliação dos requisitos funcionais do sistema e apresentam-se algumas maquetes de serviram como referência para a implementação da solução. Passa-se depois à fase de implementação, onde se explica que decisões se tomaram para responder aos requisitos do sistema e que soluções se encontraram para resolver os casos de uso identificados. Termina-se este relatório

com secções de anexos onde se pode consultar o código de todos os sistemas desenvolvidos.

Capítulo 1

Descrição e objetivos do trabalho

O artigo "*Finding the Gaps About Uses of Immersive Learning Environments: A Survey of Surveys*" [Beck, Morgado and Shea 2020] fornece uma visão geral acerca dos usos de Ambientes Imersivos de Aprendizagem e o contexto em que ocorrem, com base numa revisão sistemática de revisões da literatura entre 2000 e 2019, que analisou centenas de extratos que comprovam a ocorrência desses usos. O trabalho de base desse artigo analisou também práticas e estratégias pedagógicas empregues com esses ambientes, dados disponíveis, mas ainda em fase de publicação. Este projeto pretende contribuir para explorar este manancial de elementos sobre esta área científica, dando corpo num ambiente visualmente imersivo a formas de análise diversificadas. Este documento corresponde ao relatório final do projeto, descrevendo a sua evolução e o estado atual.

1.1. PROPOSTA PARA O TRABALHO

O projeto proposto consiste no desenvolvimento de uma solução de software que permita a geração de modelos interativos em 3 dimensões, que facilitem ao utilizador uma compreensão rápida e intuitiva de usos associados aos investigadores da área, a partir dos dados de suporte aos artigos publicados. Tratando-se de uma solução baseada em visualização imersiva, o ideal seria que os resultados fossem observados em dispositivos como capacetes ou óculos de realidade virtual. Existe, no entanto, uma limitação relativa ao meu acesso a este tipo de dispositivos, pelo que o desenvolvimento e os testes serão realizados apenas com recurso ao ecrã do computador pessoal, mas a equipa de orientação irá testar a solução também em óculos de realidade virtual. A solução será desenvolvida com recurso à ferramenta de desenvolvimento Unity [Unity 2022] para ser executada com a tecnologia WebGL [WebGL 2022].

1.2. OBJETIVOS

No final do projeto, é esperado que os seguintes objetivos tenham sido alcançados:

1. Familiarização com ferramentas que permitam o desenvolvimento de ambientes imersivos e com implementação de soluções em realidade imersiva.
2. Identificação de casos de uso para análise de redes temáticas de ciência.
3. Conceção e implementação de um protótipo de suporte aos casos de uso identificados, para realização dessa análise em realidade virtual imersiva.

1.3. RESULTADOS ESPERADOS

No final do projeto, espera-se que o sistema desenvolvido seja capaz de:

1. Apresentar um conjunto de diagramas de redes implementados a partir dos dados disponibilizados, que suportem os casos de uso identificados;
2. Permitir aos utilizadores a interação com os modelos apresentados, para suportar os casos de uso identificados.

1.4. CRONOGRAMA

O cronograma da calendarização do projeto está dividido em 3 fases. A primeira fase, *Discovery*, foi mais dedicada a atividades de preparação e análise. Foram nesta fase analisadas as metodologias, tecnologias e arquitetura que guiaram a fase seguinte, *Development*, dedicada às atividades de desenvolvimento. As atividades 1.7 e 3.2 são atividades críticas no planeamento e *milestones* do projeto, tendo a sua calendarização sido fixa. Este cronograma havia sido definido quando da proposta inicial do projeto, e podemos agora observar a sua evolução e estado atual das atividades na coluna Evolução.

Fase	Atividade	Calendarização		Evolução
		Início	Fim	
1 - Discovery	1.1 - Familiarização com Unity	25/03/2022	31/03/2022	100%
	1.2 - Familiarização com algoritmos de visualização de redes	01/04/2022	14/04/2022	100%
	1.3 - Entrega da proposta inicial	06/04/2022	06/04/2022	100%
	1.4 – Identificação de casos de estudo	15/04/2022	20/04/2022	100%
	1.5 – Análise e diagramas	25/04/2022	29/04/2022	100%
	1.6 - Preparação do relatório intermédio	02/05/2022	10/05/2022	100%
	1.7 - Entrega do relatório intermédio	11/05/2022	11/05/2022	100%
2 - Development	2.1 - Implementação casos de uso 1	12/05/2022	20/05/2022	100%
	2.2 - Testes 1	23/05/2022	25/05/2022	100%
	2.1 - Implementação casos de uso 2	26/05/2022	03/06/2022	100%
	2.2 - Testes 2	06/06/2022	10/06/2022	100%
3 - Closure	3.1 - Preparação Relatório Final	13/06/2022	28/06/2022	100%
	3.2 - Entrega Relatório Final	29/06/2022	29/06/2022	100%
	3.3 – Apresentação	a definir	a definir	

Tabela 1: Cronograma do projeto

Capítulo 2

Enquadramento teórico

Este capítulo vai identificar e avaliar os casos de uso e correspondentes requisitos operacionais do sistema, numa sequência de lógica crescente, i.e., começa-se por identificar os casos de uso do sistema, passa-se a um levantamento e análise dos requisitos necessários à implementação dos casos de uso, é definida a arquitetura e interações do sistema e no final podem ser encontradas ilustrações de maquetes do que se pretende implementar.

2.1 IDENTIFICAÇÃO DOS CASOS DE ESTUDO

O artigo publicado "*Finding the Gaps about Uses of Immersive Learning Environments: A Survey of Surveys*" [Beck, Morgado and Shea 2020] fornece um vasto e complexo conjunto de dados acerca da ocorrência e contexto dos usos de Ambientes Imersivos de Aprendizagem, dados esses que precisam agora de ser trabalhados e dispostos de forma a que possam ser avaliados e compreendidos pela comunidade científica. A aluna Madalena Sousa (up202003391), do Mestrado em Multimédia pela Faculdade de Engenharia da Universidade do Porto, identificou um conjunto de casos de uso para esses dados, que servem de base aos que iremos aqui implementar. Juntamente com os orientadores, Prof. Leonel Morgado e o Dr. Fernando Cassola, decidiu-se implementar a análise temporal da evolução dos usos da literatura analisada no artigo. A análise temporal dos usos deverá permitir dar resposta a *user stories* como "Um investigador deseja analisar como evoluem os clusters de usos ao longo do tempo, nas coordenadas do espaço da imersão. À medida que os anos passam e mais artigos vão sendo publicados, que usos aparecem? Que usos aumentam em prevalência?" e "Um investigador deseja comparar os clusters de usos atuais (critério: coordenadas no espaço da imersão) com clusters de usos que emergem segundo outros critérios e ver como se vão alterando ao longo do tempo." Foram assim identificados os seguintes casos de uso a implementar nesta fase do projeto:

1. Qual é a evolução dos usos ao longo do tempo?
    - Para cada ano, devem ser mostradas esferas, correspondentes aos usos, de dimensão correspondente aos "accounts" (relatos) provenientes de artigos publicados nesse ano.
2. Quais foram os usos publicados num ano?
    - Para cada ano, devem ser mostradas esferas correspondentes aos usos, com a dimensão correspondente ao total de "accounts" provenientes de artigos publicados até esse ano, i.e., nesse ano e em anos anteriores.

2.2 REQUISITOS DO SISTEMA

Logo desde o início do projeto, foram agendadas reuniões semanais de acompanhamento com os orientadores do projeto, o Prof. Leonel e o Prof. Fernando. Estas sessões semanais de acompanhamento, foram absolutamente determinantes para o sucesso da solução, uma vez representaram um espaço aberto de discussão para compreender e definir, quais os resultados que se pretendia no final do projeto. A identificação dos requisitos do sistema, foi acontecendo à medida que essa discussão ia sendo aprofundada e à medida que o próprio desenvolvimento ia evoluindo. O artigo que serviu de base a este projeto foi sendo esmiuçado para eu ficasse com uma ideia clara acerca do seu contexto, do seu âmbito, mas também do seu alcance. A compreensão clara do que representam os resultados desse artigo, aliada à exploração das potencialidades do *Unity*, permitiu que fossemos encontrando os nossos requisitos do sistema, a seguir descritos:

1. O sistema deve ser desenvolvido com recurso à ferramenta de desenvolvimento Unity, de forma a permitir uma representação dos modelos numa realidade imersiva 3D.
2. Para que o sistema fique globalmente acessível a toda a comunidade científica, ele ser executado com a tecnologia WebGL.
3. Por forma a permitir a execução do sistema com a tecnologia WebGL, deve ser desenvolvido um serviço Web que disponibilize os dados necessários ao seu funcionamento.
4. O sistema deve permitir aos utilizadores a seleção de um conjunto de filtros que permitam limitar os resultados apresentados à análise pretendida.
5. Os modelos devem apresentar um conjunto de esferas, correspondentes aos usos filtrados pelo utilizador.
6. A visualização dos modelos apresentados, deve ser fixa no espaço, mas animada no tempo.
7. As esferas no modelo devem estar posicionadas nas coordenadas no espaço de imersão, de acordo com o definido na tabela 4 "*Classification of use themes per immersion dimension*" do artigo base citado [Beck, Morgado and Shea 2020].
8. As esferas nos modelos devem ter uma dimensão variável, de acordo com os "accounts" que resultem dos filtros selecionados pelo utilizador.
9. As esferas no modelo devem apresentar cores diferentes, que as permitam diferenciar dos restantes usos.
10. O sistema deve permitir ao utilizador a interação com os modelos apresentados.
11. Cada esfera no modelo deve permitir ao utilizador obter a informação acerca dos dados que lhe deram origem, i.e., as suas publicações e respetivos autores e instituições.

2.3 MODELO DE DADOS

Os dados que irão alimentar e permitir o correto funcionamento do sistema, foram previamente definidos pela aluna de mestrado Madalena Sousa, que gentilmente nos disponibilizou numa base de dados SQLite. A estrutura e relações das tabelas estavam

já todas definidas, tendo existido a necessidade de adicionar apenas uma nova tabela, *uses_immersion_classification*, que permita armazenar as coordenadas no espaço de imersão de cada uso, bem com a correspondente cor a apresentar no modelo final.

O diagrama seguinte corresponde ao modelo de dados, extraído a partir da base de dados disponibilizada, já com a nova tabela adicionada.

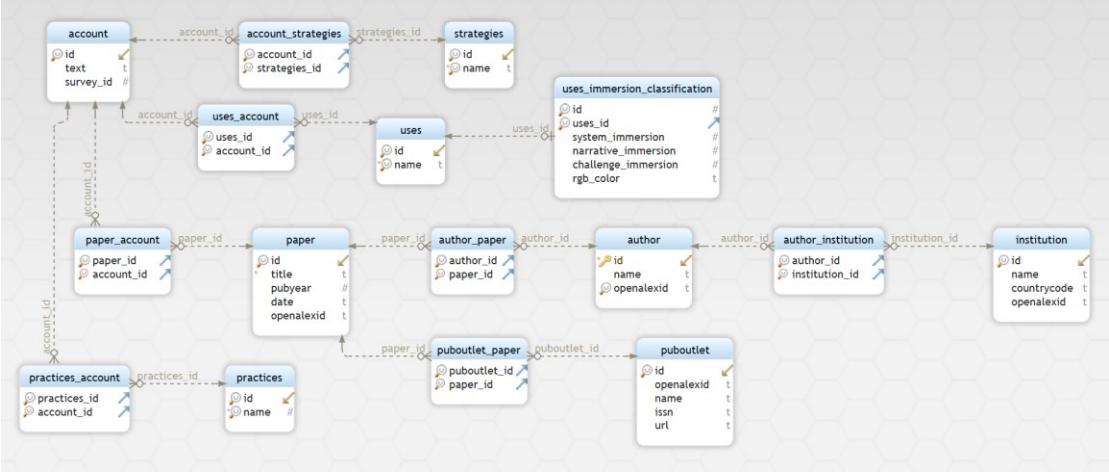


Diagrama 1: Modelo de dados

A avaliação detalhada deste modelo de dados foi absolutamente essencial para conseguir compreender de que forma os dados se relacionam e que tipo de informação, filtros e resultados podem alimentar o nosso sistema, assim como poderemos resolver os nossos casos de estudo. O foco para resolver os nossos cassos de uso, centra-se na tabela *"uses"* e na sua relação com a entidade *accounts*. É a partir daqui que conseguimos chegar a todas as outras entidades envolvidas como os *papers*, *authors*, *institutions*, etc.

2.4 ARQUITETURA E INTERAÇÕES DA SOLUÇÃO

A solução desenvolvida é essencialmente formada por dois sistemas. Uma aplicação a ser executada a partir de uma página web com a tecnologia WebGL e um serviço web que permita o acesso aos dados do modelo de dados.

O diagrama seguinte, ilustra a arquitetura global da solução, permitindo identificar quais os sistemas envolvidos na solução e como interagem entre si.

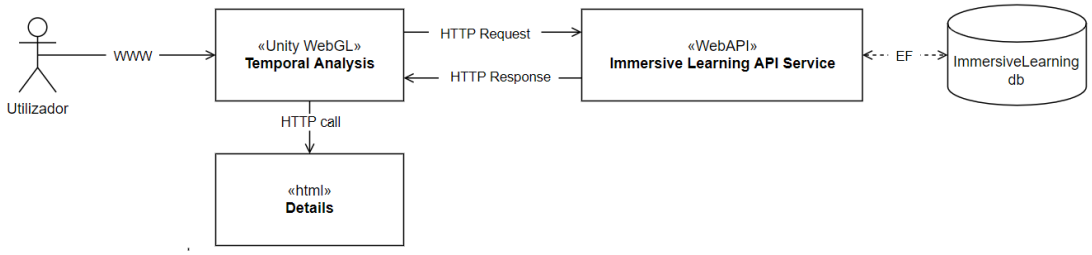


Diagrama 2: Arquitetura do sistema

Neste diagrama, podemos observar que o utilizador acede à solução através de uma página web da Internet, que corresponde ao sistema desenvolvido em *Unity*. Este sistema realiza chamadas HTTP ao serviço WebAPI para obter os dados necessários à sua utilização, a partir da base de dados. Podemos ainda observar que o sistema Unity, realiza uma chamada HTTP a uma página HTML, para apresentação de detalhes.

Diagrama de componentes do sistema Unity

O sistema Unity, é essencialmente constituído por dois componentes, o *front-end* com os elementos visuais e uma camada inferior que permite implementar a lógica da aplicação e controlar os elementos visuais, realizando chamadas HTTP ao serviço WebAPI para obter os dados necessários da base de dados e realiza o seu processamento.

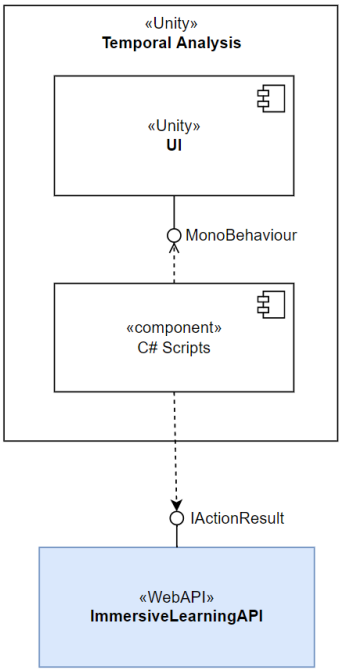


Diagrama 3: Componentes do sistema Unity

Diagrama de componentes do serviço WebAPI

O sistema corresponde ao serviço WebAPI é constituído por 5 componentes diferentes, o que permite realizar a independência e separação de responsabilidades.

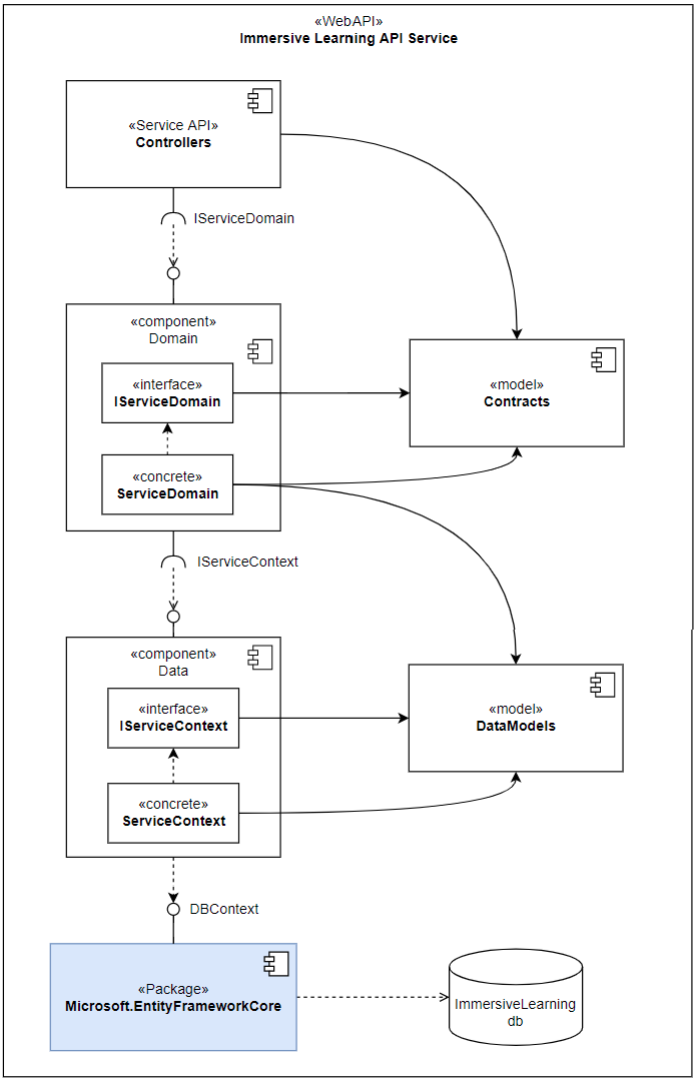


Diagrama 4: Componentes do WebAPI

Tem-se desde logo à cabeça um componente que expõe ao utilizador um conjunto de operações para a obtenção de dados. Quando evocada uma operação, existe uma chamada ao componente que detém a lógica sobre qual deve ser o resultado dessa operação, *Domain*, devolvendo objetos de *Contracts*. *Domain*, obtém os dados necessários a partir de um outro componente de acesso aos dados, *Data*, que utiliza a *Framework EntityFrameworkCore* para concretizar a comunicação com a base de dados. O componente *DataModels*, implementa as classes de mapeamento às entidades da base de dados, que *Domain* utiliza para processar os pedidos em cada operação.

Sequências de interações com a solução

Numa visão geral do funcionamento da solução, quando o utilizador acede à aplicação, ela deve iniciar carregando os elementos da interface do utilizador, com os critérios de filtragem também eles já carregados por consultas ao serviço web. O utilizador pode então definir os seus critérios de análise e depois iniciar a visualização dos modelos. Nesse momento, de acordo com os critérios selecionados, a aplicação faz novamente interações de consulta com o serviço web, de forma a conseguir obter os dados

necessários à construção dos modelos. Tratando-se de uma análise temporal, os modelos são todos contruídos e carregados por ano de análise, e só então são apresentados em forma de animação temporal. Perante a animação temporal, o utilizador pode decidir interromper a animação de forma permanente, voltando a aplicação ao seu estado inicial as preservando os critérios de filtragem da análise previamente selecionados, ou de forma temporária no ano que estiver a ser apresentado, podendo depois decidir se pretende continuar ou parar a animação. Durante a visualização dos modelos, o utilizador pode ainda interagir com eles, navegando através do rato que permitirá fazer zoom in/out, deslocar o modelo em qualquer direção e rodar o modelo em torno dos eixos. Existem 4 sequências principais de interação que o utilizador pode realizar como a solução. Num primeiro momento, o utilizador acede à aplicação, que realiza as tarefas necessárias a que ela seja apresentada corretamente. Depois de corretamente inicializada, o utilizador pode interagir com os filtros, para definir os seus critérios de análise. Uma vez definidos os critérios de análise, o utilizador pode então dar início à apresentação dos resultados. Com a animação a decorrer, o utilizador pode então interagir com os modelos gerados.

Inicialização

Esta primeira sequência corresponde ao acesso do utilizador à aplicação Unity, onde o sistema deve inicializar, carregar e apresentar os filtros disponíveis.

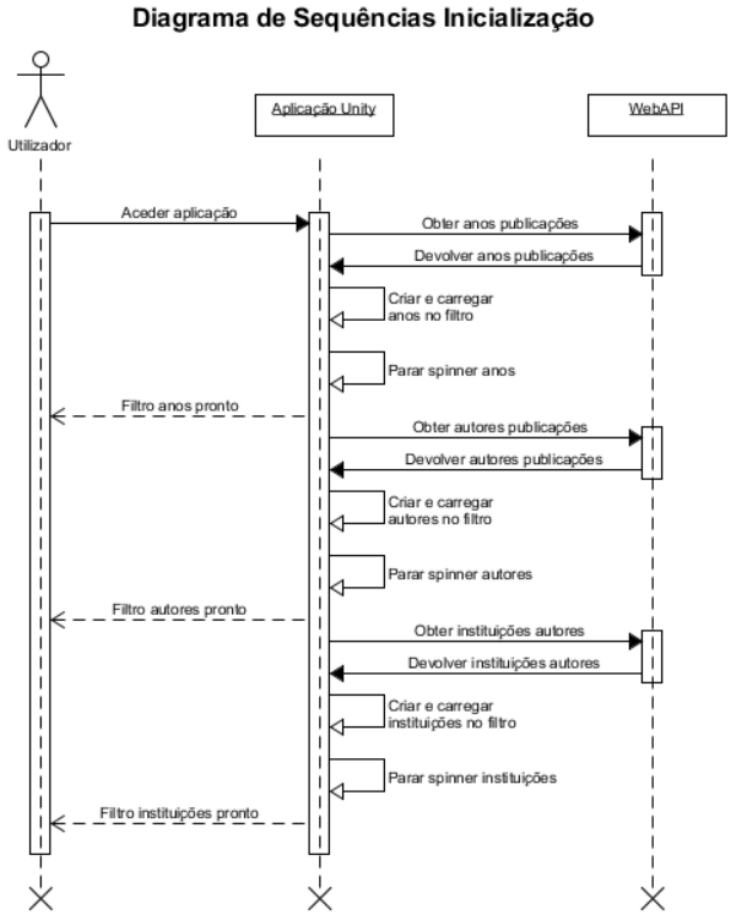


Diagrama 5: Sequências de inicialização

Interação com os filtros

Esta sequência de utilização corresponde à capacidade de o utilizador parametrizar a sua análise, interagindo com os filtros previamente inicializados.

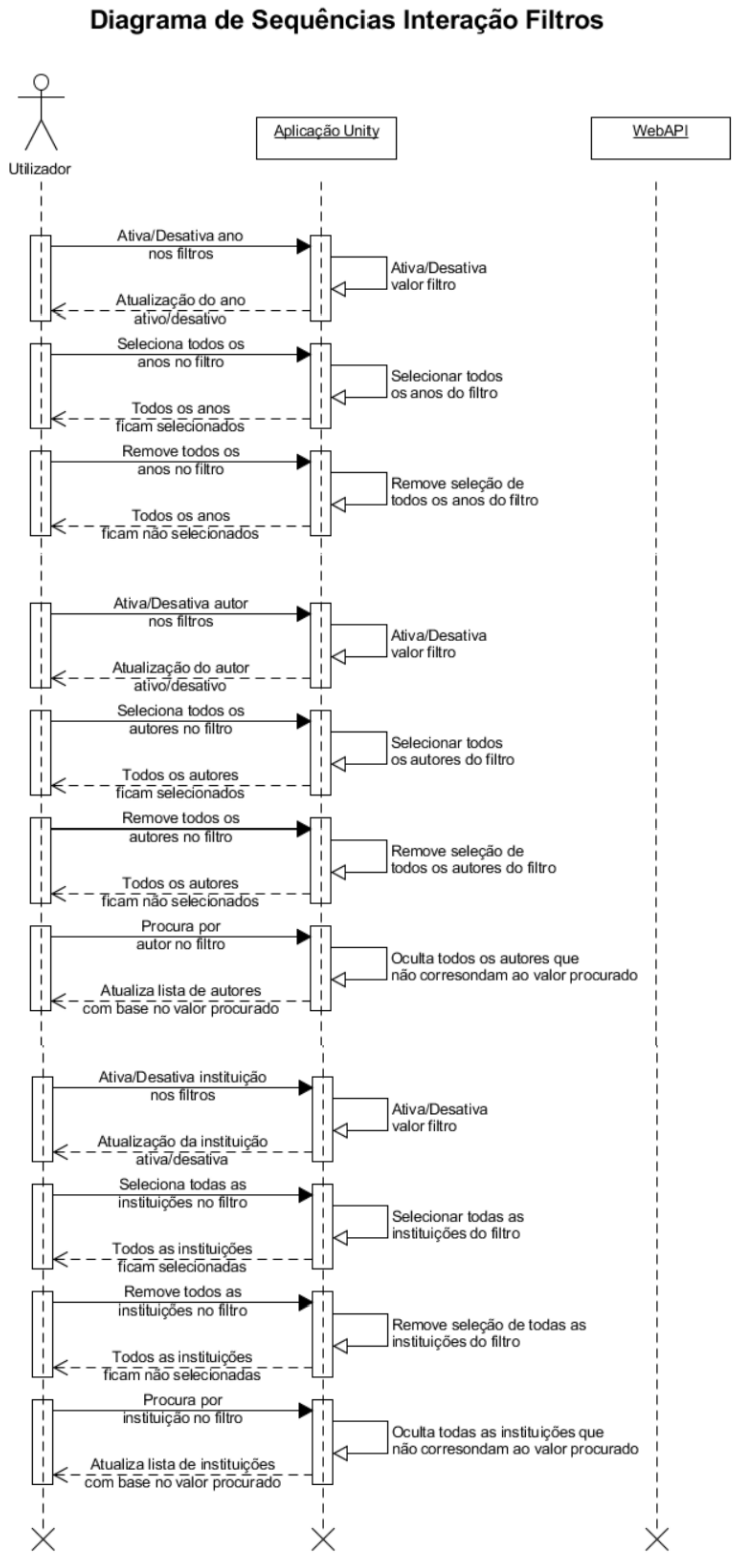


Diagrama 6: Sequência de interação com os filtros

Apresentação dos Resultados

Nesta sequência, descreve-se o processo de geração dos modelos, com base nos critérios previamente definidos pelo utilizador, e apresentação dos resultados ao utilizador.

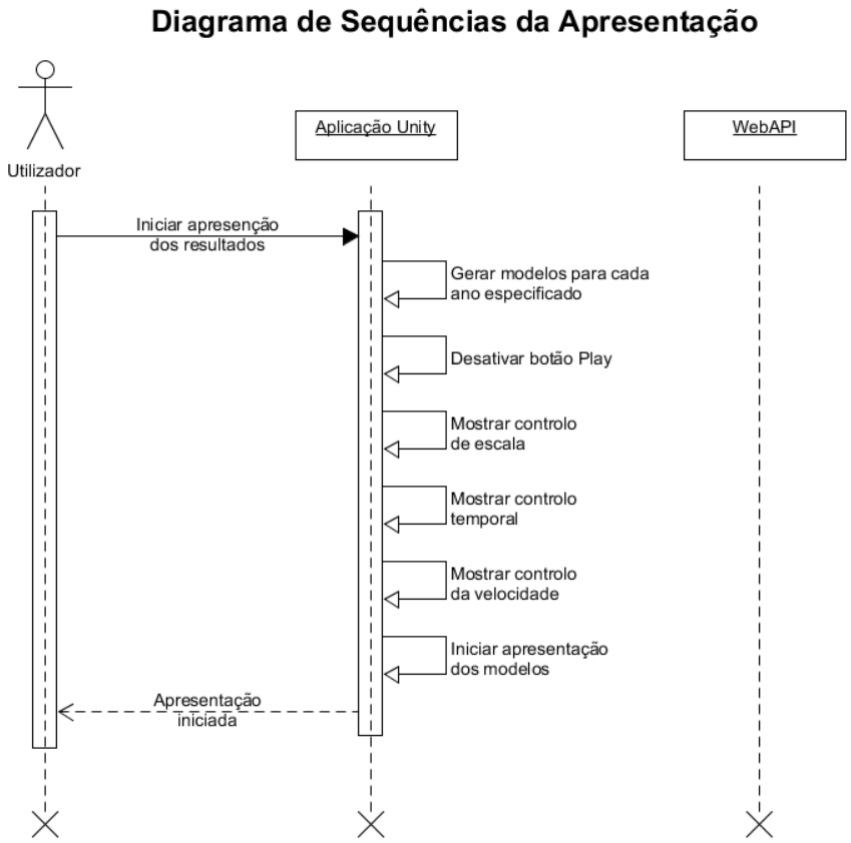
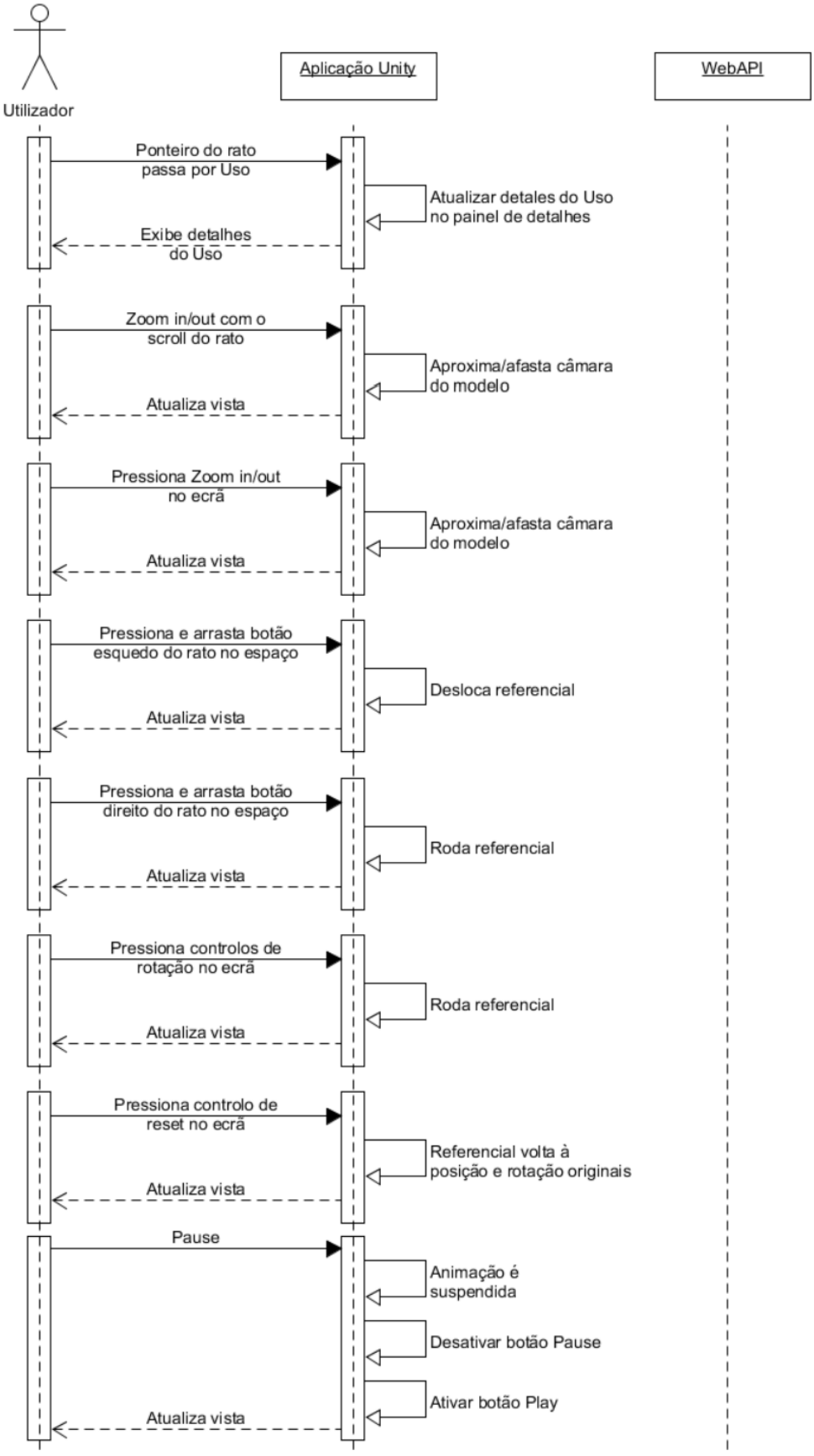


Diagrama 7: Sequências de apresentação dos resultados

Interação com a Animação

Uma vez em curso a apresentação dos resultados, o utilizador pode então interagir com os modelos que foram gerados a partir dos seus próprios critérios de análise. O utilizador pode, por exemplo, rodar e deslocar o referencial, fazer *zoom in/out* e navegar por entre os Temas de Uso, alterar a escala do referencial, alterar a velocidade da animação ou utilizar o controlo temporal para navegar manualmente por entre os modelos gerados. Note-se ainda que, quando um Tema de Uso é pressionado pelo rato, é apresentada uma página HTML com os detalhes que levaram à inclusão desse Tema de Uso no modelo.

Diagrama de Sequências de Interações Modelos



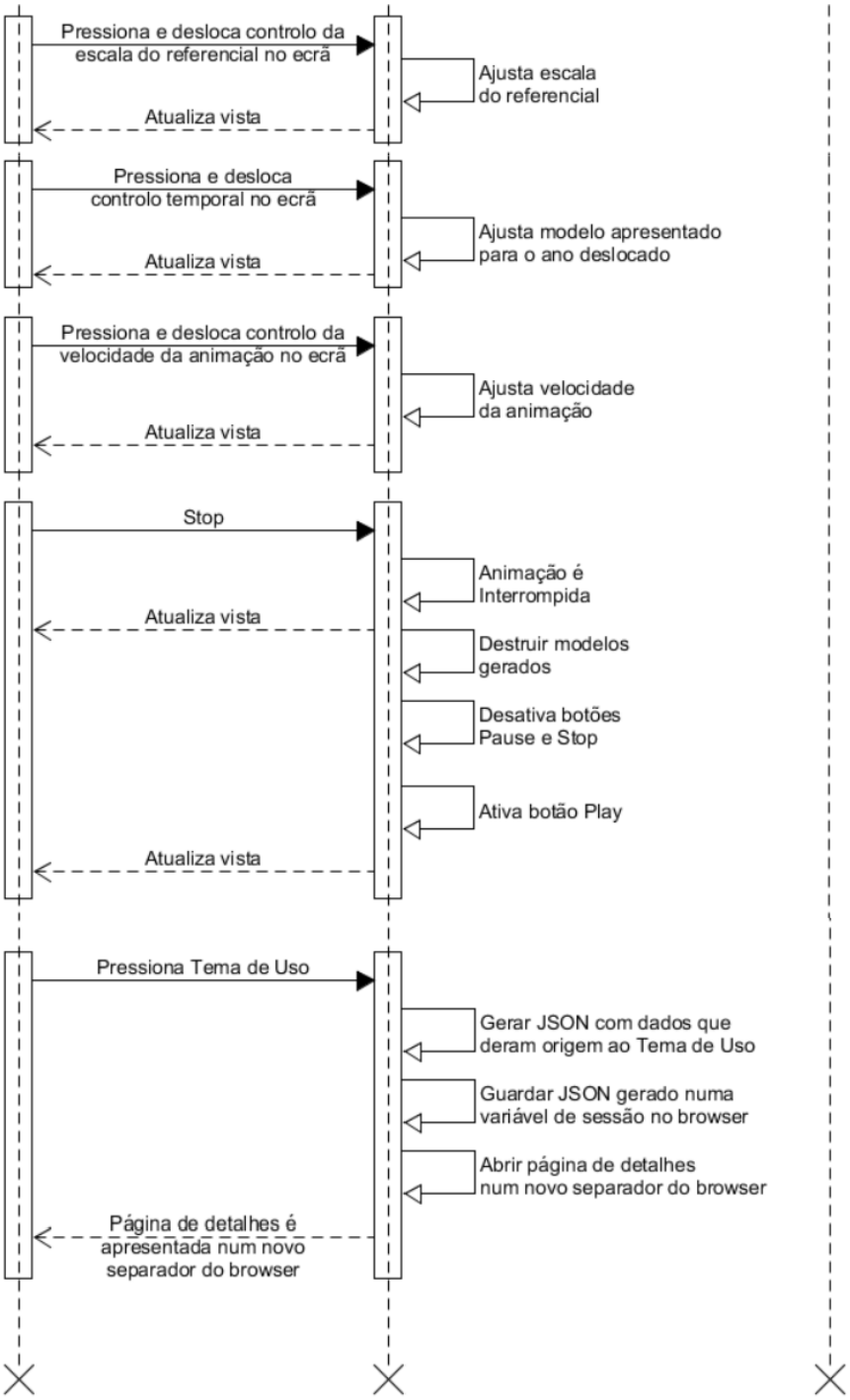


Diagrama 8: Interações com os modelos gerados

Capítulo 3

Protótipo

Nesta secção do documento, são apresentadas as diversas maquetes de interface com o utilizador, de acordo com o ponto de execução da aplicação. O aspeto visual e interativo, bem como a usabilidade da solução final, serão absolutamente determinantes para o sucesso da sua aceitação junto da comunidade científica, que se pretende como público-alvo desta solução. Estas maquetes devem ser entendidas apenas como uma referência para a implementação do sistema, sendo contante alvo de discussão e ajustamento durante as reuniões de acompanhamento do projeto.

3.1 ESTADO INICIAL

A maquete seguinte representa o estado inicial da aplicação. No painel do lado direito são apresentados os diversos critérios de filtragem para a análise pretendida, bem como as opções (no fundo do painel) que permitem ao utilizador controlar o início e fim da apresentação dos resultados. No painel esquerdo é apresentado o referencial do espaço de imersão onde vão ser apresentados os modelos gerados.

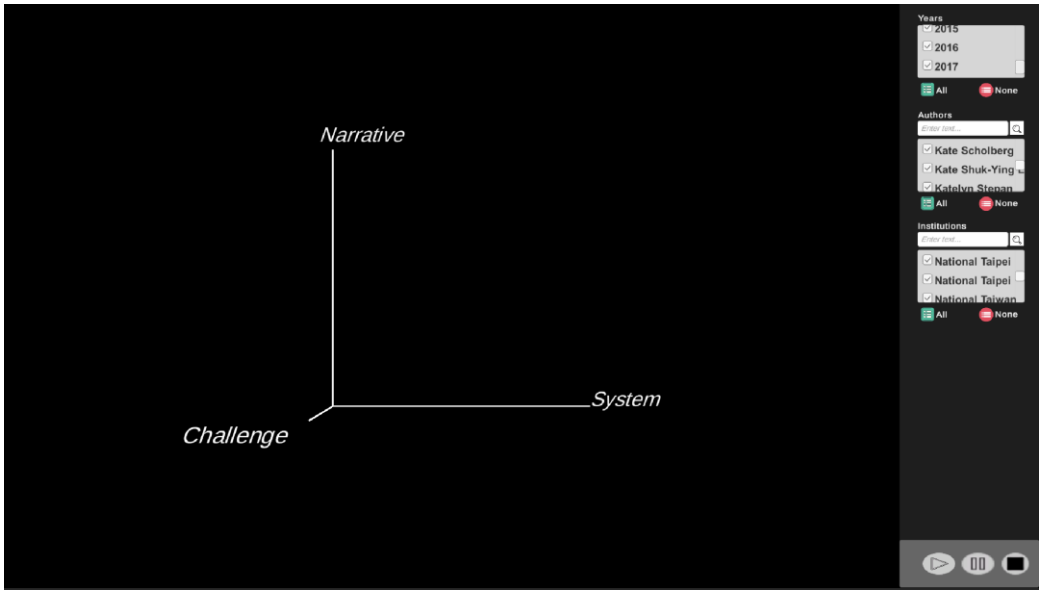


Figura 1: Estado inicial, possibilitando a seleção de critérios para análise

3.2 APRESENTAÇÃO DOS MODELOS

A maquete seguinte, representa o estado de apresentação dos modelos, de acordo com os critérios previamente selecionados. No painel esquerdo é agora apresentado o referencial do espaço de imersão com os usos carregados. Cada uso é representado por uma bola com uma cor, dimensão e ainda o nome do uso para permitir identificar facilmente o uso apresentado. No painel do lado direito, deixam de estar visíveis os filtros do utilizador, passando agora a exibir os detalhes do uso, quando o utilizador

passa com o rato por cima de um determinado uso. Ao fundo deste painel, continuam a estar disponíveis opções que permitem ao utilizador suspender/continuar a apresentação dos resultados.

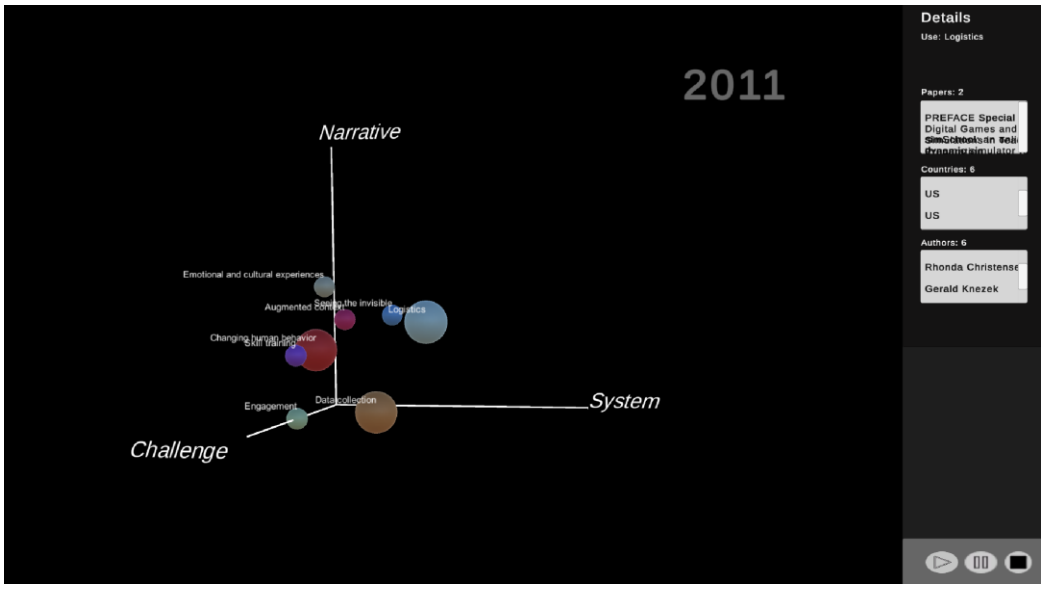


Figura 2: Apresentação do modelo em análise para um determinado ano

3.3 INTERAÇÃO COM O MODELO

A maquete seguinte, representa apenas uma vista detalhada de uma interação do utilizador com o modelo apresentado. Pode-se observar que, neste caso, o utilizador efetuou um *zoom-in* do modelo e o rodou ligeiramente em torno dos eixos *Narrative* e *System*.

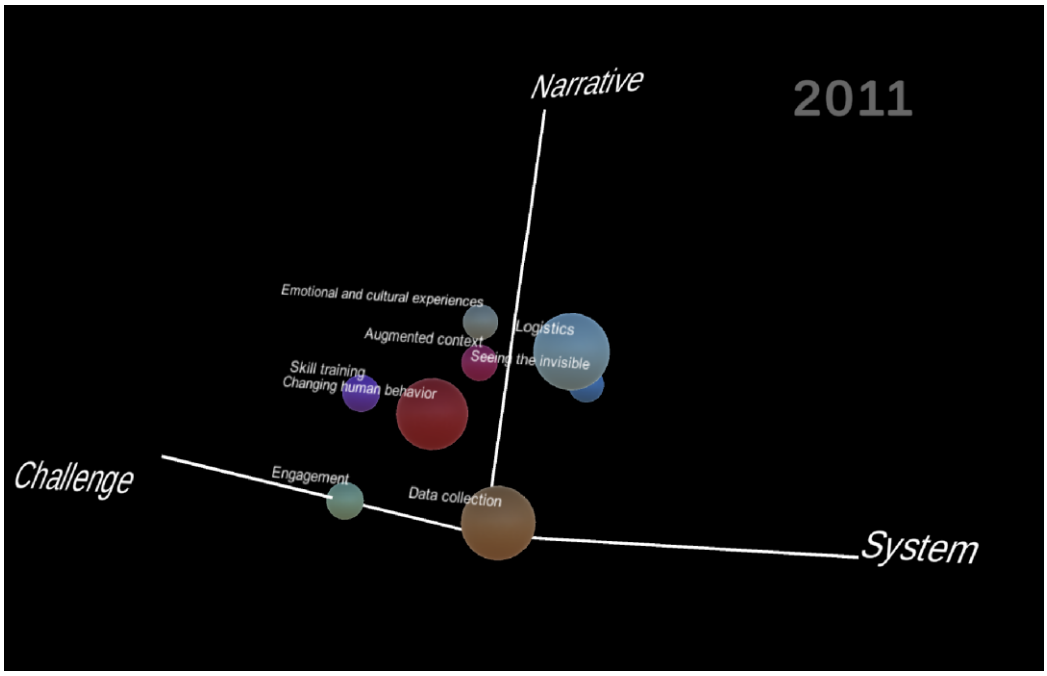


Figura 3: Interação do utilizador com o modelo apresentado

Capítulo 4

Implementação

Neste capítulo vamos abordar a fase de implementação do projeto, descrevendo as tecnologias e técnicas aplicadas em cada um dos sistemas que fazem parte da solução global do projeto. Decidimos separar a solução global em 3 sistemas principais. Um serviço Web API que permite o acesso aos dados necessários ao correto funcionamento da ferramenta, a ferramenta Unity que serve de *front-end* ao utilizador da solução e ainda uma página web de detalhes. Para cada um dos sistemas, vamos explicar a motivação que levou ao seu desenvolvimento e faremos também uma abordagem mais técnica acerca das decisões tomadas. No final deste capítulo, apresentamos ainda uma secção dos restes realizados ao sistema e depois terminamos com alguns exemplos de utilizador do serviço WebAPI.

4.1. CONTEXTO

Os requisitos do projeto definem que a solução final deve disponibilizar funcionalidades que permitam ao utilizador realizar uma análise temporal dos Temas de Uso das publicações analisadas no artigo que serve de base a este projeto [Beck, Morgado and Shea 2020]. Este artigo, gerou um manancial de dados que foram reunidos numa base de dados SQLite e posteriormente gentilmente disponibilizada pela aluna Madalena Sousa (up202003391), do Mestrado em Multimédia pela Faculdade de Engenharia da Universidade do Porto. Os requisitos da solução definem ainda que a solução deve permitir aos investigadores avaliar os resultados práticos dessa investigação, proporcionando ao utilizador uma experiência imersiva, e onde o utilizador consiga ele próprio interagir com os dados, devendo ainda a solução ser disponibilizada em ambiente WebGL. Decidimos então que o IDE de desenvolvimento Unity consegue dar resposta aos nossos requisitos funcionais.

4.2. WEBAPI

Desde logo, um dos problemas com que imediatamente se depara foi o de como conseguir ter uma solução desenvolvida em Unity, a correr sobre o WebGL e a aceder aos dados armazenados numa base de dados SQLite. Ora, o problema põe-se porque existe de facto uma limitação na tecnologia. O Unity não implementa, de forma nativa, qualquer interface que permita a comunicação direta com bases de dados SQLite. Para que essa comunicação possa acontecer, deve ser incluída no projeto Unity a biblioteca externa *sqlite3,* ficando esta dinamicamente referenciada no projeto. Esta solução resolve o problema para projetos Unity a correr em ambiente Windows, mas, neste caso, precisava-se que a solução corresse sobre o WebGL, que não permite a referenciação dinâmica de bibliotecas externas. Além disso, ao acoplar a base de dados ao projeto Unity, ela ficaria unicamente disponível para consumo desta solução em particular. O conjunto de dados gerados pela investigação de base, é tão rico que pode, e deve, ser explorado sob muitas outras perspetivas, para além dos atuais casos de uso e, portanto,

decidiu-se que se deveria aproveitar esta oportunidade para lançar a base de um serviço que, para além de disponibilizar os dados para a ferramenta, possa também ser facilmente estendido a ajustável e futuras implementações e clientes desses dados.

O serviço WebAPI disponibiliza 5 diferentes *routes*, que permitem o acesso a cada uma das entidades principais relevantes aos casos de uso do projeto, api/*accounts, api/authors, api/institutions, api/papers*, *api/uses*. Foi ainda criada uma *route* adicional que permite a obtenção de um *token* de autenticação.

4.2.1 Configuração do Projeto no Visual Studio

O WebAPI foi implementado sobre a *framework* de desenvolvimento .net Core 3.1, com recurso ao Visual Studio 2022® e está neste momento a correr num servidor IIS sob o endereço base https://immersivelearningapi.azurewebsites.net.

A estrutura da solução está distribuída entre 5 diferentes projetos do Visual Studio®, que correspondem a diferentes camadas de abstração para permitir a separação de responsabilidades e promover extensibilidade em futuras alterações.

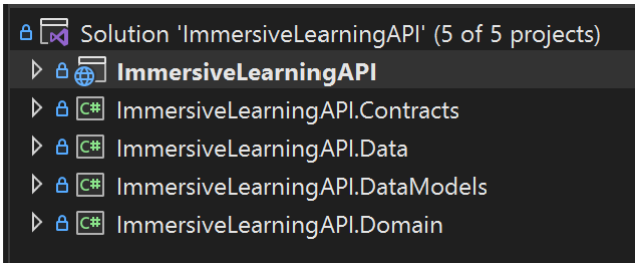


Diagrama 9: Estrutura da solução WebAPI no Visual Studio

Na imagem acima, pode-se ver os projetos que constituem a estrutura da solução, que foi definida no Visual Studio® e que se explica de seguida, incluindo também justificações para algumas das opções tomadas durante o desenvolvimento.

4.2.2 Mapeamento das Entidades

O mapeamento das Entidades da base de dados foi implementado no projeto *ImmersiveLearningAPI.DataModels*. Foram mapeadas apenas as entidades relevantes aos casos de uso do projeto. De notar que para permitir a utilização de *frameworks* que facilitem a comunicação com a base de dados, como é o caso *Entity Framework* que se utiliza na solução, devem ser observados alguns cuidados especiais aquando da definição destas classes de mapeamento. Veja-se, como exemplo, a entidade *uses_immersion_classification* da base de dados, cuja estrutura da tabela pode ser observada na imagem seguinte.

```
CREATE TABLE "uses_immersion_classification" (
    "id"    INTEGER UNIQUE,
    "uses_id"   INTEGER UNIQUE,
    "system_immersion"  REAL,
    "narrative_immersion"   REAL,
    "challenge_immersion"   REAL,
    "rgb_color"         TEXT,
    PRIMARY KEY("id" AUTOINCREMENT)
)
```

Figura 4: Tabela uses_immersion_classification - SQLite

Esta tabela está definida com o nome "uses_immersion_classification" e é constituída por 6 campos "id", "uses_id", "system_immersion" , "narrative_immersion" , "challenge_immersion" , "rgb_color", sendo o "id" a chave primária da tabela. A classe de mapeamento desta tabela no nosso projeto, corresponde a

```
[Table("uses_immersion_classification")]
2 references | Ari Silva, 36 days ago | 1 author, 1 change
public class UsesImmersionClassificationModel
{
    [Key]
    0 references | Ari Silva, 36 days ago | 1 author, 1 change
    public int Id { get; set; }
    [Column("uses_id")]
    1 reference | Ari Silva, 36 days ago | 1 author, 1 change
    public int UseId { get; set; }
    [Column("system_immersion")]
    1 reference | Ari Silva, 36 days ago | 1 author, 1 change
    public float SystemImmersion { get; set; }
    [Column("narrative_immersion")]
    1 reference | Ari Silva, 36 days ago | 1 author, 1 change
    public float NarrativeImmersion { get; set; }
    [Column("challenge_immersion")]
    1 reference | Ari Silva, 36 days ago | 1 author, 1 change
    public float ChallengeImmersion { get; set; }
    [Column("rgb_color")]
    1 reference | Ari Silva, 36 days ago | 1 author, 1 change
    public string RGBColor { get; set; }
}
```

Figura 5: Tabela uses_immersion_classification – Classe mapeada

Ora, decidiu-se chamar a esta classe *UsesImmersionClassificationModel,* que não corresponde ao nome da tabela na base de dados, *uses_immersion_classification.* Para que o Entity Framework consiga mapear esta classe na tabela da base de dados, precisa-se de incluir o atributo [Table("uses_immersion_classification")] (namespace System.ComponentModel.DataAnnotations.Schema). Da mesma forma, também os nomes das propriedades na classe não correspondem exatamente no nome dos campos na tabela, e também aqui é preciso incluir um atributo para que o mapeamento ocorra, como é o caso da propriedade UseId cujo atributo é [Column("uses_id")], e assim sucessivamente para as outras propriedades da classe. Outro atributo que também é preciso incluir na classe de mapeamento, é o que identifica qual a propriedade da classe que vai corresponder à chave primária da tabela. Neste caso, a propriedade Id deve ser identificada com o atributo [Key] (namespace System.ComponentModel.DataAnnotations).

De notar que este projeto DataModels não depende de nenhum outro projeto da solução e não existe qualquer lógica de negócio ou processamento. Existem apenas as classes de mapeamento, pelo que em qualquer futura implementação que necessite mapear outras entidades da base de dados, é aqui que essas classes de mapeamento devem ser criadas.

4.2.3 Acesso aos dados da base de dados

O acesso aos dados da base de dados foi implementado no projeto *ImmersiveLearningAPI.Data*. Este projeto da solução corresponde ao contexto da base de dados herdando da classe *DBContext* (*namespace Microsoft.EntityFrameworkCore)* e depende do projeto DataModels, uma vez que é nesse projeto onde estão definidas as classes de mapeamento que o Entity Framework vai utilizar para comunicar com a base de dados.

Este projeto tem apenas duas classes definidas. A classe de *interface IServiceContext* e a classe de implementação *ServiceContext*, que implementa *IServiceContext*. As classes *interface* na solução permitem estabelecer um contrato entre a implementação e quem dela pretenda usufruir. Estes contratos são, não apenas úteis a potenciais clientes das classes, que conseguem assim um nível de abstração relativamente à implementação, mas também absolutamente necessários a mecanismos como *dependency invection* e à definição de testes unitários. Ver-se-á mais adiante, no projeto da solução *ImmersiveLearningAPI*, como se definiu o mecanismo de *dependency injaction*. Por agora, necessita-se apenas de compreender que o construtor da classe de implementação *ServiceContext* tem de ser informado da localização e tipo de base de dados com que vai comunicar. Essa informação está definida no ficheiro de configurações do projeto principal, e está a ser injetado no contexto, aquando da instanciação da classe.

```
public class ServiceContext : DbContext, IServiceContext
{
    0 references | Ari Silva, 36 days ago | 1 author, 1 change
    public ServiceContext(DbContextOptions<ServiceContext> options)
        : base(options)
    {
```

Figura 6: Contexto de dados

O contexto da base de dados define que entidades da base de dados estão disponíveis para consumo. Nesta implementação, decidiu-se apenas disponibilizar as entidades que são efetivamente necessárias à solução. Futuras alterações terão apenas de acrescentar aqui novas entidades que necessitem de consumir.

```
//Entities
2 references | Ari Silva, 36 days ago | 1 author, 1 change
public DbSet<AccountModel> Accounts { get; set; }
3 references | Ari Silva, 36 days ago | 1 author, 1 change
public DbSet<AuthorModel> Authors { get; set; }
3 references | Ari Silva, 36 days ago | 1 author, 1 change
public DbSet<InstitutionModel> Institutions { get; set; }
8 references | Ari Silva, 36 days ago | 1 author, 1 change
public DbSet<PaperModel> Papers { get; set; }
5 references | Ari Silva, 36 days ago | 1 author, 1 change
public DbSet<UseModel> Uses { get; set; }
```

Figura 7: Coleção de Entidades no contexto de dados

Uma nota importante para futuras implementações é que cada contexto tem de ser utilizado no âmbito de uma única fonte de dados. Isto significa que, caso exista a necessidade de referenciar alguma outra base de dados, deve ser criado um contexto

novo (com a correspondente interface), totalmente independente de algum outro contexto já definido.

4.2.4 Classes de resposta ao cliente

As classes de resposta do serviço aos pedidos dos clientes estão definidas no projeto *ImmersiveLearningAPI.Contracts*. Algumas dessas respostas podem até coincidir com os mapeamentos de entidades da base de dados, definidas no projeto *DataModels*. No entanto, por uma separação de responsabilidades, essas classes são aqui novamente definidas. Para facilitar e agilizar a escrita de código, e até para evitar eventuais erros de codificação, ver-se-á mais adiante que, sempre que possível, se está a realizar um mapeamento direto entre estes dois objetos.

Pode-se também aqui observar que algumas das classes de resposta definidas vão um pouco mais além do que o simples mapeamento das entidades da base de dados. Veja-se, como exemplo, a classe *AuthorDetails*.

```
/// <summary>
/// Detailled information about an author, including all
/// institutions where this author can be found.
/// </summary>
7 references | Ari Silva, 37 days ago | 1 author, 1 change
public class AuthorDetails : Author
{
    2 references | Ari Silva, 37 days ago | 1 author, 1 change
    public List<Institution> Institutions { get; set; }
}
```

Figura 8: Resposta do serviço - AuthorDetails

Esta classe herda as propriedades públicas de *Author*, que corresponde aos campos da tabela *author* na base de dados. No entanto, tratando-se de uma classe de detalhe, ela inclui ainda uma lista de todas as instituições onde se pode encontrar o autor.

Deve ainda notar-se que o projeto da solução *Contracts* é totalmente independente de qualquer outro projeto na solução e é neste projeto que futuras implementações devem adicionar novas respostas do serviço.

4.2.5 Processamento dos pedidos dos clientes

O processamento dos pedidos de clientes é realizado no projeto ImmersiveLearningAPI.*Domain*. É neste componente se define toda a lógica e regras de negócio que permitem o processamento dos pedidos vindos de clientes do serviço. Este projeto depende dos projetos da solução *DataModels, Data* e *Contracts*, na medida em que utiliza o contexto de dados (definido em *Data*) para realizar *queries* à base de dados, processando os dados obtidos (*DataModels*) e devolvendo uma resposta (*Contracts*).

Este projeto da solução é constituído por duas classes apenas, *IServiceDomin* e *ServiceDomain*. *IServiceDomin* é uma classe de interface que permite estabelecer um contrato entre um consumidor da classe de implementação *ServiceDomain* e a própria classe de implementação, que implementa essa interface. Os métodos disponíveis na interface podem ser consultados na secção de Anexos.

A classe de implementação *ServiceDomain* possui um construtor onde são injetados, aquando da sua instanciação, os objetos correspondentes ao contexto de dados e um mapper. Este mapper permite o mapeamento entre objetos semelhantes definidos nos projetos da solução *DataModels* e *Contracts*. Para garantir a separação de contexto, um objeto dos *DataModels* nunca deve ser devolvido ao consumidor do domínio da solução, e, portanto, os conteúdos desses objetos são mapeados automaticamente em objetos de *Contracts*. Ver-se-á mais adiantes como ocorre esse mapeamento automático, assim como é realizada a injeção de objetos no construtor.

Os métodos implementados no *ServiceDomain* podem ser consultados na secção de Anexos. Vai-se apenas analisar um dos métodos implementados, a título de exemplo, para compreender que metodologias se aplicaram na codificação. Analise-se, então, o método *GetAllUses*.

```
/// <summary>
/// This method finds all returns all Theme of Uses, according
/// to defined Params.
/// </summary>
/// <param name="year">
/// This method returns all uses with publications within the specified
/// year. When 0, it returns all uses with publication, regardless of the
/// publication Year..
/// </param>
/// <param name="includeAllUpToYear">
/// When true, this method considers all publications up until the
/// specified Year (when a year was specified).
/// Otherwise, considers only publication within the specified
/// Year (when a yeas was specified)
/// </param>
/// <returns>
/// Uses, including details such as Papers) according
/// to specified Params.
/// </returns>
2 references | Ari Silva, 24 days ago | 1 author, 2 changes
public IEnumerable<Use> GetAllUses(int year, bool includeAllUpToYear)
```

Figura 9: Exemplo de método de processamento - GetAllUses

Este método recebe dois parâmetros, *year* e *includeAllUpToYear*. O parâmetro *year* permite ao cliente do método indicar se pretende obter todos os temas de uso que contenham publicações apenas no, ou até, até ao ano especificado. Como se verá a seguir, caso o valor deste parâmetro seja o (zero), o método vai devolver todos os temas de uso para todas as publicações na base de dados, independentemente do ano de publicação, como se pode ver na imagem seguinte.

```
List<Use> uses = null;
if (year == 0)
{
    //Year = 0 means all themes of use should be consider, regardless
    //of the publication year
    uses = _mapper.Map<List<Use>>(_serviceContext.Uses.ToList());
}
else
```

Figura 10: Obtenção e mapeamento da lista de Temas de Uso no método GetAllUses

Nesta figura, pode-se observar que foi indicado o valor o (zero) no parâmetro *Year*. O método decide então que deve ser construída uma lista com todos os Temas de Uso existentes na base de dados. Note-se que neste caso, se está a construir uma lista simples

de resultados sem detalhes acerca de cada um dos Temas de Uso. Pode-se aqui também observar que se está a aplicar um mapeamento entre o resultado do contexto de dados, que retorna um objeto de *DataModel*, e um objeto de *Contracts*. Ver-se-á mais adiante como foi parametrizado este mapeamento automático.

Por outro lado, quando é indicado um determinado ano nos parâmetros, tem-se então que este método vai considerar todos os Temas de Uso com publicações no ano indicado ou até ao ano indicado, de acordo com o segundo parâmetro *includeAllUpToYear*.

```
else
{
    //Year != 0 means we should restrict to the Themes of Use with
    //publications on/up to the specified year.
    IEnumerable<UseModel> usesModel;

    usesModel = from u in _serviceContext.Uses
                join ua in _serviceContext.UsesAccounts on u.Id equals ua.UseId
                join pa in _serviceContext.PapersAccounts on ua.AccountId equals pa.AccountId
                join p in _serviceContext.Papers on pa.PaperId equals p.Id
                where (includeAllUpToYear && p.PubYear <= year) || (!includeAllUpToYear && p.PubYear == year)
                select u;
```

Figura 11: Query LINQ para obtenção dos Temas de Uso no método GetAllUses

Na imagem acima, pode-se observar que se está a utilizar uma expressão LINQ (*Language Integrated Query*) para filtrar os resultados. Alternativamente, para evitar estes *joins* no código, poderia ter sido definida uma Vista diretamente na base de dados. Experimentou-se essa abordagem, mas tratando-se de uma base de dados SQLite notou-se uma significativa degradação do desempenho ao consultar essa vista. Por isso optou-se por manter a expressão no código. Para implementações futuras, em que a base de dados seja migrada para outro tipo de servidor dados como o SQL Server ou Oracle, seria interessante voltar a realizar estes testes de desempenho para compreender o que será mais vantajoso para o serviço.

Quando é especificado um determinado ano no parâmetro *year*, a lista de Temas de Uso vai conter informação adicional acerca de cada Tema de Uso encontrado, incluindo os seus detalhes como de publicações, autores e instituições.

```
uses = _mapper.Map<List<Use>>(usesModel.Distinct());

if(uses != null)
{
    //lets get the details for each Theme of Use found
    uses.ForEach(u => {
        IEnumerable<PaperModel> papersModel;

        papersModel = from p in _serviceContext.Papers
                      join pa in _serviceContext.PapersAccounts on p.Id equals pa.PaperId
                      join ua in _serviceContext.UsesAccounts on pa.AccountId equals ua.AccountId
                      where
                        (
                            (includeAllUpToYear && p.PubYear <= year) ||
                            (!includeAllUpToYear && p.PubYear == year)
                        ) && ua.UseId == u.Id
                      select p;

        if (papersModel != null)
        {
            u.Papers = new List<PaperDetails>();
            papersModel.ToList().ForEach(p =>
            {
                //this will cascade getting the paper details, authors and institutions
                u.Papers.Add(GetPaperDetailsById(p.Id));
            });
        }
    });
}
```

Figura 12: Obter os detalhes sobre cada Tema de Uso encontrado no método GetAllUses

No final, resta apenas carregar os detalhes da dimensão de imersão para cada Tema de Uso encontrado.

```
if (uses != null)
{
    uses.ForEach(u =>
    {
        //for each theme of use, we get its immersion classification
        var usesClassification = _serviceContext.UsesImmersionClassification
                                    .Where(c => c.UseId == u.Id)
                                    .FirstOrDefault();

        if (usesClassification != null)
        {
            u.SystemImmersion = usesClassification.SystemImmersion;
            u.NarrativeImmersion = usesClassification.NarrativeImmersion;
            u.ChallengeImmersion = usesClassification.ChallengeImmersion;
            u.RGBColor = usesClassification.RGBColor;
        }
    });
}
return uses;
```

Figura 13: Obter os detalhes sobre a dimensão de imersão de cada Tema de Uso encontrado no método GetAllUses

4.2.6 Ponto de arranque e controladores

O projeto *ImmersiveLearningAPI*, corresponde ao *front-end* do serviço com o utilizador e é o projeto de arranque da solução. São aqui definidas todas as configurações iniciais que permitem a correta execução do serviço. Vai-se então começar por dar uma breve explicação acerca das configurações mais relevantes do serviço, antes de passar aos controladores disponíveis.

As configurações do serviço estão distribuídas, de acordo com a sua área de configuração, entre os ficheiros *appsettings.json*, *AutoMapperServiceProfile.cs* e o ficheiro de inicialização *Startup.cs*.

No ficheiro de configuração *appsettings.json*, podem-se encontrar as configurações acerca da localização da base de dados e credenciais de segurança.

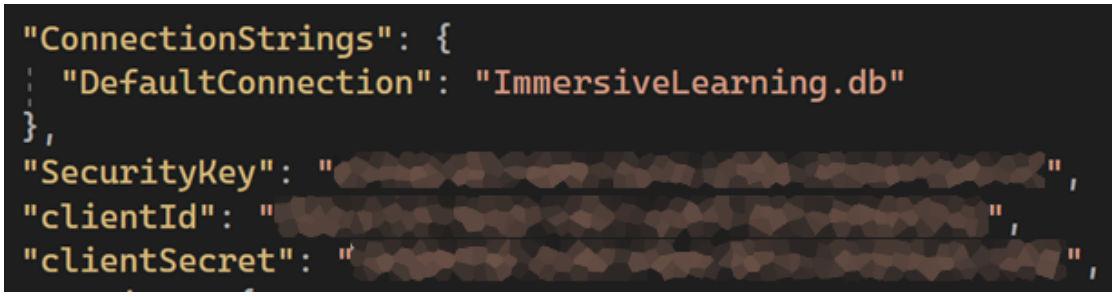


Figura 14: appsettings do WebAPI

O ficheiro da base de dados SQLite está a ser distribuído na raiz do serviço com o nome *ImmersiveLearning.db*, tal como se pode ver no parâmetro *ConnectionStrings*. É a única conexão a uma base de dados neste momento, sendo a conexão por defeito, mas alterações futuras, com necessidade de outras conexões, devem ser aqui acrescentadas.

Foram ainda aqui incluídas configurações de segurança que irão permitir a autenticação de clientes do serviço e geração dos respetivos *tokens* de autenticação. Dada a natureza sensível destas configurações, deveriam, idealmente, ser injetadas por um pipeline que realize a publicação do serviço. Após ponderação, considerou-se que não existia essa necessidade no âmbito deste projeto, dado o contexto restrito, mas implementações futuras devem avaliar e considerar essa necessidade, face ao grau de criticidade.

No ficheiro de configuração *AutoMapperServiceProfile.cs*, podem-se encontrar as definições de mapeamentos automáticos entre as classes dos projetos da solução *DataModels* e *Contracts*. Este mapeamento automático é conseguido por recurso à biblioteca *AutoMapper* (*AutoMapper.Extensions.Microsoft.DependencyInjection V11.0.0*) [Automapper 2022] que se considerou ser bastante útil, seja para evitar erros de codificação, seja simplesmente para evitar tarefas repetitivas e entediantes de cópia de valores entre classes.

```
public class AutoMapperServiceProfile : Profile
{
    0 references | Ari Silva, 29 days ago | 1 author, 4 changes
    public AutoMapperServiceProfile()
    {
        CreateMap<AccountModel, Account>();
        CreateMap<PaperModel, Paper>();
        CreateMap<PaperModel, PaperDetails>();
```

Figura 15: Configuração do AutoMapper

O ficheiro de configuração do mapeamento das classes pode ser encontrado na integra na secção dos Anexos, mas apenas a título de exemplo, podemos ver na imagem acima

o mapeamento de 3 classes do projeto *DataModels* em 3 classes do projeto *Contracts*. Quando este mapeamento entre classes não for um mapeamento direto, ou seja, o nome das propriedades não seja coincidente, ou exista a necessidade de algum tipo de transformação dos antes ao mapear os objetos, configurações adicionais são permitidas.

Finalmente, o ficheiro *Startup.cs* corresponde ao ponto de início e onde todas as configurações se tornam efetivas. Como foi já referido, e se verá em detalhe na secção seguinte, o serviço está protegido por um *JSON token* que deve ser gerado a pedido do cliente e mediante a validação das suas credenciais de autenticação. Para aceder aos recursos disponíveis, o cliente deve sempre incluir um *Bearer token* como parâmetro de autenticação no cabeçalho dos pedidos ao servidor. O serviço, encarrega-se então de validar o *token* recebido, de acordo com a configuração por nós definida no método *ConfigureServices*.

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            RequireExpirationTime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = "[illegible]",
            ValidAudience = "[illegible]",
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["SecurityKey"])),
            ClockSkew = TimeSpan.FromSeconds(1),
        };
    }
);
```

Figura 16: Configuração do token de autenticação

Esta configuração deve coincidir com a que tenha sido usada originalmente para a geração do *token* e utilizando a mesma chave simétrica fixa. De notar que esta validação incide apenas sobre a autenticação do cliente e não sobre a sua autorização de acesso aos recursos, ou seja, as suas *Claims*. Apesar de ter sido considerado um mecanismo de autorização dos clientes, decidiu-se que não seria relevante a sua implementação neste projeto. Fica, no entanto, como uma sugestão para futuras implementações.

O mecanismo de *dependy injection* [DI 2022] é também configurado no método *ConfigureServices*.

```
//Setting up dependency invection
services.AddDbContext<ServiceContext>(x =>
    x.UseSqlite($"Data Source={_appHost.ContentRootPath}/{Configuration.GetConnectionString("[illegible]")}")
);
services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
services.AddScoped<IServiceContext, ServiceContext>();
services.AddScoped<IServiceDomain, ServiceDomain>();
services.AddApplicationInsightsTelemetry();
```

Figura 17: Configuração da dependency injection

Começa-se por injetar a configuração da ligação à base de dados no construtor do contexto de dados, obtendo o nome do ficheiro da base de dados a partir do ficheiro *appsettings.json*. Depois são carregadas as configurações para o mapeamento automático das classes, que está definido no ficheiro *AutoMapperServiceProfile.cs* e

finalmente indica-se, para cada interface, qual é a classe de implementação dessa interface.

Ao nível das configurações, foi ainda necessário parametrizar o *CORS*, para prevenir que a aplicação cliente a correr num b*rowser* da Internet, não seja bloqueado ao tentar comunicar com o serviço.

```
//Making sure the browser allows the web app talking to this service
services.AddCors(options =>
{
    options.AddDefaultPolicy(
        builder =>
        {
            builder.AllowAnyOrigin()
                   .AllowAnyHeader()
                   .AllowAnyMethod();
        });
});
```

Figura 18: Configuração do CORS

4.2.7 Autenticação

Tal como já foi referido, a utilização de cada um dos recursos está protegida por um *JSON token*, que deve ser enviado como parâmetro de autenticação no *header* do *request*. O token de autenticação deve ser obtido no *recurso api/token*, que o irá gerar e devolver ao cliente, após correta validação das credenciais do cliente, que terá a duração de 60 minutos.

Para proteger os recursos e forçar os clientes à utilização do *token* de autenticação, inclui-se o atributo *Authorize*, (do *namespace Microsoft.AspNetCore.Authorization*) no recurso que se pretendeu proteger.

```
[Authorize(AuthenticationSchemes = "[redacted]")]
[Route("[redacted]")]
[ApiController]
[redacted]
public class AuthorsController : Controller
{
```

Figura 19: Protegendo um recurso do serviço WebAPI

A especificação do código utilizado na geração e validação do *token* de autenticação do serviço pode ser consultado na secção dos anexos.

4.2.8 Recursos

Os recursos desenvolvidos e disponíveis no serviço WebAPI, correspondem, genericamente, às Entidades da base de dados, existindo diferentes operações em cada recurso, de acordo com as necessidades próprias do projeto.

api/token (POST)

Este recurso é responsável pela validação das credenciais do cliente e geração do respetivo *token* de autenticação. Este é o único recurso que não está protegido, para permitir que fique acessível a qualquer cliente. Disponibiliza apenas uma operação *POST* que deve receber no corpo do *request* uma mensagem em *JSON* incluindo o *clientId* e *clientSecret* do cliente que permite obter o *token*. Esta operação responde com um *ActionResult* que pode ser *200 OK* com o respetivo *token* quando o processamento do pedido se tiver concluído corretamente, *400 Bad Request* com a respetiva descrição de erro, quando a mensagem estiver mal formatada, ou *401 Unauthorized* quando as credenciais do cliente estiverem incorretas.

Quando as credenciais de acesso especificadas pelo cliente forem corretas, esta operação inicia a geração do respetivo *token* de autenticação utilizando a chave simétrica fixa que está especificada no ficheiro de configuração do serviço. O *token* é então gerado por recurso à utilização da classe *JwtSecurityToken* (*namespace System.IdentityModel.Tokens.Jwt*). A implementação do código deste recurso pode ser consultada na secção dos anexos.

api/accounts (GET)

Este recurso representa a entidade *account* da base de dados e devolve uma lista com todos os valores da tabela.

api/authors (GET)

Este recurso representa a entidade *author* da base de dados, e devolve uma lista com todos os autores de publicações consideradas no estudo base deste projeto.

api/authors/{authorId} (GET)

Esta operação permite obter os detalhes de um autor, incluindo a sua lista de instituições.

api/institutions (GET)

Este recurso representa a entidade *institution* da base de dados, que corresponde às instituições dos autores das publicações consideradas no estudo base deste projeto.

api/papers (GET)

Este recurso representa a entidade *paper* da base de dados, que corresponde às publicações consideradas no estudo base deste projeto.

api/papers/{paperId} (GET)

Esta operação permite obter os detalhes de um artigo analisado, incluindo a sua lista de autores e respetivas instituições.

api/uses (GET)

Este recurso representa a entidade *uses* da base de dados, que corresponde aos Temas de Uso das publicações consideradas no estudo base deste projeto. Devolve uma lista com todos os valores da tabela.

api/uses/years (GET)

Este recurso devolve uma lista com todos os anos das publicadas associadas a Temas de Uso.

4.3. UI FRONT-END (UNITY)

Este sistema assume um papel de relevo na solução final, uma vez que é aquele com que o utilizador interage mais diretamente. É aqui que resolvemos uma grande parte dos requisitos identificados no Capítulo 2 deste relatório, no que respeita ao aspeto e interações do utilizador. Veremos, ao longo desta secção, como utilizamos o UI de desenvolvimento *Unity*, para proporcionar ao utilizador uma experiência imersiva, e de alguma forma inovadora, para que ele próprio consiga interagir com os dados que está a analisar. Este sistema está neste momento publicado e disponível no URL https://temporal-analysis.azurewebsites.net/.

O *Unity*, é uma plataforma de desenvolvimento de jogos 2D e 3D, grafismos em tempo real, filmes 3D, ferramentas de simulação ou realidade mista, unindo caraterísticas da realidade virtual com a realidade aumentada. O *Unity* utiliza a framework de desenvolvimento .Net para permitir a escrita de *scripts* C#, que podem ser diretamente editados no *Visual Studio®*, e permite a compilação dos projetos para serem executados em diferentes plataformas como *Windows®, Android®, IOS®* ou *WebGL*, bem como dispositivos de Realidade Virtual (VR) ou Realidade Aumentada (AR), entre muitas outras plataformas.

A base do sistema UI, foi criada a partir do template para a criação de projetos 3D, disponível no *Unity* Hub. A imagem seguinte, representa o aspeto do ambiente de desenvolvimento no *Unity*, quando foi criado um novo projeto 3D.

Uma das primeiras decisões que foi preciso tomar quando se iniciou um projeto no *Unity*, foi acerca da plataforma onde se pretendeu que o projeto fosse executado. No presente caso, existe um requisito que impõe que a execução seja possível sobre *WebGL*, e, portanto, deve-se desde logo realizar essa configuração do projeto na opção *Build Settings* do *Unity*.

O template de criação de projetos 3D, adiciona automaticamente uma *scene (SampleScene)* ao projeto, que funciona como uma espécie de contentor de todos os objetos que vão fazer parte da cena. Habitualmente, em projetos Unity, outras *scenes* costumam ser adicionadas para facilitar a organização do projeto, mas, neste caso, não existiu essa necessidade, decorrendo toda a execução numa única *scene*. Optou-se por manter essa *Scene* e apenas a renomear para *ImmersiveLearning*. O template de criação, adicionou ainda à *scene* os objetos *Main Camera* e *Directional Light*. Uma vez que se pretendeu simular um espaço visualmente imersivo, para a apresentação das formas, foi

preciso alterar o fundo da câmara para uma cor sólida. Alterou-se por isso a propriedade *Clear Flags* de *Skybox* para *Solid Color* e depois modificou-se o fundo para a cor preta RGBA = (0,0,0,0). No presente caso não é preciso utilizar a luz direcional nos objetos, pelo que se optou por desativá-la, e também não é preciso acrescentar qualquer outra câmara. A câmara é um elemento fundamental no *Unity*, uma vez que é a através dela que o utilizador pode ver o que se vai desenrolando na cena. Pode-se pensar na câmara como se fosse, de facto, uma câmara de gravação de um filme. O resultado dessa filmagem é apresentado numa tela que o utilizador pode visualizar, só que aqui, vai existir a possibilidade de ser o próprio utilizador a controlar o local onde a câmara está posicionada e para onde está a ser direcionada.

O desenvolvimento de projetos no *Unity* consiste, essencialmente, em adicionar objetos à cena, ajustar as propriedades desses objetos e controlar as suas variáveis e propriedades. Um pormenor importante a referir aqui, e que deve estar sempre presente quando se desenvolve um projeto em *Unity*, é que os objetos possuem componentes, que são no fundo aquilo que define o próprio. Ora, um dos componentes que está sempre presente em todos os objetos é o componente *Transform*, que permite definir as suas propriedades relativas à posição, rotação e dimensões. O pormenor importante a ter sempre presente, é que esta posição se trata da posição relativa do objeto, em relação ao seu objeto *parent*, e não a sua posição absoluta na *scene*. Suponha-se, por exemplo, que se tem o objeto Y que é um *child* do objeto X. Quando se altera a posição relativa do objeto Y, o *Unity* calcula automaticamente a sua posição absoluta. E o mesmo acontece quando se altera a posição do objeto *parent* X, preservando-se a posição relativa do objeto *child* Y, mas sendo automaticamente calculada a sua nova posição absoluta.

4.3.1. Referencial 3D

O requisito 7 (ver secção 2.2) identifica que os Temas de Uso devem estar posicionados em coordenadas no espaço conceptual da imersão, pelo que os primeiros objetos que se adicionam à *scene*, servem exatamente para construir um referencial 3D, que se irá usar mais tarde para posicionar os Temas de Uso, tal como se pode observar na imagem seguinte.

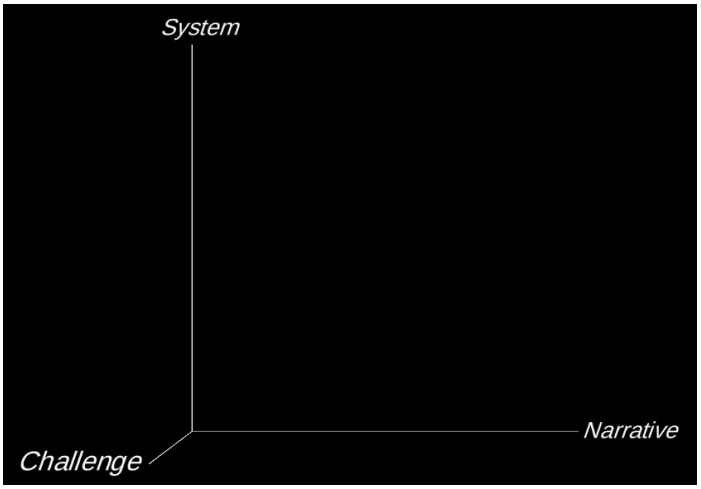


Figura 20: Referencial 3D do espaço conceptual da imersão

Por se tratar de um referencial 3D, vai apresentar 3 eixos de coordenadas, que foram definidos de acordo com o requisito 7 e a tabela 4, "*Classification of use themes per immersion dimension*" do artigo-base citado [Beck, Morgado and Shea 2020]. Ver-se-á mais adiante que a escala do referencial pode ser ajustada pelo utilizador, mas decidiu-se que, como configuração inicial por defeito, todos os eixos devem apresentar uma dimensão de 50 unidades. Para a criação dos eixos, foi preciso então adicionar 3 objetos vazios à *scene*, e a cada um dos objetos adicionou-se no *Inspector* o componente *Line Renderer*. Este componente permite criar uma linha reta entre 2 pontos do espaço. Para cada um dos objetos, o primeiro ponto deve ficar posicionado na coordenada de origem (x,y,z) = (0,0,0) e o segundo ponto ficará a uma distância de 50 unidades, de acordo com o próprio eixo. Assim, a segunda coordenada do eixo *Narrative*: (x,y,z) = (50,0,0), *System:* (x,y,z) = (0,50,0) e *Challenge:* (x,y,z) = (0,0,50). É ainda necessário adicionar ainda uma *label* a cada um dos eixos, para que o utilizador compreenda o significado dos eixos, no espaço conceptual da imersão. Estas *labels* são adicionadas como objetos *child* de cada eixo, permitindo assim beneficiar do posicionamento local relativamente ao seu eixo. Ver-se-á mais adiante como isto é útil quando se tratar da rotação do referencial.

Na imagem ao lado, pode-se ver como os objetos ficam aninhados uns nos outros, de acordo com o seu "grau" de parentesco. O *parent* do referencial é o objeto designado *3DAxis*. Como objetos-filhos tem-se os 3 eixos, *DimensionX, DimensionY* e *DimensionZ*. Pode-se então observar que cada um dos eixos tem ainda os seus próprios objetos aninhados, que são utilizados para exibir a *label* do eixo. Note-se que, uma vez que os objetos estão aninhados, qualquer operação de translação, rotação vai afetar as suas posições locais. Se, por exemplo, se decidir transladar o objeto *3DAxis* em 3 unidades em direção à ordenada X, todos estes objetos serão transladados 3 unidades em direção à mesma ordenada, preservando-se a sua posição local, mas sendo automaticamente recalculada a posição global na *scene*.

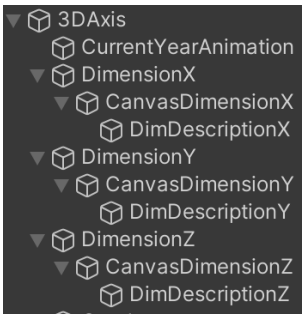


Figura 21: Estrutura do eixo 3D

4.3.2. Filtros

O requisito 4 (ver secção 2.2), define que o sistema deve permitir a seleção de um conjunto de filtros que permitam limitar os resultados da análise. Para responder a este requisito, adicionou-se um painel lateral, com um conjunto de filtros com os quais o utilizador pode interagir.

Na imagem ao lado, pode-se observar que foram adicionados os seguintes critérios de filtragem:

- *Uses*: permite responder diretamente aos Casos de Uso identificados para o projeto final (ver secção 2.1). Os valores disponíveis são mutuamente exclusivos e apenas um dos valores pode ser selecionado de cada vez;
- *Years*: permite selecionar o(s) ano(s) em que se pretende analisar os Temas de Uso. Um ou vários anos podem ser selecionados;
- *Authors*: permite selecionar o(s) autor(es) dos artigos considerados nas publicações analisadas. Um ou vários autores podem ser selecionados;
- *Institutions*: permite restringir a análise apenas às instituições selecionadas. Uma ou várias instituições podem ser selecionadas.

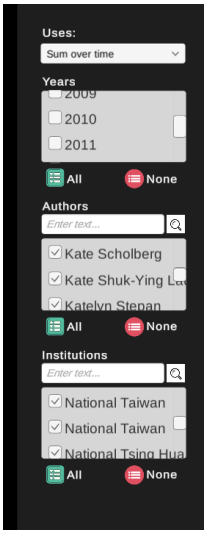


Figura 22: Filtros da análise

Veja-se então como foram criados estes filtros e como estão a ser carregados os seus valores iniciais.

O filtro *Uses* é um objeto do tipo *ComboBox*, cujos valores foram previamente introduzidos nas suas propriedades. A escolha deste tipo de objeto permite que, no futuro, outros valores possam ser acrescentados, facilitando assim futuras alterações ao projeto. Como referido, este filtro permite resolver os 2 Casos de Uso identificados na secção 2.1. Quando se seleciona o valor "*Specific to year*", são apresentados Temas de Uso, considerando apenas as publicações nos anos especificados, respondendo desta for ao Caso de Uso 2, ou seja, "Quais foram os usos publicados num ano?". Por outro lado, selecionando o valor "*Sum over time*", a construção dos Temas de Uso leva em consideração todas as publicações até ao ano considerado. Ver-se-á mais adiante alguns cenários de utilização que podem ajudar a compreender a seleção dos filtros.

O filtro *Years* consiste num conjunto de caixas de seleção, onde o utilizador pode restringir os anos de análise das publicações de Temas de Uso. Estas caixas de seleção, estão a ser criadas dinamicamente aquando da inicialização do sistema, por uma chamada assíncrona ao serviço WebAPI. Mais adiante neste relatório, ver-se-á em mais detalhe como este sistema comunica com o serviço WebAPI. Este filtro, como os restantes, possui dois botões que podem ser selecionados para marcar e desmarcar todos os elementos do filtro, neste caso os anos. Na imagem à direta pode-se observar a estrutura, uma vez mais uma estrutura aninhada, dos principais objetos que é preciso adicionar para criar este filtro.

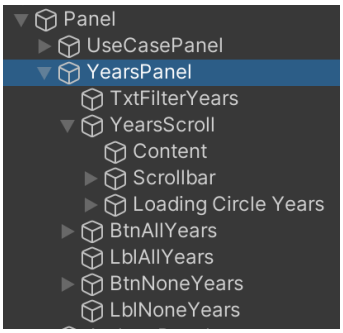


Figura 23: Estrutura do filtro Years

Os dois filtros seguintes, *Authors* e *Institutions*, consistem, tal como o filtro anterior, num conjunto de caixas de seleção, onde o utilizador pode restringir a sua análise apenas aos autores e às instituições pretendidos das publicações analisadas. Estas caixas de seleção estão a ser criada dinamicamente aquando da inicialização do sistema, por uma chamada assíncrona ao serviço WebAPI.

Nas imagens seguintes, a imagem da esquerda corresponde à estrutura de objetos do filtro *Authors* e a imagem da direita do filtro *Institutions*.

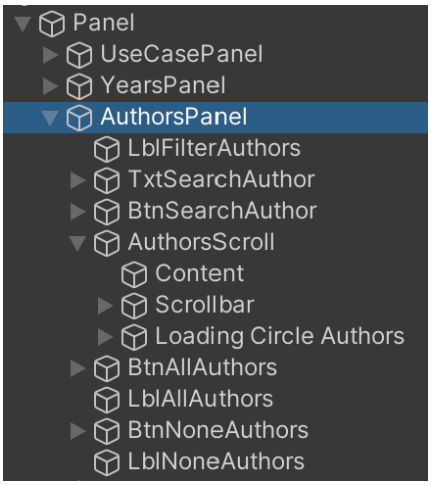


Figura 24: Estrutura do filtro Authors

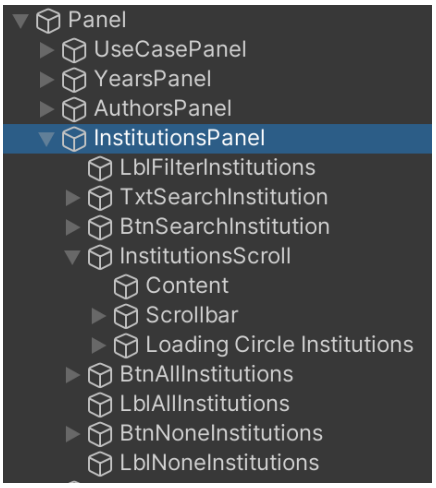


Figura 25: Estrutura do filtro Institutions

4.3.3. Controlo da Câmara e Movimento

No mesmo painel lateral, pode-se ainda encontrar um conjunto de controlos por permite ao utilizador rodar o referencial e fazer o *zoom-in/out* da câmara, no próprio ecrã. Pode-se ver na imagem seguinte a legenda dos controlos que permitem interagir com a representação do espaço conceptual da imersão.

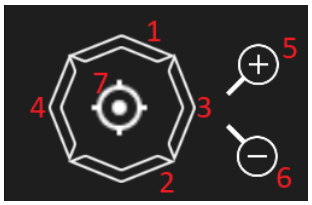


Figura 26: Controlos da Câmara no ecrã

1. Roda em torno da dimensão +Narrative
2. Roda em torno da dimensão -Narrative
3. Roda em torno da dimensão +System
4. Roda em torno da dimensão -System
5. Zoom-in
6. Zoom-out
7. Centra a apresentação no ecrã

Na figura seguinte, pode-se observar como foi também aqui criada uma estrutura de objetos aninhados que permite tirar partidos das propriedades dos objetos *parent*.

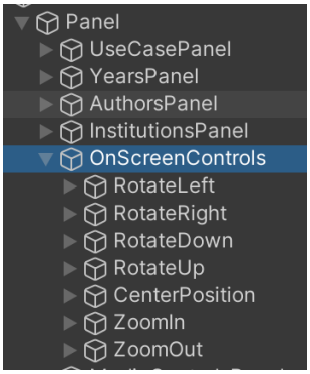


Figura 27: Estrutura dos controlos da Câmara no ecrã

Para além destes controlos no ecrã, o utilizador pode ainda utilizar o rato para realizar as seguintes interações com o espaço de imersão:

- Botão esquerdo do rato: Deslocação do referencial
- Botão direto do rato: Rotação do referencial
- Botão de *scroll* do rato: *Zoom-in/out*

A tecla Ctrl aumenta a velocidade das funções de *zoom-in/out*. Os controlos no ecrã tornam-se muito úteis quando se utiliza dispositivos móveis para interagir com a apresentação, como é o caso dos telemóveis ou *tablets*. Caso contrário, utilizar apenas o rato nestes dispositivos seria, de todo, impraticável. Ver-se-á mais adiante como foram implementados estes controlos através de *scripts* em C#.

4.3.4. Controlos da Animação

Ainda no painel lateral, ao fundo, encontram-se os botões que permitem iniciar, pausar e terminar a apresentação dos resultados, na forma de uma animação.

Figura 28: Controlos da apresentação

1. Start – Inicia/continua a apresentação
2. Pause – Suspende a apresentação
3. Stop – Termina a apresentação

Depois de definir os parâmetros de análise nos filtros, o utilizador pode então dar início à apresentação. Nesse momento, são realizadas todas as tarefas de processamento necessária a criar os modelos de dados a apresentar, para cada um dos anos selecionados. Quando a apresentação é iniciada, o botão de início da apresentação fica desativado, ativando-se os botões de pausa e terminação. Pausando a apresentação, volta a ficar ativo o botão de início, neste caso servirá para retomar a animação. Além disso, quando se dá início à apresentação, são ainda apresentados no ecrã, 3 controlos adicionais que permitem alterar a escala do referencial, um controlo temporal que permite ao utilizador avançar ou recuar nos anos gerados e um controlo que permite alterar a velocidade da apresentação.

Figura 29: Controlos da animação

1. Scale – Controla a escala do referencial
2. Temporal Control – Controla a posição temporal da animação
3. Speed – Controla a velocidade da animação

4.3.5. C# Scripting

Nesta secção passa-se então a explicar como foi implementado o código C#, que permite controlar todos os objetos do sistema, a sua inicialização, interações do utilizador e controlo de animação. Para evitar a confusão de conceitos, adota-se como convenção que, quando seja referido o termo objeto se está a referir a um elemento visual do *Unity* e não ao resultado de uma instanciação de uma classe.

No IDE do *Unity*, existe um separador *Project*, onde se pode encontrar toda a estrutura de ficheiros do projeto. Dentro da pasta *Assets*, cria-se outra pasta que se designa *Scripts*, e onde, por uma questão de organização do projeto, se criam todas as scripts C#. No *Unity*, uma script C# pode ficar associada a um objeto, adicionando essa script como mais um componente desse próprio objeto.

Quando uma script C# é criada pelo no *Unity*, é gerada uma classe que herda da classe base *MonoBehaviour* (*namespave UnityEngine.MonoBehaviour*). Esta classe *MonoBehaviour* vem do próprio *Unity*, e disponibiliza um conjunto de métodos pré-definidos, eventos e comportamentos determinados pelo *Unity*. Dois desses métodos, *Start()* e *Update()*, são gerados automaticamente nas classes do *Unity*. O método *Start()* é evocado aquando da inicialização do objeto a que essa script está acoplada, e atua como se fosse o construtor da própria classe, pelo que é o sítio ideal para que se realizem as tarefas de inicialização. Por outro lado, o método *Update()* é evocado a cada *frame* gerada desse objeto.

Foi isso o que se fez para o objeto *3DAxis*, associando a script *init*, que funciona como o ponto de arranque do sistema, a partir do qual se realizam todas a inicializações necessárias.

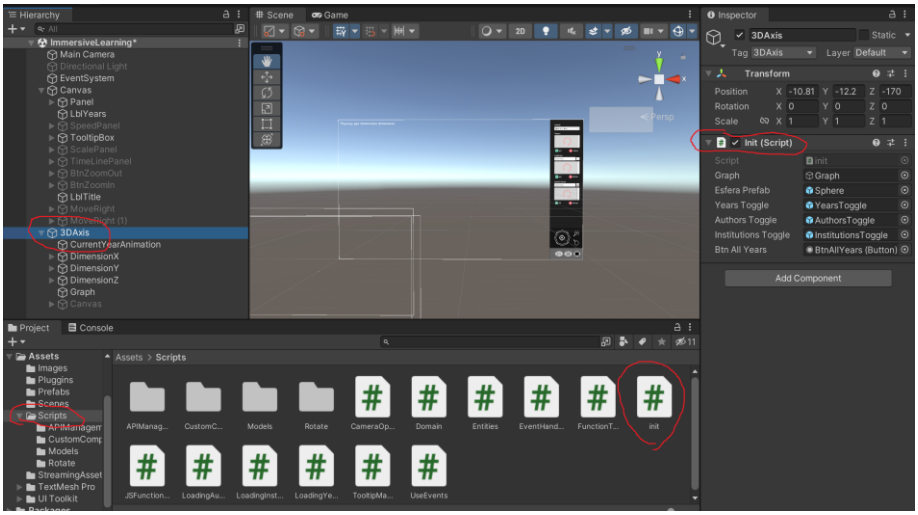


Figura 30: Ponto de início do sistema

4.3.5.1. Estrutura do projeto C# do Unity

Por questão de organização da estrutura do projeto, decidiu-se criar pastas para agrupar conjuntos de scripts que partilhem a mesma lógica. Na pasta *APIManagement* pode-se encontrar código relativo à interação com o serviço WebAPI. A classe *APIHelper*, implementa já um conjunto de métodos que ajudam a realizar chamadas ao serviço de uma forma simples e padronizada. Na verdade, esta classe disponibiliza apenas um método público, *SendAPIRequest*, para ser evocado pelo consumidor do serviço e um evento que, por ser tratar de uma chamada assíncrona, vai informar quando o pedido termina. Toda a lógica de autenticação, requisitando o *Bearer token* e a sua inclusão no cabeçalho do pedido, é realizada de forma automática para que, sempre que se precisar de realizar chamadas ao serviço se possa focar apenas nessas chamadas e não em todos os mecanismos que isso envolve. Como sugestão para futuras alterações/implementações neste projeto, as credenciais de autenticação, *clientId* e *clientSecret*, foram adicionadas ao próprio código. No presente caso, entende-se que essa abordagem é aceitável e suficiente, mas no futuro deve ser ponderado se isto ainda faz sentido, podendo essas credenciais ser movidas para, por exemplo, um ficheiro de configuração ou até mesmo serem injetadas nas classes aquando de uma publicação automática do projeto por um pipeline.

Dentro da pasta *Models*, decidiu-se concentrar todos os modelos de dados que se vão utilizar em todo o projeto. Alguns destes modelos de dados são apenas uma replicação daqueles que já haviam sido definidos no serviço WebAPI. Recorde-se que o serviço WebAPI retorna objetos serializados e não objetos, pelo que, após uma chamada bem-sucedida ao serviço, deve ocorrer uma desserialização dos resultados para que possam ser mais facilmente utilizados e manipulados. Existe ainda outro tipo de modelos de dados que foram aqui acrescentados e que servem apenas o propósito deste projeto.

Dentro da pasta *CustomComponents*, foram criados também modelos de dados, mas neste caso, servem um propósito muito específico. Como já se referiu, as scripts C# podem ficar acopladas aos objetos do *Unity*, sob a forma de componentes desses objetos.

Existe, no entanto, um requisito para que isso aconteça. Essas classes precisam sempre de herdar de *MonoBehaviour*. A motivação para se ter decidido criar estas estruturas de dados, acopladas a objetos *Unity*, vem da necessidade de manter o bom desempenho e fluidez do sistema, acedendo diretamente à informação que diz respeito a esse objeto em particular. Por exemplo, quando é gerado um modelo com um Tema de Uso, que para o *Unity* corresponda a um objeto *Sphere*, é-lhe adicionado um componente que corresponde à classe *UseCompoenet*, e que detém toda a informação que diz respeito a esse Tema de Uso em particular. Como se verá mais adiante, quando se passa com o ponteiro do rato por cima de um Tema de Uso, é mostrada uma *label* com alguns detalhes. Ao se deter esta informação diretamente no objeto, consegue-se significativamente agilizar o processo de geração dessa *label* em tempo real.

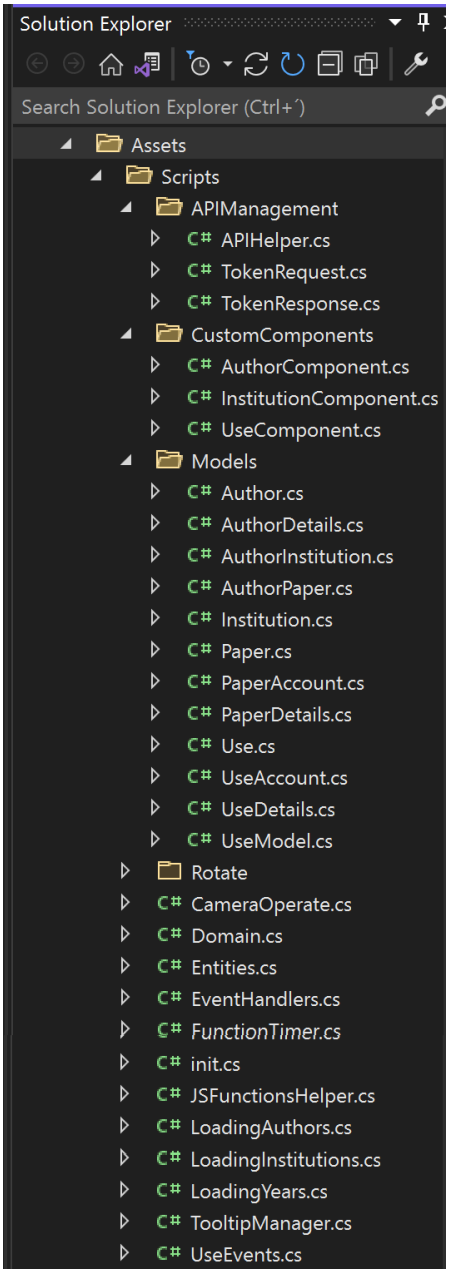


Figura 31: Estrutura do projeto C# do Unity

Pode-se ainda observar na estrutura do projeto, duas outras classes muito relevantes, *Domain* e *Entities*. Estas classes são absolutamente fundamentais a todo o projeto, pois é aqui que se situa a lógica de construção dos modelos que são apresentados na animação e, portanto, permitem implementar os casos de uso referidos na secção 2.1 deste relatório. A classe *Entities* trata-se de uma classe estática, acessível a todo o projeto, que permite aceder ao conjunto de entidades carregadas a partir da base de dados, bem como ao domínio do nosso projeto onde se pode encontrar o método *GenerateModels*.

Outras classes que se podem observar na estrutura do projeto correspondem a classes que herdam de *MonoBeaviour* e estão diretamente acopladas aos objetos *Unity*, como componentes desses mesmos objetos.

4.3.5.2. Inicialização do sistema

Como já se referiu acima, considera-se o ponto de arranque do sistema a partir da classe *init*, que está acoplada ao objeto *3DAxis*. Quando este objeto é inicializado, é então evocado o método *Start()*, que realiza pedidos assíncronos ao serviço WebAPI para receber informação acerca das entidades que necessitam de ser carregadas no sistema. É a partir destes dados recebidos que se vai inicializar os filtros *Years, Authors* e *Institutions*. Uma vez que estas chamadas são assíncronas, não se sabe quando elas terminam e, portanto, depende-se do evento lançado pelo método que acede ao serviço para informar o seu término. Para garantir que o utilizador está informado acerca deste processo de inicialização do sistema, criou-se o próprio controlo *spinner*, que foi adicionado a cada um destes filtros e controlado individualmente.

4.3.5.3. Inicialização dos filtros

Para analisar como são criados os valores dos filtros, veja-se, como exemplo, o método *AddAuthors()*, da classe *init*, que adiciona as caixas de seleção de autores ao respetivo filtro.

```
/// <summary>
/// Loads all authors to the corresponding filter
/// </summary>
1 reference
private void AddAuthors()
{
    try
    {
        //Find container to values
        GameObject authorsContent = GameObject.FindGameObjectWithTag("AuthorsContent");

        if(authorsContent != null)
        {
            //Iterate through the entity and add a checkbox to each value
            foreach (Author author in Entities.Authors.OrderBy(a => a.Name))
            {
                //Create a new checkbox
                GameObject newToggle = (GameObject)Instantiate(authorsToggle);
                newToggle.GetComponentInChildren<Text>().text = author.Name;

                //Set the parent of the newly checkbox to the container
                newToggle.transform.SetParent(authorsContent.transform, false);

                //Saving the value's id to be used later when applying the filter
                AuthorComponent authorComponent = newToggle.GetComponent<AuthorComponent>();
                authorComponent.AuthorId = author.Id;
                authorComponent.AuthorName = author.Name;
            }

            //Loading is conplete and we can stop the spinning
            Entities.LoadingAuthors = false;
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Figura 32: Método para a construção do filtro Authors

Este método começa por procurar o objeto que irá conter as caixas de seleção relativas aos autores. Depois, realizam-se as iterações necessárias na entidade correspondente, que já terá sido previamente inicializada a partir do serviço WebAPI, e por cada autor adiciona uma nova caixa de seleção ao contentor. Note-se que se está aqui a utilizar uma das classes *custom*, *AuthorComponent*, que se tinha visto em cima na estrutura. Mais tarde, esta classe vai permitir identificar, de forma única, cada elemento selecionado no contentor. No final das iterações pode-se dar o processo como concluído e interromper o *spinner*, dando a indicação ao utilizador que o filtro está pronto a ser utilizado.

Este filtro permite ainda que o utilizador procure autores pelo nome, para facilitar a aplicação de restrições da análise. Para tal, o utilizador deve introduzir o nome a procurar e depois selecionar o botão de lupa. A função que trata o evento *Click* do botão lupa, *BtnSearchAuthor_Click()*, pode ser encontrada na classe *EventHandlers* e vai-se aqui analisar esta função, apenas a título de exemplo. O carregamento dos restantes filtros funciona de forma muito semelhante. Este código, assim como o de todo o projeto, pode ser consultado nos anexos.

```
/// <summary>
/// User is trying to filter by a specific value
/// </summary>
0 references
public void BtnSearchAuthor_Click()
{
    //This search is only active when presentation is not in progress
    if (!started)
    {
        //Find the container os values to be filter
        GameObject authorsContent = GameObject.FindGameObjectWithTag("AuthorsContent");

        //Find the filter field
        GameObject searchAuthors = GameObject.FindGameObjectWithTag("SearchAuthors");
        TMPro.TMP_InputField filterValue = searchAuthors.GetComponent<TMPro.TMP_InputField>();

        //Iterate through values of container to apply the filter
        foreach (Transform child in authorsContent.transform)
        {
            AuthorComponent author = child.GetComponent<AuthorComponent>();

            if (filterValue.text.Equals(""))
            {
                //The filter value is empty meaning the user wants to clear all previous selection
                child.gameObject.SetActive(true);
            }
            else if (author != null && author.AuthorName != null)
            {
                //The checkbox is set to active/inactive according to the filter criteria
                child.gameObject.SetActive(
                    author.AuthorName.ToUpper().Contains(filterValue.text.ToUpper())
                );
            }
        }
    }
}
```

Figura 33: Método que realiza a filtragem de Autores no próprio filtro

Esta filtro funciona apenas quando não existe qualquer apresentação a decorrer, que se controla a partir da variável *started*, como se verá mais adiante. Começa-se então por procurar a referência para o contentor de valores a serem filtrados e outra referência para o valor a filtrar. Percorre-se então, iterativamente, cada um dos valores do contentor e oculta-se/mostra-se os valores filtrados, utilizando a propriedade *SetActive*.

4.3.5.4. Geração dos modelos

A geração dos modelos a apresentar é iniciada quando o utilizador pressiona o botão *Start* no ecrã, cujo evento *Click* é tratado em *EventHandlers.BtnPlay_Click()*. Neste momento, todos os filtros devem ter já sido parametrizados pelo utilizador, uma vez que se vai utilizar estes filtros para restringir os Temas de Uso gerados nos modelos. Começa-se então por recolher todos os parâmetros selecionados, i.e., anos, autores, instituições e Caso de Uso. Estes valores são então enviados ao método *GenerateModels*, na classe *Domain*, que vai tratar de gerar os modelos. A geração dos modelos consiste numa consulta às Entidades, previamente carregadas a partir do serviço WebAPI. Nessa consulta, começa-se por procurar Temas de Uso para cada ano selecionado, considerando os filtros dos autores e instituições. Esta consulta permite resolver os dois Casos de Uso possíveis, considerando apenas publicações daquele ano, ou todas as publicações até ao ano em análise. Uma vez encontrados os Temas de Uso de um ano, vai-se agrupá-los numa coleção de Temas de Uso do ano e guardar esse modelo numa outra coleção que vai conter, para cada ano, todos os Temas de Uso desse ano.

4.3.5.5. Inicio da apresentação

Uma vez gerados todos os modelos, a coleção desses modelos é enviada ao método *AddModel*, em *EventHandler*, que vai tratar de vai tratar de criar os objetos do *Unity* necessários à animação. Os Temas de Uso são representados na animação por esferas do *Unity*, que foram previamente definidas e guardadas para serem usadas como templates na pasta *Prefab*. Quando se está a criar uma nova esfera a partir desse template, é preciso definir o diâmetro da esfera, que corresponde ao número total de artigos publicados nesse ano (ou até esse ano, dependendo do Caso de Uso selecionado) e para esse Tema de Uso. É preciso ainda definir a cor do componente *Renderer* da esfera, a sua posição relativa no referencial e uma *label* que permite identificar o Tema de Uso. Adicionalmente, é adicionado um componente à esfera, *UseComponent*, que tem todos os dados que levaram à geração dessa esfera. Este componente serve dois propósitos. Quando se passa com o rato por cima de uma esfera, é apresentada informação adicional que incluí o número de artigos publicados, o número de autores e o número de instituições. Esta informação adicional está a ser obtida diretamente do componente *UseComponent* da esfera, evitando potenciais problemas de desempenho por novas consultas às entidades. O outro propósito deste componente é para a geração da página de detalhes do Tema de Uso, quando se pressiona com o rato sobre ele.

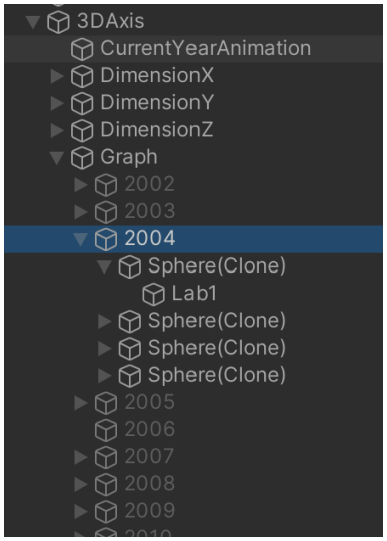


Figura 34: Estrutura de objetos do Unity gerada dinamicamente como os modelos de Temas de Uso

Depois de gerados todos os objetos do *Unity* para os modelos gerados, tornam-se então visíveis os controladores da escala, tempo e velocidade, e inicia-se então a animação que irá percorrer e apresentar todos os anos na estrutura de objetos.

4.3.5.6. Controlador da velocidade da animação

O intervalo de apresentação entre cada ano é, por defeito, 2 segundos. Definiu-se, no entanto, um controlador que o utilizador pode utilizar para alterar esta velocidade da animação. Uma vez o *Unity* não suporta múltiplas *Threads* em WebGL, existiu a necessidade de definir as funções de controlo do tempo. Estas funções podem ser consultadas na classe *FunctionTimer* e, essencialmente, desempenham uma ação ao fim de determinado tempo. Tem-se uma variável que define após quanto tempo deve ser desencadeada a ação. Esta variável é decrementada no método *Update* que é chamado a cada *frame* renderizada no ecrã. Quando o tempo se esgota, a ação é desencadeada e a variável é reinicializada para o tempo inicial. O controlo de velocidade no ecrã apenas controla esta variável do tempo inicial, aumentando ou diminuindo o seu valor, de acordo com a escolha do utilizador.

4.3.5.7. Controlador temporal da animação

O controlador temporal permite ao utilizador navegar por entre os anos dos modelos manualmente. Este processo de navegação consiste em mostrar e ocultar os objetos do *Unity* que foram previamente gerados a partir dos modelos. O controlador temporal manipula um índice que corresponde ao ano que está a ser mostrado na animação. Movendo o controlador temporal para a frente ou para trás, vai incrementar ou decrementar esse índice, definindo, portanto, qual é o ano atual a ser apresentado na animação.

4.3.5.8. Controlador da escala do referencial

O controlador da escala surgiu como uma necessidade de permitir ao utilizador o ajuste da escala do referencial. Dependendo do ano em análise, pode acontecer que alguns Temas de Uso fiquem demasiados próximos entre si no referencial e aconteça uma sobreposição das esferas. Aumentando a escala do referencial, pode-se obter uma visualização mais clara de cada Tema de Uso, evitando as sobreposições.

4.3.5.9. Controlador da Câmara

O controlador da câmara, definido na classe *CamaraOperate*, dá ao utilizador a capacidade de interagir com os modelos apresentados na animação. Este controlador permite girar o referencial em tornos dos eixos, deslocar o referencial no espaço, aproximar ou afastar a câmara do referencial e central o referencial na sua posição inicial.

Este controlador responde aos eventos do rato e aos controlos adicionados ao ecrã. Estes controlos no ecrã surgiram como necessidade de se poder continuar a interagir com os modelos, ainda que se esteja a utilizar dispositivos táteis como um telemóvel ou um *tablet*.

4.3. PÁGINA DE DETALHES DOS TEMAS DE USO

Para que o utilizador pudesse examinar mais em pormenor cada Tema de Uso apresentado na animação, foi necessário definir uma página html com JavaScript que exibisse todos os esses detalhes. Esta página é na verdade um template carregado dinamicamente em função do Tema de Uso selecionado.

Quando se pressiona com o rato em cima de um Tema de Uso, é gerado um JSON que inclui todos os detalhes como artigos, autores e instituições. Esse JSON é então guardado numa variável de sessão do browser com um identificador único, gerado nesse momento. O template HTML é então aberto num novo separador do browser, enviando-lhe como parâmetro no URL, o identificador único do JSON gerado.

Quando o template é aberto, é carregado, a partir das variáveis de sessão, o JSON que corresponde ao identificador único recebido no URL, de onde são carregados todos os detalhes necessários para gerar uma página HTML que apresente esses detalhes.

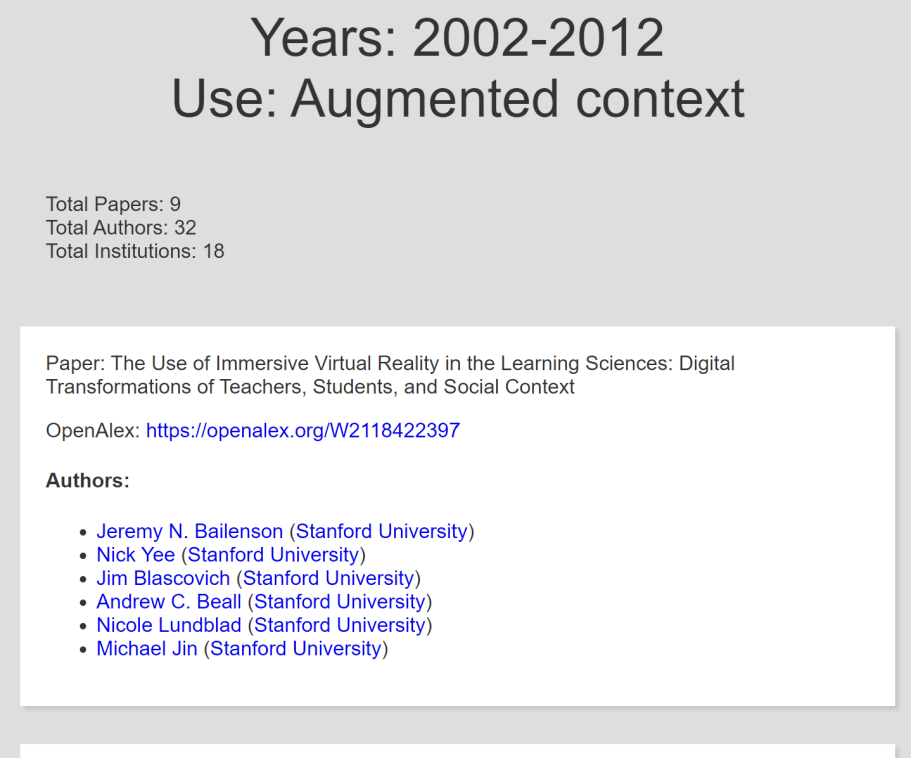


Figura 35: Página de detalhes de um Tema de Uso gerado

4.4. TESTES

Para garantir a qualidade da solução desenvolvida, ambos os sistemas da solução global foram exaustivamente testados, não existindo, à data de redação deste relatório, nenhum erro identificado.

4.4.1. Serviço WebAPI

O nosso serviço WebAPI foi testado com recurso à ferramenta *Postman* que funciona como um cliente para o serviço. De seguida apresentam-se os cenários usados nos testes, bem como os resultados obtidos.

Teste	Cenário	Resultado Esperado	Resultado Obtido
1	O serviço está disponível	O serviço responde	OK
2	É obtido um Bearer token mediante credenciais válidas	Bearer token	OK
3	Não é obtido um Bearer token meadiante credenciais inválidas	401 Unauthorized	OK
4	Todos os recursos, com excepção de api/token, devem recusar pedidos sem um Bearer token válido no cabeçalho	401 Unauthorized	OK
5	O recurso api/accounts devolve a lista com todos os valores da tabela correspondente	200 OK	OK
6	O recurso api/papers devolve a lista com todos os valores da tabela correspondente	200 OK	OK
7	O recurso api/authors devolve a lista com todos os valores da tabela correspondente	200 OK	OK
8	O recurso api/uses devolve a lista com todos os valores da tabela correspondente	200 OK	OK
9	O recurso api/institutions devolve a lista com todos os valores da tabela correspondente	200 OK	OK
10	O recurso api/accounts/{authorid} devolve devolve os detalhes do autor com identificador {authorid}	200 OK	OK
11	O recurso api/api/uses?year={Year}&includealluptoyear={val} devolve todos os Temas de Uso para os critérios Years e val	200 OK	OK

4.4.2. *UI Front-End* (Unity)

O sistema *Unity* foi testado manualmente e apresentam-se de seguida os cenários testados, assim com os correspondentes resultados obtidos.

Cenário 1: A aplicação Unity está disponível e carrega corretamente os filtros dos anos, autores e instituições

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/

Resultado: A aplicação foi inicializada corretamente.

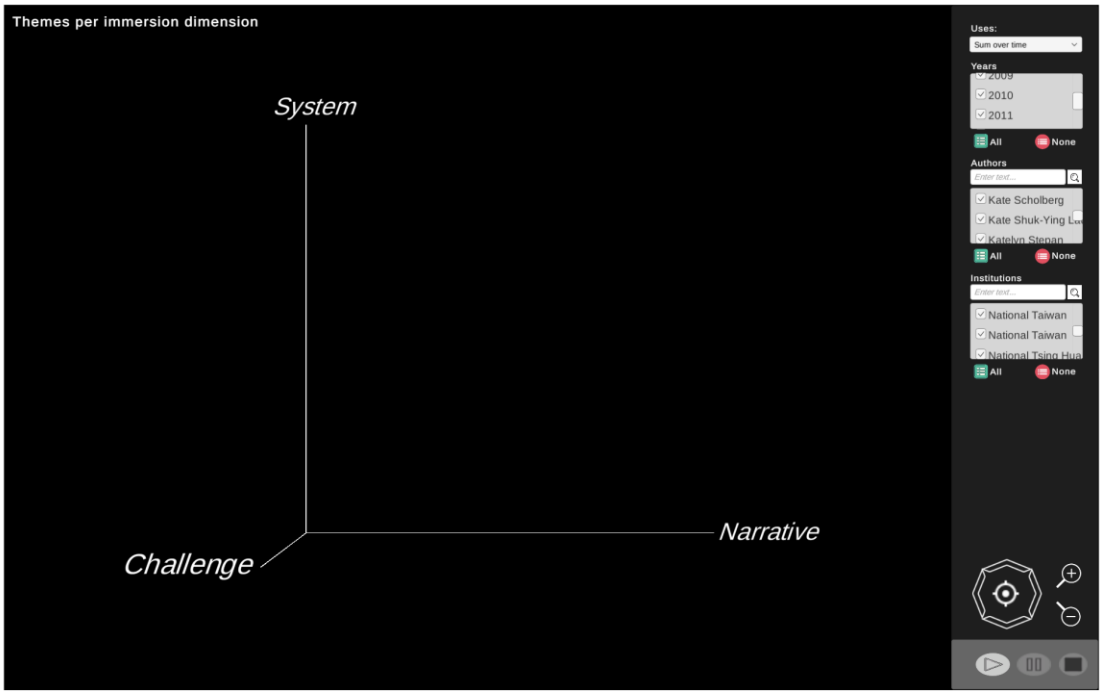


Figura 36: Resultado do teste cenário 1

Cenário 2: Os filtros são responsivos

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. Clicar em alguns valores dos filtros Years, Authors e Institutions
3. Selecionar as opções All e None em cada um dos filtros

Resultado: Os valores dos filtros foram selecionados de acordo com o esperado.

Cenário 3: Os modelos são gerados corretamente para o Caso de Uso "Qual é a evolução dos usos ao longo do tempo?"

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. No filtro Uses selecionar: "Sum over time"
3. Adicionar os anos 2011, 2012 2 2013
4. Selecionar todos os autores e instituições
5. Pressionar o botão Start
6. Pressionar com o rato em cima do Temas de Uso "Augmented Context"

Resultado: Foi iniciada uma animação com os anos selecionados. Os Temas de Uso não vão desaparecendo ao longo do tempo, mas antes, vão sendo acrescentados novos Temas de Uso com o passar dos anos e alguns do que já existiam vão aumentando o diâmetro da esfera, à medida que aumentam as publicações nesses Temas de Uso. Ao pressionar o rato em cima do Tema de Uso, foi apresentada a página de detalhes.

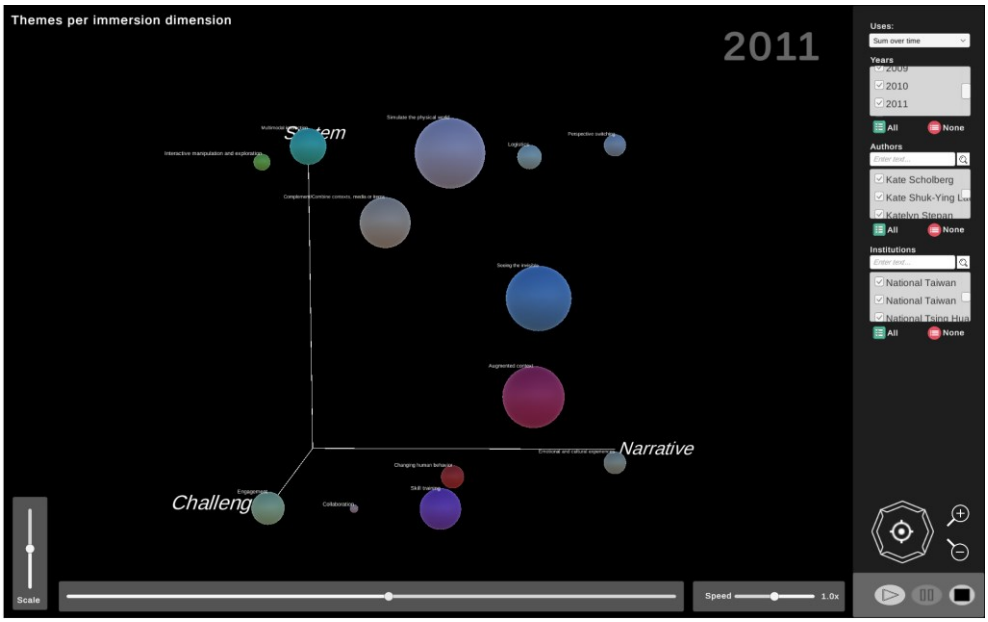


Figura 37: Resultado 1 teste cenário 3 no ano 2011

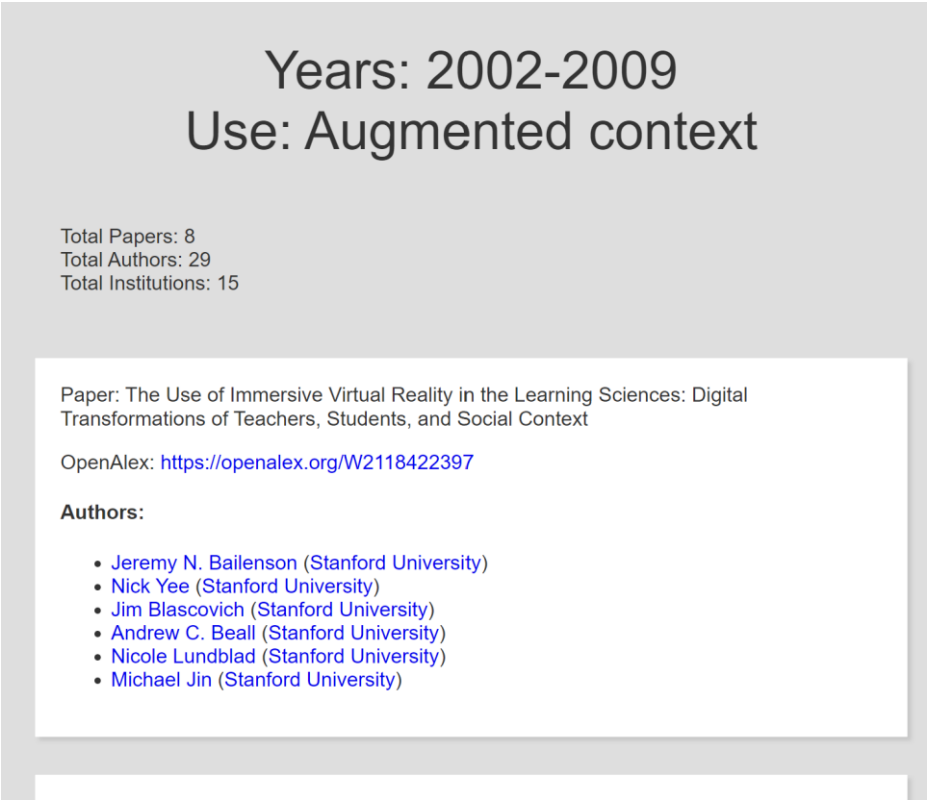


Figura 38: Página de detalhes do cenário 3 de um Tema de Uso no ano 2011

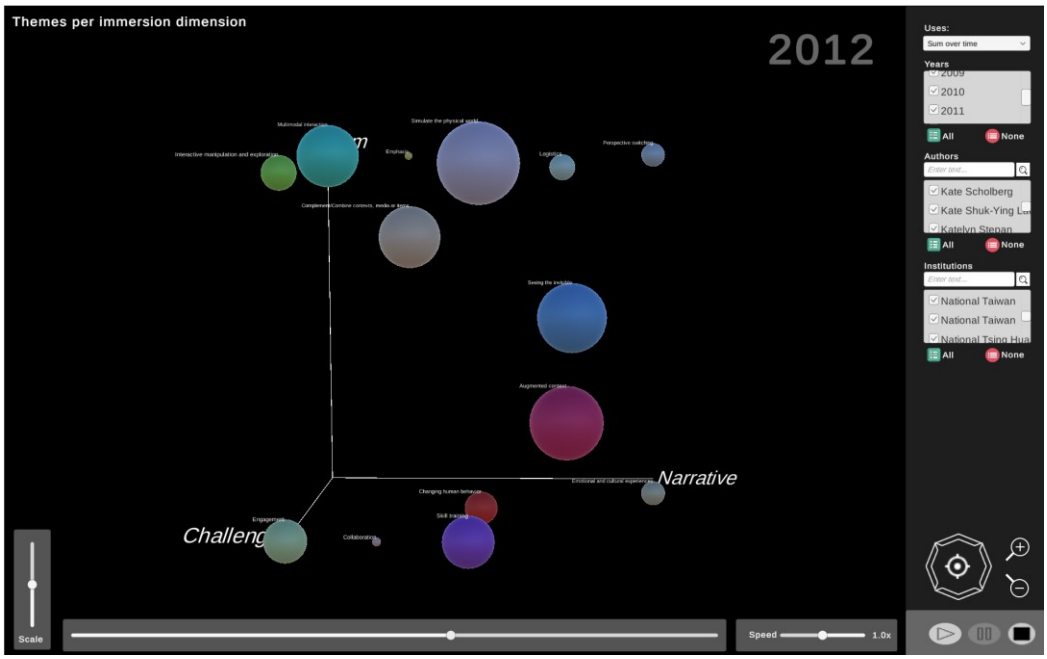


Figura 39: Resultado 2 teste cenário 3 no ano 2012

Years: 2002-2012
Use: Augmented context

Total Papers: 9
Total Authors: 32
Total Institutions: 18

Paper: The Use of Immersive Virtual Reality in the Learning Sciences: Digital Transformations of Teachers, Students, and Social Context

OpenAlex: https://openalex.org/W2118422397

Authors:

- Jeremy N. Bailenson (Stanford University)
- Nick Yee (Stanford University)
- Jim Blascovich (Stanford University)
- Andrew C. Beall (Stanford University)
- Nicole Lundblad (Stanford University)
- Michael Jin (Stanford University)

Figura 40: Página de detalhes do cenário 3 de um Tema de Uso no ano 2012

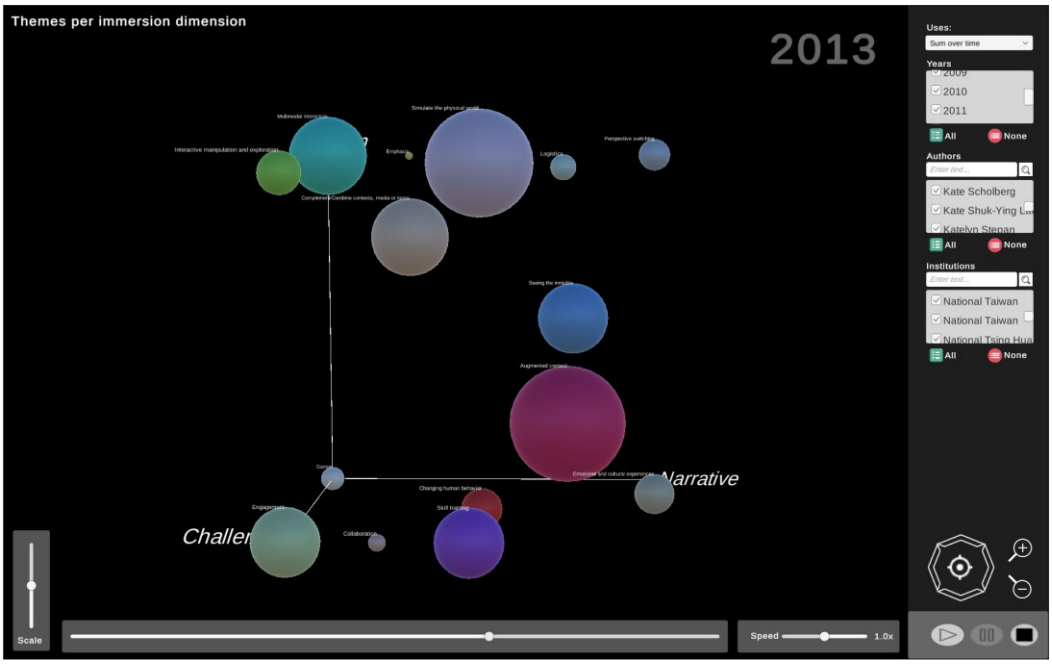


Figura 41: Resultado 3 teste cenário 3 no ano 2013

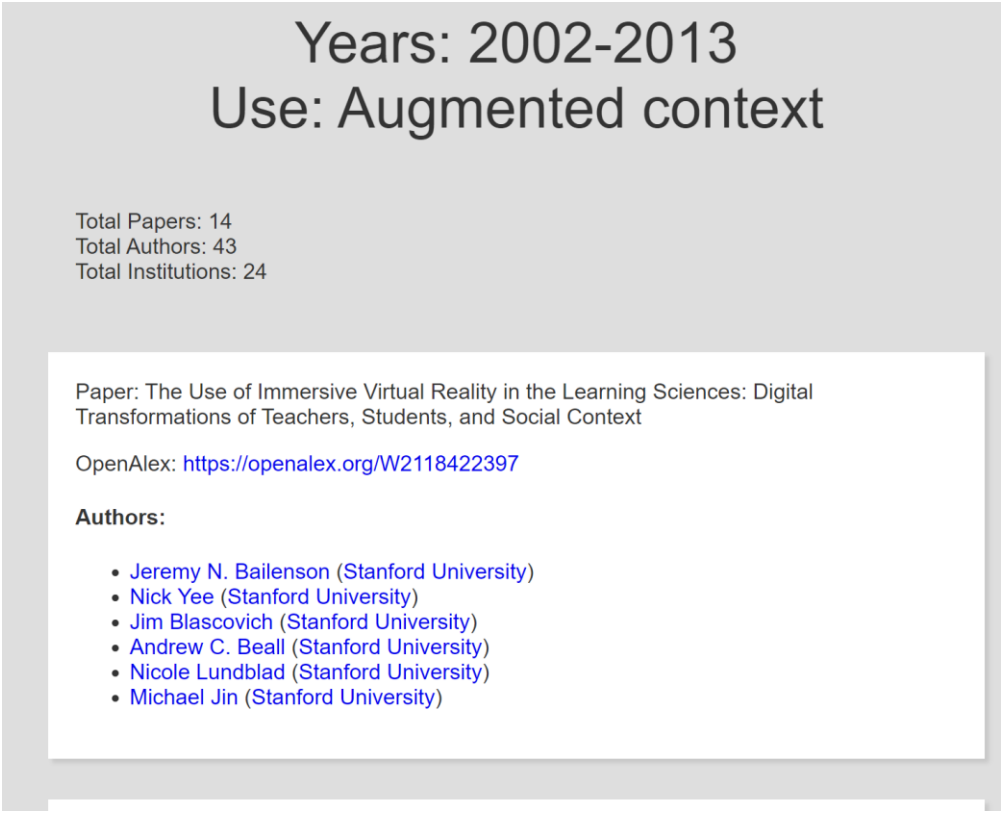


Figura 42: Página de detalhes do cenário 3 de um Tema de Uso no ano 2013

Cenário 4: Os modelos são gerados corretamente para o Caso de Uso "Quais foram os usos publicados num ano?"

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. No filtro Uses selecionar: "Specific to year"
3. Adicionar os anos 2011, 2012 2 2013
4. Selecionar todos os autores e instituições
5. Pressionar o botão Start
6. Pressionar com o rato em cima do Temas de Uso "Augmented Context"

Resultado: Foi iniciada uma animação com os anos selecionados. Os Temas de Uso vão surgindo e desaparecendo ao longo do tempo. Alguns Temas de Uso que já existiam vão aumentando o diâmetro da esfera, enquanto outros vão diminuindo. Ao pressionar o rato em cima do Tema de Uso, foi apresentada a página de detalhes. O Tema de Uso não surge no ano 2011.

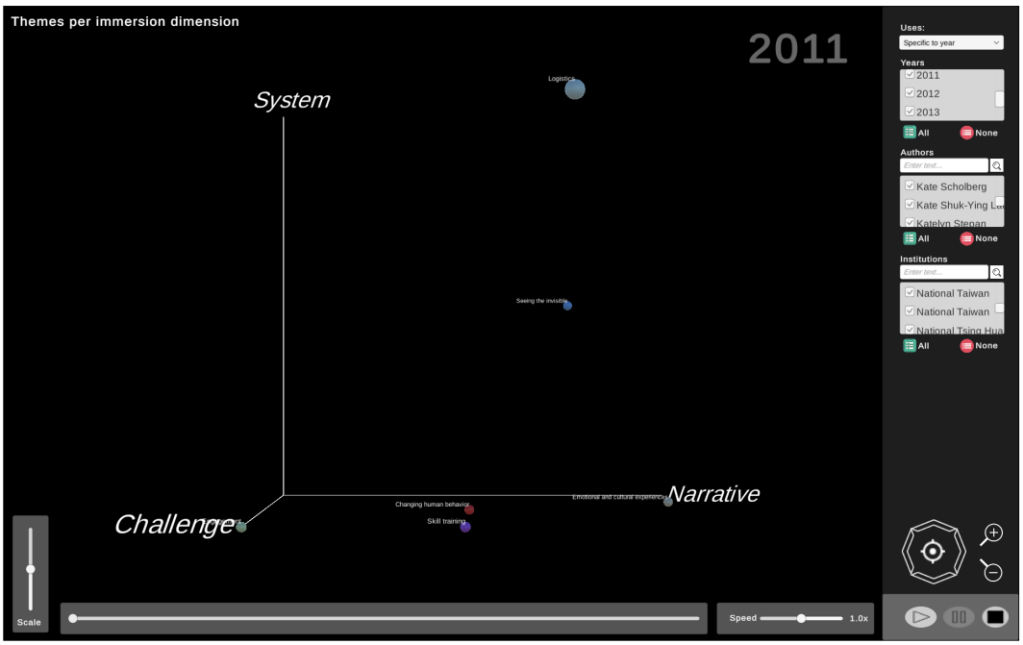


Figura 43: Resultado 1 teste cenário 4 no ano 2011

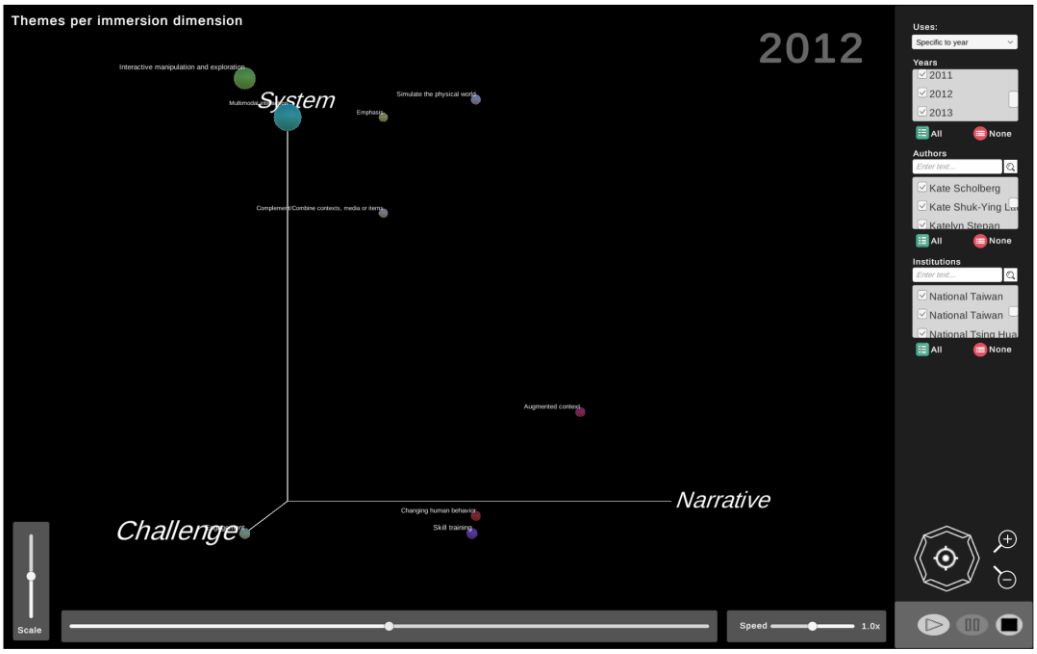


Figura 44: Resultado 2 teste cenário 4 no ano 2012

Year: 2012
Use: Augmented context

Total Papers: 1
Total Authors: 3
Total Institutions: 3

Paper: Learning while exercising for science education in augmented reality among adolescents

OpenAlex: https://openalex.org/W1965146748

Authors:

- Kuei-Fang Hsiao (Ming Chuan University)
- Nian-Shing Chen (National Sun Yat-sen University) (Sun Yat-sen University)
- Shih-Yu Huang (Ming Chuan University)

Figura 45: Página de detalhes do cenário 4 de um Tema de Uso no ano 2012

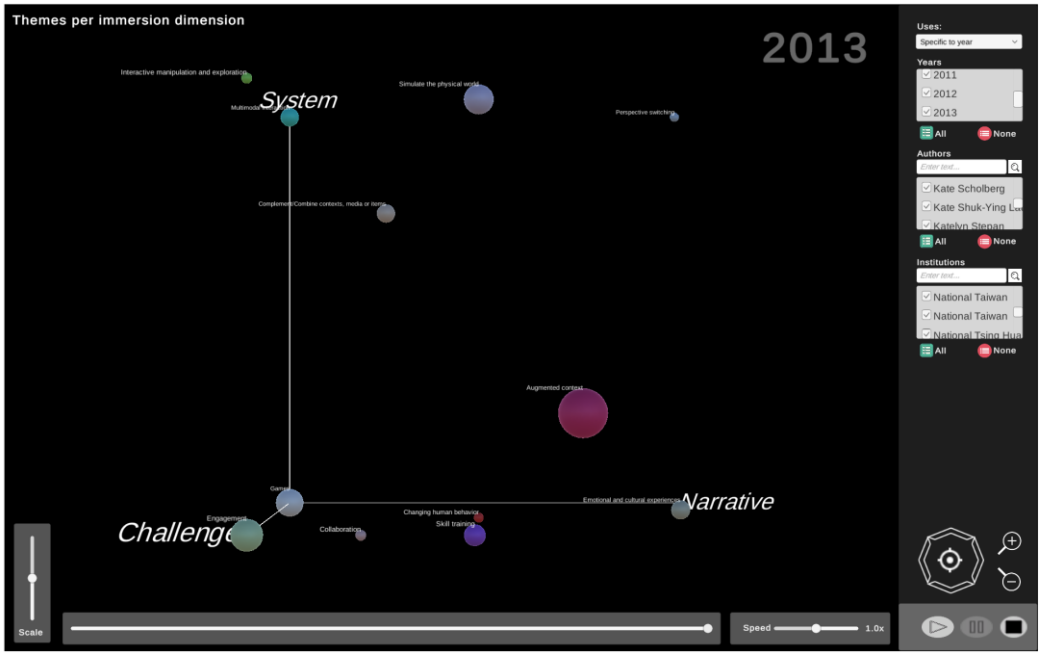


Figura 46: Resultado 3 teste cenário 4 no ano 2013

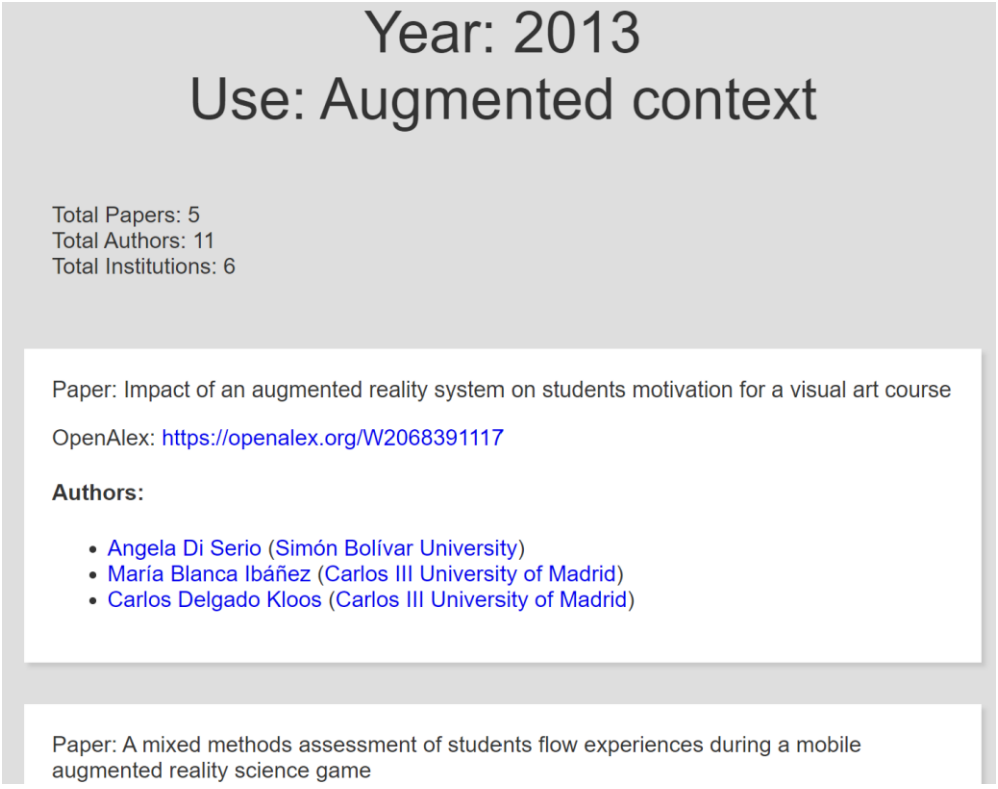


Figura 47:Página de detalhes do cenário 4 de um Tema de Uso no ano 2013

Cenário 5: Desempenho na geração dos modelos

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. No filtro Uses selecionar: "Sum over time"
3. Adicionar todos os anos, autores e instituições
4. Pressionar o botão Start

Resultado: O processo de geração de todos os modelos e início da apresentação leva menos de 1 segundo a terminar.

Cenário 6: Interação com os modelos

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. No filtro Uses selecionar: "Sum over time"
3. Adicionar todos os anos, autores e instituições
4. Pressionar o botão Start
5. Utilizar o botão esquerdo do rato para deslocar o referencial no espaço
6. Utilizar o botão direito do rato e controlos no ecrã para girar o referencial no espaço
7. Utilizar o *scroll* do rato e controlos no ecrã para aproximar/afastar referencial
8. Utilizar controlo no ecrã para centrar referencial na sua posição inicial

Resultado: Todas as operações realizadas tiveram sucesso.

Cenário 7: Interação com a apresentação

Instruções:

1. Aceder ao URL https://temporal-analysis.azurewebsites.net/
2. No filtro Uses selecionar: "Sum over time"
3. Adicionar todos os anos, autores e instituições
4. Pressionar o botão Start
5. Alterar escala do referencial
6. Alterar velocidade da animação
7. Navegar no controlador temporal

Resultado: Todas as operações realizadas tiveram sucesso.

4.5. EXEMPLOS

Uma vez que a secção de testes descreve já um bom conjunto de exemplos de utilização do sistema *Unity*, vão-se aqui deixar alguns exemplos acerca da utilização do serviço WebAPI.

Obter um token de autenticação do serviço WebAPI

Para a obtenção deste *token* deve ser submetido um *post request*, especificando o respetivo clientId e clientSecret, de acordo com a figura seguinte

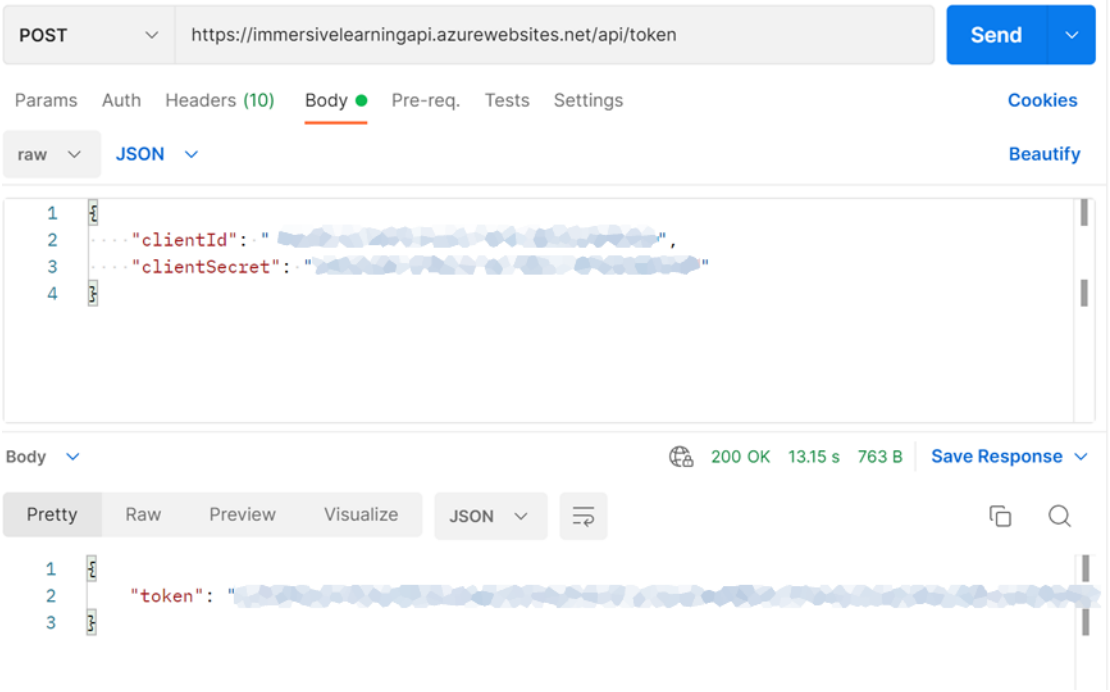


Figura 48: Pedido/Resposta do token de autenticação

Obter do serviço WebAPI a lista de todos os autores

Na figura seguinte, pode-se ver um exemplo de um pedido ao serviço, para obter a lista completa com todos os autores dos artigos analisados.

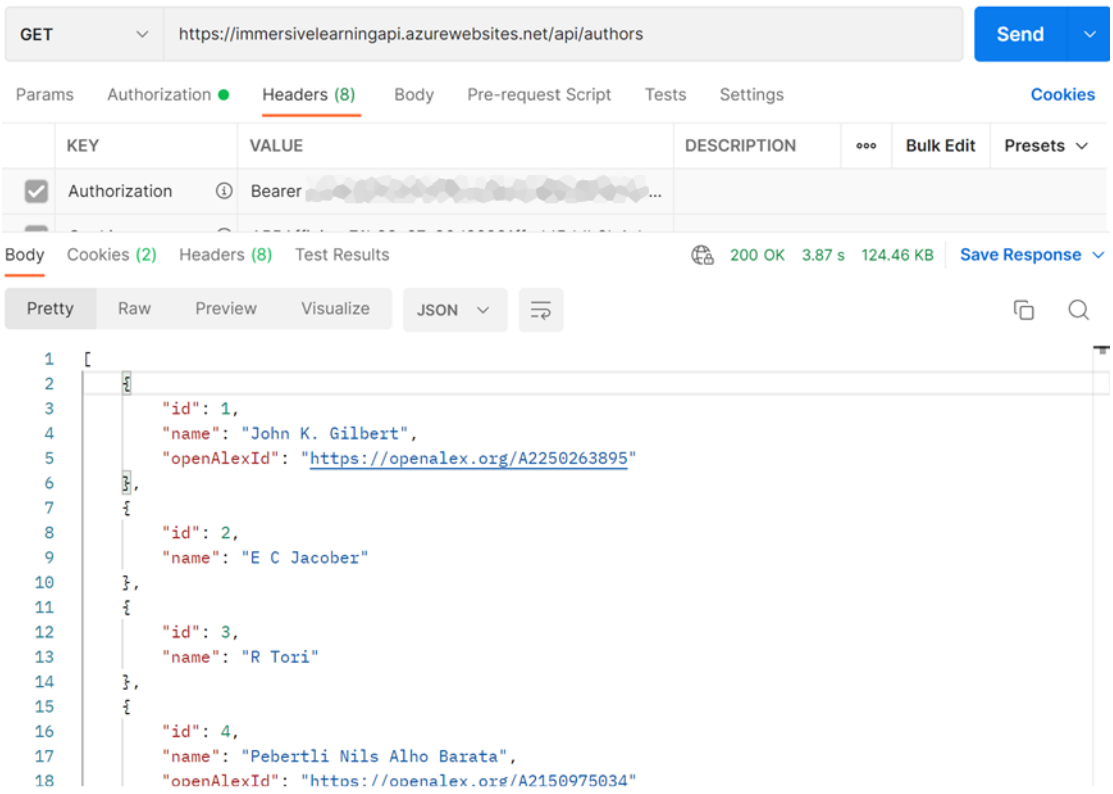


Figura 49: Pedido ao serviço WebAPI para a obtenção de todos os autores

Nesta figura, pode-se observar que foi enviado um *request* ao *recurso api/authors*, incluindo o *token* de autenticação como parâmetro no *header*. O serviço response com um *ActionResult*, que neste caso se trata de um 200 *OK*, uma vez que o processamento do pedido foi corretamente concluído pelo serviço. Caso o *token* de autenticação não tivesse sido especificado, estivesse incorreto ou tivesse excedido a sua validade, o serviço responde com um *ActionResult 401 Unauthorized*, como podemos ver na figura seguinte

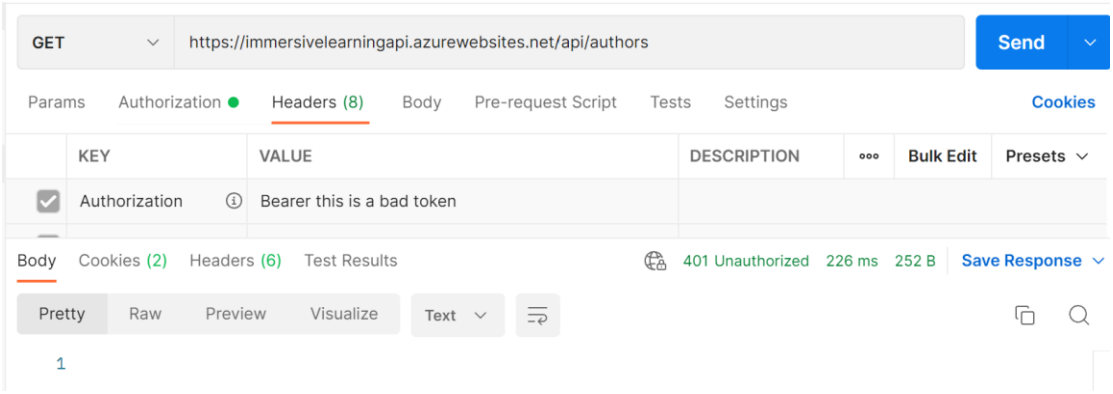


Figura 50: Pedido ao serviço utilizando um token de autenticação inválido

Capítulo 5

Conclusões e Trabalho Futuro

Neste capítulo descrevem-se os resultados principais obtidos, faz-se uma reflexão face aos objetivos pretendidos, às dificuldades e limitações encontradas no trabalho, além das possíveis melhorias e trabalho futuro que poderia ser feito nos resultados alcançados.

5.1. RESULTADOS PRINCIPAIS

O projeto decorreu sempre a um ritmo contínuo e crescente, dentro da calendarização inicialmente prevista. Para isso muito contribuiu o agendamento semanal das sessões de acompanhamento, que foi sempre um espaço aberto para a discussão de ideias, mas foi também um espaço de orientação e reflexão acerca do trabalho já desenvolvido e das próximas metas a alcançar. Os objetivos inicialmente propostos foram globalmente atingidos, permitindo a minha familiarização com o desenvolvimento para ambientes imersivos, e atrevo-me a afirmar que fomos ainda mais além, uma vez que fomos não só capazes de desenvolver um mero protótipo de suporte aos Casos de Uso identificados, mas fomos capazes de desenvolver uma ferramenta de apresentação de dados visuais que permite à comunidade não ser um mero agente passivo na visualização desses dados, mas antes interagir com esses dados. Essa é, aliás, uma das marcas de distinção dos produtos de realidade imersiva, a sua capacidade de envolver e cativar o utilizador. Através da execução dos Casos de Uso na ferramenta, conseguiu-se rapidamente

compreender qual tem sido a evolução dos estudos no contexto da aprendizagem imersiva e os investigadores na área vão conseguir identificar as tendências e as lacunas de investigação no campo.

A investigação publicada no artigo "*Finding the Gaps About Uses of Immersive Learning Environments: A Survey of Surveys*" [Beck, Morgado and Shea 2020], fornece a base de trabalho para este projeto e, com a ferramenta desenvolvida, acredito que se conseguiu, de forma eficaz, contribuir para responder ao desafio lançado. A qualidade do trabalho foi, aliás, já atestada pelos professores Dennis Beck e Daphne Economou, que tiveram a gentileza de se disponibilizar a assistir a uma apresentação da versão final da ferramenta e onde demonstraram o interesse em publicá-la no portal da iLRN.

5.2. LIMITAÇÕES

Durante o desenvolvimento do projeto, foram várias as dificuldades encontradas. Desde logo, os meus próprios conhecimentos acerca do IDE de desenvolvimento *Unity*, com o qual eu nunca tinha trabalhado, bem como a minha inexperiência em ambientes imersivos. Além disso, foi ainda um desafio conseguir ter a solução final a correr em WebGL, tendo sido vários os problemas com que me deparei no momento de tentar executar a ferramenta no WebGL. Como exemplo dessas dificuldades tem-se que o *Unity* sobre WebGL não suporta todos os namespaces do framework .Net, não suporta a referenciação dinâmica de componentes externos, como é o caso do componente *sqlite3.dll* necessário à comunicação com bases de dados SQLite; e também não suporta múltiplas *threads* a correr no browser. Todos estes desafios foram sendo ultrapassados.

Uma limitação que não se foi capaz de ultrapassar, no tempo proposto para o desenvolvimento do projeto, foi que as soluções *Unity* em WebGL não permitem que a página HTML gerada automaticamente para apresentação no browser seja uma página responsiva. Esta limitação pode ser ultrapassada, criando uma página responsiva, que interaja com o *Unity* através do browser. Esta solução, difícil de implementar, foi identificada como uma solução teórica, mas não existiu o tempo necessário para a explorar, pelo que fica aqui como limitação e sugestão futura de melhoria.

5.3. DESENVOLVIMENTO FUTURO E MELHORIAS

A ferramenta desenvolvida está em condições de passar de protótipo à etapa seguinde, um produto de produção. Mas isso não significa que não haja muitas outras melhorias e alterações que possam ainda ser implementadas.

A decisão que se tomou de criar um serviço WebAPI que permite processar e obter os dados da base de dados, abriu portas a muitas implementações futuras que necessitem consumir desses dados, uma vez que permite aos clientes do serviço uma abstração relativamente aos dados. Neste serviço implementaram-se apenas os recursos necessários ao desenvolvimento da solução particular, no entanto, existem muitas outras Entidades na base de dados, para as quais não foram implementados os recursos necessários. Esta é uma potencial ação de melhoria ao serviço, uma vez que a sua concretização permite alargar o que a audiência pode consumir do serviço, para os seus próprios propósitos. Outra sugestão de melhoria no serviço, ou que deverá pelo menos

ser alvo de reflexão em alterações futuras, é o método de autenticação para a utilização dos recursos. Neste momento, existe apenas um cliente do serviço, que é a ferramenta *Unity*, e as credenciais de autenticação foram definidas num ficheiro de configuração do serviço. Em futuras alterações ao serviço, deve ser ponderado se isto ainda faz sentido, ou se deve ser alterado para armazenar as credenciais de autenticação em algum outro lugar, como uma base de dados, ou mesmo se o próprio método de autenticação continua a ser o mais adequado. Deve ainda ser observado que, no serviço, se implementa um método de autenticação, mas não de autorização. Fica também como sugestão para o futuro, avaliar a necessidade de implementar a autorização do acesso dos clientes aos recursos. Numa última nota de sugestão de melhorias ao serviço, a base de dados que está a ser utilizada é uma base de dados SQLite. Apesar de ser, neste momento, uma base de dados adequada às necessidades, uma vez que o volume de dados armazenados não é muito significativo, conseguiu-se ainda assim observar alguns problemas de desempenho no acesso aos dados, tendo existido a necessidade de em alguns casos tentar diferentes estratégias para a obtenção dos dados. Fica como sugestão para o futuro, ponderar a migração dos dados para um servidor de dados mais robusto como o SQL Server ou Oracle.

Sobre a ferramenta desenvolvida em *Unity*, um primeiro aspeto a referir, enquanto sugestão de melhoria futura, tem a ver como o padrão de desenvolvimento seguido. Não houve o tempo necessário para avaliar e implementar um padrão de desenvolvimento uniforme no projeto C# do *Unity* e, apesar de o código implementado ser claro e ter existido uma preocupação com a separação de responsabilidades, pode ser avaliado no futuro um *refactoring* do projeto para lhe conferir propriedades desejáveis como seja a sua própria extensibilidade. Outra sugestão para melhorias, ainda que relacionada com a anterior, tem exatamente a ver com a extensibilidade da ferramenta. Acredito que a ferramenta desenvolvida tem potencialidade para ser tornar ela própria numa framework que possa ser importada noutros projetos e tirar partido dos elementos visuais desenvolvidos. Esta foi, aliás, umas das possibilidades discutidas durante a sessão de apresentação com o Prof. Dennis Beck e a Prof. Daphne Economou. Seria interessante que, no futuro, pudessem ser acrescentados outros parâmetros e critérios que permitam obter outro tipo de resultados, mantendo a dinâmica dos elementos visuais desenvolvidos. Foi referido nas limitações que seria interessante o desenvolvimento de uma página HTML responsiva que interaja com o *Unity*. Isto iria permitir a definição de filtros mais apelativos e visualmente agradáveis, além de permitir a extensão da aplicação a diferentes critérios de filtragem desenvolvidos externamente.

Bibliografia

[Automapper 2022] *Automapper*. AutoMapper. (n.d.). Consultado a 13 de junho de 2022 em https://automapper.org/

[Beck, Morgado and Shea 2020] Beck, D., Morgado, L., & Shea, P. (2020). Finding the gaps about uses of immersive learning environments: A survey of surveys. *JUCS - Journal of Universal Computer Science*, *26*(8), 1043–1073. https://doi.org/10.3897/jucs.2020.055

[Beck, Morgado, Lee, Gutl, Dengel, Wang, Warren and Richter 2021] Beck, D., Morgado, L., Lee, M., Gutl, C., Dengel, A., Wang, M., Warren, S., & Richter, J. (2021). Towards an immersive learning knowledge tree - A conceptual framework for mapping knowledge and tools in the field. *2021 7th International Conference of the Immersive Learning Research Network (ILRN)*. https://doi.org/10.23919/ilrn52045.2021.9459338

[iLRN 2022] The iLRNetwork Hub. (n.d.). Consultado a 13 de junho de 2022, em https://immersivelrn.org/

[DI 2022] Rick-Anderson. (n.d.). *Dependency injection in ASP.NET Core*. Microsoft Docs. Consultado a 13 de junho de 2022, em https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0

[Swagger 2022] RicoSuter. (n.d.). *ASP.NET core web API documentation with swagger / openapi*. ASP.NET Core web API documentation with Swagger / OpenAPI. Consultado a 13 de junho de 2022, em https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-3.1

[Unity 2022] Technologies, U. (n.d.). Unity. Consultado a 13 de junho de 2022, em https://unity.com/

[WebGL 2022] *WebGL*. The Khronos Group. (2011, July 19). Consultado a 13 de junho de 2022, em https://www.khronos.org/webgl/

Anexo I – Temporal Analysis (Unity)

Neste anexo, pode-se encontrar o código correspondente ao projeto C# implementado para o sistema desenvolvido em *Unity*. À data de envio deste relatório, este sistema está disponível em https://temporal-analysis.azurewebsites.net/

init.cs

```
using Assets.Scripts;
using Assets.Scripts.APIManagement;
using Assets.Scripts.CustomComponents;
using Assets.Scripts.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;

public class init : MonoBehaviour
{
    public GameObject graph;
    public GameObject esferaPrefab;
    public GameObject yearsToggle;
    public GameObject authorsToggle;
    public GameObject institutionsToggle;
    public Button btnAllYears;

    private APIHelper apiCalls;

    // Start is called before the first frame update
    void Start()
    {
        Entities.LoadingYears = true;
        Entities.LoadingAuthors = true;
        Entities.LoadingInstitutions = true;

        Input.multiTouchEnabled = false;

        apiCalls = new APIHelper();

        apiCalls.RequestComplete += this.APIRequestCompleteHandler;
        Entities.Domain.EntityLoaded += this.EntityLoadedHandler;

        //Initial set up
        StartCoroutine(apiCalls.SendAPIRequest("api/Uses/Years", "GET"));
    }
```

```
/// <summary>
/// Handles event RequestComplete
/// </summary>
/// <param name="uri">uri originally called</param>
/// <param name="jsonContent">service response</param>
private void APIRequestCompleteHandler(string uri, string jsonContent)
{
    switch(uri)
    {
        case "api/Uses/Years":
            StartCoroutine(apiCalls.SendAPIRequest("api/authors", "GET"));
            AddYears(jsonContent);
            break;

        case "api/papers":
            StartCoroutine(apiCalls.SendAPIRequest("api/papers/papersaccounts", "GET"));
            Entities.Domain.GeneratePapers(jsonContent);
            break;

        case "api/papers/papersaccounts":
            StartCoroutine(apiCalls.SendAPIRequest("api/authors/authorspapers", "GET"));
            Entities.Domain.GeneratePapersAccounts(jsonContent);
            break;

        case "api/authors":
            StartCoroutine(apiCalls.SendAPIRequest("api/institutions", "GET"));
            Entities.Domain.GenerateAuthors(jsonContent);
            break;

        case "api/authors/authorspapers":
            StartCoroutine(apiCalls.SendAPIRequest("api/authors/authorsinstitutions", "GET"));
            Entities.Domain.GenerateAuthorsPapers(jsonContent);
            break;

        case "api/authors/authorsinstitutions":
            StartCoroutine(apiCalls.SendAPIRequest("api/Uses", "GET"));
            Entities.Domain.GenerateAuthorsInstitutions(jsonContent);
            break;

        case "api/institutions":
            StartCoroutine(apiCalls.SendAPIRequest("api/papers", "GET"));
            Entities.Domain.GenerateInstitutions(jsonContent);
            break;

        case "api/Uses":
            StartCoroutine(apiCalls.SendAPIRequest("api/Uses/usesaccounts", "GET"));
            Entities.Domain.GenerateUses(jsonContent);
            break;

        case "api/Uses/usesaccounts":
            Entities.Domain.GenerateUsesAccounts(jsonContent);
            break;

    }
```

```
}

/// <summary>
/// Handles event EntityLoaded
/// </summary>
/// <param name="Entity">The loaded Entity</param>
private void EntityLoadedHandler(EntitiesType Entity)
{
    switch(Entity)
    {
        case EntitiesType.Papers:
            //AddYears();
            break;

        case EntitiesType.Authors:
            AddAuthors();
            break;

        case EntitiesType.Institutions:
            AddInstitutions();
            break;
    }
}

/// <summary>
/// Loads all Years to the corresponding filter
/// </summary>
private void AddYears(string jsonContent)
{
    try
    {
        //There is no Enity Years so the values are obtained directly from the source
        List<int> years = JsonConvert.DeserializeObject<List<int>>(jsonContent);
        years.Sort();

        //Find container to values
        GameObject yearsContent = GameObject.FindGameObjectWithTag("YearsContent");

        if(yearsContent != null)
        {
            //Iterate through the Years and add a checkbox to each value
            foreach (int year in years.Where(y => y > 0))
            {
                //Create a new checkbox
                GameObject newToggle = (GameObject)Instantiate(yearsToggle);
                newToggle.GetComponentInChildren<Text>().text = year.ToString();
                newToggle.GetComponent<Toggle>().isOn = true;

                //Set the parent of the newly checkbox to the container
                newToggle.transform.SetParent(yearsContent.transform, false);
            }

            //Loading is conplete and we can stop the spinning
            Entities.LoadingYears = false;
```

```
            }
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    /// <summary>
    /// Loads all authors to the corresponding filter
    /// </summary>
    private void AddAuthors()
    {
        try
        {
            //Find container to values
            GameObject authorsContent = GameObject.FindGameObjectWithTag("AuthorsContent");

            if(authorsContent != null)
            {
                //Iterate through the entity and add a checkbox to each value
                foreach (Author author in Entities.Authors.OrderBy(a => a.Name))
                {
                    //Create a new checkbox
                    GameObject newToggle = (GameObject)Instantiate(authorsToggle);
                    newToggle.GetComponentInChildren<Text>().text = author.Name;

                    //Set the parent of the newly checkbox to the container
                    newToggle.transform.SetParent(authorsContent.transform, false);

                    //Saving the value's id to be used later when applying the filter
                    AuthorComponent authorComponent =
newToggle.GetComponent<AuthorComponent>();
                    authorComponent.AuthorId = author.Id;
                    authorComponent.AuthorName = author.Name;
                }

                //Loading is conplete and we can stop the spinning
                Entities.LoadingAuthors = false;
            }
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    /// <summary>
    /// Loads all institutions to the corresponding filter
    /// </summary>
    private void AddInstitutions()
    {
        try
        {
```

```
        //Find container to values
        GameObject institutionsContent =
GameObject.FindGameObjectWithTag("InstitutionsContent");

        if(institutionsContent != null)
        {
            //Iterate through the entity and add a checkbox to each value
            foreach (Institution institution in Entities.Institutions.OrderBy(a => a.Name))
            {
                //Create a new checkbox
                GameObject newToggle = (GameObject)Instantiate(institutionsToggle);
                newToggle.GetComponentInChildren<Text>().text = institution.Name;

                //Set the parent of the newly checkbox to the container
                newToggle.transform.SetParent(institutionsContent.transform, false);

                //Saving the value's id to be used later when applying the filter
                InstitutionComponent institutionComponent =
newToggle.GetComponent<InstitutionComponent>();
                institutionComponent.InstitutionId = institution.Id;
                institutionComponent.InstitutionName = institution.Name;

            }

            //Loading is conplete and we can stop the spinning
            Entities.LoadingInstitutions = false;
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
  }

  // Update is called once per frame
  void Update()
  {

  }

}
```

APIManagement/TokenRequest.cs

```
using System;

namespace Assets.Scripts.APIManagement
{
  [Serializable]
  public class TokenRequest
  {
    public string clientId;
```

```
        public string clientSecret;
    }
}
```

APIManagement/TokenResponse.cs

```
using System;

namespace Assets.Scripts.APIManagement
{
    [Serializable]
    public class TokenResponse
    {
        public string token;
    }
}
```

APIManagement/APIHelper.cs

```
using System.Collections;
using System.Text;
using UnityEngine;
using UnityEngine.Networking;

namespace Assets.Scripts.APIManagement
{
    public class APIHelper
    {
        private static string urlBase = "https://immersivelearningapi.azurewebsites.net/";
        private static string token = "";

        public delegate void APIRequestsEvents(string uri, string jsonContent);
        public event APIRequestsEvents RequestComplete;

        public IEnumerator SendAPIRequest(string uri, string type = "GET")
        {
            var tokenRequest = new TokenRequest
            {
                clientId = "NÃO INCLUÍDO NA DISPONIBILIZAÇÃO PÚBLICA",
                clientSecret = " NÃO INCLUÍDO NA DISPONIBILIZAÇÃO PÚBLICA "
            };

            var request = CreateRequest("api/token", "POST", tokenRequest);

            yield return request.SendWebRequest();

            while (!request.isDone)
            {
                yield return new WaitForEndOfFrame();
            }
```

```
        if (request.result == UnityWebRequest.Result.ConnectionError)
        {
            System.Diagnostics.Debug.WriteLine($"Error: {uri}");
            Debug.Log(request.error);
        }
        else
        {
            var tokenResponse =
JsonUtility.FromJson<TokenResponse>(request.downloadHandler.text);
            token = tokenResponse.token;
        }

        yield return SendRequest(uri, type);

    }


    private IEnumerator SendRequest(string uri, string type = "GET")
    {
        var request = CreateRequest(uri, "GET");

        request.SetRequestHeader("Authorization", "Bearer " + token);
        yield return request.SendWebRequest();

        while (!request.isDone)
        {
            yield return new WaitForEndOfFrame();
        }

        if (request.result == UnityWebRequest.Result.ConnectionError)
        {
            System.Diagnostics.Debug.WriteLine($"Error: {uri}");
            Debug.Log(request.error);
        }
        else
        {
            var jsonContent = request.downloadHandler.text;
            RequestComplete(uri, jsonContent);
        }

        yield break;
    }

    private UnityWebRequest CreateRequest(string uri, string type = "GET", object data = null)
    {
        var request = new UnityWebRequest(urlBase + uri, type);
        if (data != null)
        {
            var bodyRaw = Encoding.UTF8.GetBytes(JsonUtility.ToJson(data));
            request.uploadHandler = new UploadHandlerRaw(bodyRaw);
        }

        request.downloadHandler = new DownloadHandlerBuffer();
```

```
            request.SetRequestHeader("Content-Type", "application/json");
            request.timeout = 15;

            return request;
        }
    }
}
```

Models/Author.cs

```
namespace Assets.Scripts.Models
{
    [System.Serializable]
    public class Author
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

Models/AuthorDetails.cs

```
using System.Collections.Generic;

namespace Assets.Scripts.Models
{
    public class AuthorDetails : Author
    {
        public List<Institution> Institutions { get; set; }
    }
}
```

Models/AuthorInstitution.cs

```
namespace Assets.Scripts.Models
{
    public class AuthorInstitution
    {
        public int AuthorId { get; set; }
        public int InstitutionId { get; set; }
    }
}
```

Models/AuthorPaper.cs

```
namespace Assets.Scripts.Models
{
    public class AuthorPaper
    {
        public int AuthorId { get; set; }
        public int PaperId { get; set; }
    }
}
```

Models/Institution.cs

```
namespace Assets.Scripts.Models
{
    public class Institution
    {

        public int Id { get; set; }
        public string Name { get; set; }
        public string CountryCode { get; set; }
        public string OpenAlexId { get; set; }
    }

}
```

Models/Paper.cs

```
namespace Assets.Scripts.Models
{
    public class Paper
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int PubYear { get; set; }
        public string Date { get; set; }
        public string OpenAlexId { get; set; }
    }

}
```

Models/PaperAccount.cs

```
namespace Assets.Scripts.Models
{
    public class PaperAccount
    {
        public int PaperId { get; set; }
        public int AccountId { get; set; }
    }
}
```

Models/PaperDetails.cs

```
using System.Collections.Generic;

namespace Assets.Scripts.Models
{
    public class PaperDetails : Paper
    {
        public List<AuthorDetails> Authors { get; set; }
    }
}
```

Models/Use.cs

```
namespace Assets.Scripts.Models
{
    public class Use
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public float SystemImmersion { get; set; }
        public float NarrativeImmersion { get; set; }
        public float ChallengeImmersion { get; set; }
        public string RGBColor { get; set; }
    }
}
```

Models/UseAccount.cs

```
namespace Assets.Scripts.Models
{
    public class UseAccount
    {
        public int UseId { get; set; }
        public int AccountId { get; set; }
    }
}
```

Models/UseDetails.cs

```
using System.Collections.Generic;

namespace Assets.Scripts.Models
{
    public class UseDetails
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int StartYear { get; set; }
        public int FinalYear { get; set; }
```

```
        public int PapersCount { get; set; }
        public int AuthorsCount { get; set; }
        public int InstitutionsCount { get; set; }
        public List<PaperDetails> Papers { get; set; }
    }
}
```

Models/UseModel.cs

```
using System.Collections.Generic;

namespace Assets.Scripts.Models
{
    public class UseModel
    {
        public int UseId { get; set; }
        public string Name { get; set; }
        public int Year { get; set; }
        public float ChallengeImmersion { get; set; }
        public float NarrativeImmersion { get; set; }
        public float SystemImmersion { get; set; }
        public string RGBColor { get; set; }
        public int PapersCount { get; set; }
        public int AuthorsCount { get; set; }
        public int InstitutionsCount { get; set; }
        public List<int> Papers { get; set; }
    }
}
```

CustomComponents/AuthorComponent.cs

```
using UnityEngine;

namespace Assets.Scripts.CustomComponents
{
    public class AuthorComponent : MonoBehaviour
    {
        [SerializeField]
        internal int AuthorId;
        [SerializeField]
        internal string AuthorName;
    }
}
```

CustomComponents/InstitutionComponent.cs

```
using UnityEngine;

namespace Assets.Scripts.CustomComponents
```

```
{
    public class InstitutionComponent : MonoBehaviour
    {
        [SerializeField]
        internal int InstitutionId;
        [SerializeField]
        internal string InstitutionName;
    }
}
```

CustomComponents/UseComponent.cs

```
using System.Collections.Generic;
using UnityEngine;

namespace Assets.Scripts.CustomComponents
{
    public class UseComponent : MonoBehaviour
    {
        [SerializeField]
        internal int Year;
        [SerializeField]
        internal int UseId;
        [SerializeField]
        internal string Name;
        [SerializeField]
        internal bool IncludePapersOverTime;
        [SerializeField]
        internal int PapersCount;
        [SerializeField]
        internal int AuthorsCount;
        [SerializeField]
        internal int InstitutionsCount;
        [SerializeField]
        internal List<int> Papers;
        [SerializeField]
        internal float NarrativeImmersion;
        [SerializeField]
        internal float SystemImmersion;
        [SerializeField]
        internal float ChallengeImmersion;
    }
}
```

Domain.cs

```
using Assets.Scripts.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
```

```
using System.Linq;

namespace Assets.Scripts
{
    public class Domain
    {
        public delegate void EntitiesEvents(EntitiesType Entity);
        public event EntitiesEvents EntityLoaded;

        public void GeneratePapers(string jsonContent)
        {
            try
            {
                List<Paper> papers = JsonConvert.DeserializeObject<List<Paper>>(jsonContent);
                Entities.Papers = papers;

                EntityLoaded(EntitiesType.Papers);
            }
            catch (Exception ex)
            {

                throw ex;
            }
        }

        public void GeneratePapersAccounts(string jsonContent)
        {
            try
            {
                List<PaperAccount> values =
JsonConvert.DeserializeObject<List<PaperAccount>>(jsonContent);

                Entities.PapersAccounts = values;

                EntityLoaded(EntitiesType.PapersAccounts);
            }
            catch (Exception ex)
            {

                throw ex;
            }

        }

        public void GenerateAuthors(string jsonContent)
        {
            try
            {
                List<Author> authors = JsonConvert.DeserializeObject<List<Author>>(jsonContent);

                Entities.Authors = authors;

                EntityLoaded(EntitiesType.Authors);
            }
```

```
        catch (Exception ex)
        {

            throw ex;
        }

    }

    public void GenerateAuthorsPapers(string jsonContent)
    {
        try
        {
            List<AuthorPaper> values =
JsonConvert.DeserializeObject<List<AuthorPaper>>(jsonContent);

            Entities.AuthorsPapers = values;

            EntityLoaded(EntitiesType.AuthorsPapers);
        }
        catch (Exception ex)
        {
            throw ex;
        }

    }

    public void GenerateAuthorsInstitutions(string jsonContent)
    {
        try
        {
            List<AuthorInstitution> values =
JsonConvert.DeserializeObject<List<AuthorInstitution>>(jsonContent);

            Entities.AuthorsInstitutions = values;

            EntityLoaded(EntitiesType.AuthorsInstitutions);
        }
        catch (Exception ex)
        {
            throw ex;
        }

    }


    public void GenerateInstitutions(string jsonContent)
    {
        try
        {
            List<Institution> values =
JsonConvert.DeserializeObject<List<Institution>>(jsonContent);

            Entities.Institutions = values;
```

```
            EntityLoaded(EntitiesType.Institutions);
        }
        catch (Exception ex)
        {
            throw ex;
        }

    }

    public void GenerateUses(string jsonContent)
    {
        try
        {
            List<Use> values = JsonConvert.DeserializeObject<List<Use>>(jsonContent);

            Entities.Uses = values;

            EntityLoaded(EntitiesType.Uses);
        }
        catch (Exception ex)
        {
            throw ex;
        }

    }

    public void GenerateUsesAccounts(string jsonContent)
    {
        try
        {
            List<UseAccount> values =
JsonConvert.DeserializeObject<List<UseAccount>>(jsonContent);

            Entities.UsesAccounts = values;

            EntityLoaded(EntitiesType.UsesAccounts);
        }
        catch (Exception ex)
        {
            throw ex;
        }

    }

    /// <summary>
    /// Generates models of Themes of Use based on selected param
    /// </summary>
    public List<(int Year, List<UseModel> models)> GenerateModels(List<int> years, List<int>
authors, List<int> institutions, bool includeAllUpToYear)
    {
        try
        {
            var usesModels = new List<(int Year, List<UseModel> models)>();
            foreach (var year in years)
```

```
            {
                var uses = new List<UseModel>();

                var usesList = (from u in Entities.Uses
                                join ua in Entities.UsesAccounts on u.Id equals ua.UseId
                                join pa in Entities.PapersAccounts on ua.AccountId equals pa.AccountId
                                join p in Entities.Papers on pa.PaperId equals p.Id
                                join ap in Entities.AuthorsPapers on p.Id equals ap.PaperId
                                join ai in Entities.AuthorsInstitutions on ap.AuthorId equals ai.AuthorId
                                where ((includeAllUpToYear && p.PubYear <= year) || (!includeAllUpToYear
&& p.PubYear == year)) &&
                                      authors.Contains(ap.AuthorId) &&
                                      institutions.Contains(ai.InstitutionId)
                                select new
                                {
                                    UseId = u.Id,
                                    PaperId = p.Id,
                                    AuthorId = ap.AuthorId,
                                    InstitutionId = ai.InstitutionId
                                }).ToList().Distinct();

                foreach (var use in usesList.Select(u => u.UseId).Distinct())
                {
                    var useTemp = Entities.Uses.Where(u => u.Id.Equals(use)).FirstOrDefault();
                    var useModel = new UseModel()
                    {
                        UseId = use,
                        Name = useTemp.Name,
                        Year = year,
                        ChallengeImmersion = useTemp.ChallengeImmersion,
                        NarrativeImmersion = useTemp.NarrativeImmersion,
                        SystemImmersion = useTemp.SystemImmersion,
                        RGBColor = useTemp.RGBColor,
                        PapersCount = usesList.Where(u => u.UseId.Equals(use)).Select(p =>
p.PaperId).Distinct().Count(),
                        AuthorsCount = usesList.Where(u => u.UseId.Equals(use)).Select(a =>
a.AuthorId).Distinct().Count(),
                        InstitutionsCount = usesList.Where(u => u.UseId.Equals(use)).Select(i =>
i.InstitutionId).Distinct().Count(),
                        Papers = usesList.Where(u => u.UseId.Equals(use)).Select(p =>
p.PaperId).Distinct().ToList()
                    };


                    uses.Add(useModel);
                }
                usesModels.Add((year, uses));
            }

            return usesModels;
        }
        catch (Exception ex)
        {
```

```
            throw ex;
        }
    }
}


}
```

Entities.cs

```
using Assets.Scripts.Models;
using System.Collections.Generic;

namespace Assets.Scripts
{
    public enum EntitiesType
    {
        Papers,
        PapersAccounts,
        Authors,
        AuthorsPapers,
        AuthorsInstitutions,
        Institutions,
        Uses,
        UsesAccounts
    }

    public static class Entities
    {
        private static Domain _domain;

        public static List<Paper> Papers { get; set; }
        public static List<PaperAccount> PapersAccounts { get; set; }
        public static List<Author> Authors { get; set; }
        public static List<AuthorPaper> AuthorsPapers { get; set; }
        public static List<AuthorInstitution> AuthorsInstitutions { get; set; }
        public static List<Institution> Institutions { get; set; }
        public static List<Use> Uses { get; set; }
        public static List<UseAccount> UsesAccounts { get; set; }

        public static bool LoadingYears { get; set; }
        public static bool LoadingAuthors { get; set; }
        public static bool LoadingInstitutions { get; set; }

        public static Domain Domain {
            get {
                if (_domain == null)
                    _domain = new Domain();
                return _domain;
            }
        }
    }
```

```
}
```

CameraOperate.cs

```
using UnityEngine.EventSystems;
using UnityEngine;
using System.Collections.Generic;
using Assets.Scripts.CustomComponents;

public class CameraOperate : MonoBehaviour
{
    [Tooltip("Mouse wheel rolling control lens please enter, the speed of the back")]
    [Range(0.5f, 2f)] public float scrollSpeed = 1f;
    [Tooltip("Right mouse button control lens X axis rotation speed")]
    [Range(0.5f, 2f)] public float rotateXSpeed = 1f;
    [Tooltip("Right mouse button control lens Y axis rotation speed")]
    [Range(0.5f, 2f)] public float rotateYSpeed = 1f;
    [Tooltip("Mouse wheel press, lens translation speed")]
    [Range(0.5f, 2f)] public float moveSpeed = 1f;
    [Tooltip("The keyboard controls how fast the camera moves")]
    [Range(0.5f, 2f)] public float keyMoveSpeed = 1f;

    public GameObject model;

    //Whether the lens control operation is performed
    public bool operate = true;

    //Whether keyboard control lens operation is performed
    public bool isKeyOperate = true;

    //Whether currently in rotation
    private bool isRotate = false;

    //Is currently in panning
    private bool isMove = false;

    //Camera transform component cache
    private Transform m_transform;

    //The initial position of the camera at the beginning of the operation
    private Vector3 traStart;

    //The initial position of the mouse as the camera begins to operate
    private Vector3 mouseStart;

    //Is the camera facing down
    private bool isDown = false;

    //private Transform objectClick;


    // Start is called before the first frame update
```

```
    void Start()
    {
        m_transform = transform;
    }

    // Update is called once per frame
    void Update()
    {

        if (operate && !IsMouseOverUIIgnore())
        {
            //When in the rotation state, and the left mouse button is released, then exit the rotation
state
            if (isRotate && Input.GetMouseButtonUp(1))
            {
                isRotate = false;
            }
            //When it is in the translation state, and the mouse wheel is released, it will exit the
translation state
            if (isMove && Input.GetMouseButtonUp(0))
            {
                isMove = false;
            }

            //Whether it's in a rotational state
            if (isRotate)
            {
                //Gets the offset of the mouse on the screen
                Vector3 offset = Input.mousePosition - mouseStart;

                // whether the lens is facing down
                if (isDown)
                {
                    // the final rotation Angle = initial Angle + offset, 0.3f coefficient makes the rotation
speed normal when rotateYSpeed, rotateXSpeed is 1
                    GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
                    axis3D.transform.rotation = Quaternion.Euler(traStart + new Vector3(offset.y * 0.3f *
rotateYSpeed, -offset.x * 0.3f * rotateXSpeed, 0));

                }
                else
                {
                    // final rotation Angle = initial Angle + offset
                    GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
                    axis3D.transform.rotation = Quaternion.Euler(traStart + new Vector3(offset.y * 0.3f *
rotateYSpeed, -offset.x * 0.3f * rotateXSpeed, 0));
                }

                // simulate the unity editor operation: left click, the keyboard can control the lens
movement
                if (isKeyOperate)
                {
                    float speed = keyMoveSpeed;
                    // press LeftShift to make speed *2
```

```
            if (Input.GetKey(KeyCode.LeftShift))
            {
                speed = 2f * speed;
            }
            // press W on the keyboard to move the camera forward
            if (Input.GetKey(KeyCode.W))
            {
                m_transform.position += m_transform.forward * Time.deltaTime * 10f * speed;
            }
            // press the S key on the keyboard to back up the camera
            if (Input.GetKey(KeyCode.S))
            {
                m_transform.position -= m_transform.forward * Time.deltaTime * 10f * speed;
            }
            // press A on the keyboard and the camera will turn left
            if (Input.GetKey(KeyCode.A))
            {
                m_transform.position -= m_transform.right * Time.deltaTime * 10f * speed;
            }
            // press D on the keyboard to turn the camera to the right
            if (Input.GetKey(KeyCode.D))
            {
                m_transform.position += m_transform.right * Time.deltaTime * 10f * speed;
            }
        }

        if (model != null)
        {
            foreach (Transform year in model.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(transform.position -
Camera.main.transform.position);
                    }
                }
            }
        }
        // press the right mouse button to enter the rotation state
        else if (Input.GetMouseButtonDown(1) && !IsMouseClickUse())
        {
            // enter the rotation state
            //进入旋转状态
            isRotate = true;
            // record the initial position of the mouse in order to calculate the offset
            //记录鼠标初始位置，为了计算偏移量
            mouseStart = Input.mousePosition;
            // record the initial mouse Angle
            //traStart =  m_transform.rotation.eulerAngles;
```

```
            GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
            traStart = axis3D.transform.localEulerAngles;
            // to determine whether the lens is facing down (the Y-axis is <0 according to the
position of the object facing up),-0.0001f is a special case when x rotates 90
            isDown = m_transform.up.y < -0.0001f ? true : false;
        }

        // whether it is in the translation state
        if (isMove)
        {
            // mouse offset on the screen
            Vector3 offset = Input.mousePosition - mouseStart;
            m_transform.position = traStart + m_transform.up * -offset.y * 0.1f * moveSpeed +
m_transform.right * -offset.x * 0.1f * moveSpeed;
        }
        // press the left mouse button to enter the translation mode
        else if (Input.GetMouseButtonDown(0) && !IsMouseClickUse() && !isMove)
        {
            // translation begins
            isMove = true;
            // record the initial position of the mouse
            mouseStart = Input.mousePosition;
            // record the initial position of the camera
            traStart = m_transform.position;
        }

        // how much did the roller roll
        float scroll = Input.GetAxis("Mouse ScrollWheel");
        // scroll to scroll or not
        if (scroll != 0)
        {
            // position = current position + scroll amount
            if (Input.GetKey(KeyCode.LeftControl))
            {
                m_transform.position += m_transform.forward * scroll * 1000f * Time.deltaTime *
scrollSpeed * 4f;
            }
            else
            {
                m_transform.position += m_transform.forward * scroll * 1000f * Time.deltaTime *
scrollSpeed;
            }

        }


    }
  }

  private bool IsMouseOverUI()
  {
    return EventSystem.current.IsPointerOverGameObject();
  }
```

```
    private bool IsMouseOverUIIgnore()
    {
        var restrictedControls = new List<string> { "SpeedPanel", "ScalePanel", "TimeLinePanel",
"Panel" };
        PointerEventData pointerEventData = new PointerEventData(EventSystem.current);
        pointerEventData.position = Input.mousePosition;

        List<RaycastResult> raycastResultsList = new List<RaycastResult>();
        EventSystem.current.RaycastAll(pointerEventData, raycastResultsList);
        for(int i = 0; i < raycastResultsList.Count; i++)
        {
            if(restrictedControls.Contains(raycastResultsList[i].gameObject.name))
            {
                return true;
            }
        }

        return false;
    }

    private bool IsMouseClickUse()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit))
        {
            var isUse = hit.transform.GetComponent<UseComponent>();
            if (isUse != null)
            {
                return true;
            }
        }

        return false;
    }
}
```

EventHandlers.cs

```
using Assets.Scripts;

using Assets.Scripts.APIManagement;

using Assets.Scripts.CustomComponents;

using Assets.Scripts.Models;

using System.Collections.Generic;

using System.Linq;

using System.Timers;
```

```
using UnityEngine;
using UnityEngine.UI;

public class EventHandlers : MonoBehaviour
{
    public GameObject panelDetails;
    public GameObject esferaPrefab;
    public GameObject slider;
    public GameObject useCases;
    public GameObject speedPanel;
    public GameObject speedLabel;
    public GameObject sliderSpeed;
    public GameObject timeLinePanel;
    public GameObject spreedTimesLabel;
    public GameObject dimensionX;
    public GameObject dimensionY;
    public GameObject dimensionZ;
    public GameObject scalePanel;
    public Slider scaleSlider;
    public Canvas canvasDimensionX;
    public Canvas canvasDimensionY;
    public Canvas canvasDimensionZ;

    public Button buttonPlay;
    public Button buttonPause;
    public Button buttonStop;

    //TODO: Timer
```

```
    private static float initialAnimationSpeed = 2f; //2000f;
    private static float initialScale = 50;

    private float currentAnimationSpeed = 0;
    private float currentScale = 0;
    private bool started = false;
    private bool paused = false;

    private List<(int index, int year, GameObject yearObject)> ControlTimer = new
List<(int index, int year, GameObject yearObject)>();
    private int currYearToShow = -1;

    private int currIndexShow = -1;
    private int updatedIndex = -1;

    private FunctionTimer functionTimer;

    List<int> years = new List<int>();

    // Start is called before the first frame update
    void Start()
    {
        if (scaleSlider != null)
        {
            scaleSlider.onValueChanged.AddListener(delegate { SetScale(); });
        }
        if(buttonPlay != null)
        {
```

```
        buttonPlay.interactable = true;
        buttonPause.interactable = false;
        buttonStop.interactable = false;
    }

}

public void BtnPlay_Click()
{
    if(started && !paused)
    {
        BtnStop_Click();
    }

    if (!started)
    {
        GameObject yearsContent =
GameObject.FindGameObjectWithTag("YearsContent");
        GameObject authorsContent =
GameObject.FindGameObjectWithTag("AuthorsContent");
        GameObject institutionsContent =
GameObject.FindGameObjectWithTag("InstitutionsContent");

        GameObject graph = GameObject.FindGameObjectWithTag("Graph");
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");

        years.Clear();
        ControlTimer.Clear();
        List<int> authors = new List<int>();
```

```
List<int> institutions = new List<int>();
slider.GetComponent<Slider>().maxValue = 0;

//Consider only selected Years
foreach (Transform child in yearsContent.transform)
{
    Toggle toggle = child.GetComponent<Toggle>();
    if (toggle.isOn)
    {
        Text text = child.GetComponentInChildren<Text>();
        years.Add(int.Parse(text.text));
    }
}

//Consider only selected Authors
foreach (Transform child in authorsContent.transform)
{
    Toggle toggle = child.GetComponent<Toggle>();
    if (toggle.IsActive() && toggle.isOn)
    {
        AuthorComponent author = child.GetComponent<AuthorComponent>();
        authors.Add(author.AuthorId);
    }
}

//Consider only selected Institutions
foreach (Transform child in institutionsContent.transform)
{
```

```
            Toggle toggle = child.GetComponent<Toggle>();
            if (toggle.IsActive() && toggle.isOn)
            {
                InstitutionComponent institution =
child.GetComponent<InstitutionComponent>();
                institutions.Add(institution.InstitutionId);
            }
        }

        if (years.Where(y => y > 0).Any())
        {
            years.Sort();

            var selectedUseCase = useCases.GetComponent<Dropdown>();
            if (selectedUseCase != null)
            {
                var apiCalls = new APIHelper();
                var includealluptoyear = selectedUseCase.value == 0;

                var models = Entities.Domain.GenerateModels(years, authors, institutions,
includealluptoyear);
                AddModel(models);
            }

        }


    }
```

```
    if (started && paused)
    {
        paused = false;
        buttonPlay.interactable = false;
        buttonPause.interactable = true;
        buttonStop.interactable = true;
    }

}

public void AddModel(List<(int Year, List<UseModel> Uses)> models)
{
    try
    {
        if (!started)
        {
            GameObject graph = GameObject.FindGameObjectWithTag("Graph");

            var index = 0;
            ControlTimer.Clear();
            foreach (var model in models)
            {
                GameObject newYear;

                newYear = new GameObject(model.Year.ToString());
                newYear.SetActive(false);
                newYear.transform.SetParent(graph.transform, false);
```

```
            ControlTimer.Add((index++, model.Year, newYear));


            foreach (var use in model.Uses)
            {
                GameObject e1 = Instantiate(esferaPrefab);
                float scale = ((float)use.PapersCount) * 1.2f;
                e1.transform.localScale = new Vector3(scale, scale, scale);
                e1.transform.parent = newYear.transform;


                GameObject golab = new GameObject("Lab1");
                TextMesh objlabel = golab.AddComponent<TextMesh>();
                objlabel.text = use.Name;
                objlabel.alignment = TextAlignment.Center;
                objlabel.anchor = TextAnchor.UpperRight;
                objlabel.characterSize = 0.6f;
                objlabel.color = Color.white;
                objlabel.transform.parent = e1.transform;
                objlabel.transform.position = e1.transform.position + new Vector3(0,
scale / 2f + 0.5f, 0);



                e1.transform.localPosition = new Vector3(use.NarrativeImmersion,
use.SystemImmersion, use.ChallengeImmersion * (-1));
                e1.transform.localPosition *= scaleSlider.value;


                Color color;
                ColorUtility.TryParseHtmlString(use.RGBColor, out color);
                e1.GetComponent<Renderer>().material.color = color;
```

```
        var useComponent = e1.GetComponent<UseComponent>();
        useComponent.Year = use.Year;
        useComponent.UseId = use.UseId;
        useComponent.Name = use.Name;
        useComponent.PapersCount = use.PapersCount;
        useComponent.AuthorsCount = use.AuthorsCount;
        useComponent.InstitutionsCount = use.InstitutionsCount;
        useComponent.Papers = use.Papers;
        useComponent.NarrativeImmersion = use.NarrativeImmersion;
        useComponent.SystemImmersion = use.SystemImmersion;
        useComponent.ChallengeImmersion = -use.ChallengeImmersion;

    }

}

currIndexShow = -1;
currYearToShow = -1;

if (ControlTimer.Count() > 1)
    slider.GetComponent<Slider>().maxValue = ControlTimer.Count() - 1;

scalePanel.gameObject.SetActive(true);
timeLinePanel.gameObject.SetActive(true);
var sliderComponent = slider.GetComponent<Slider>();
sliderComponent.value = 0;
speedPanel.gameObject.SetActive(true);
updatedIndex = 0;
```

```
            currentAnimationSpeed = initialAnimationSpeed;
            currentScale = initialScale;
            var sliderSpeedComponent = sliderSpeed.GetComponent<Slider>();
            sliderSpeedComponent.value = sliderSpeedComponent.maxValue / 2;

            var lableSpeed =
spreedTimesLabel.GetComponent<TMPro.TextMeshProUGUI>();
            lableSpeed.text = $"1.0x";
            started = true;
            buttonPlay.interactable = false;
            buttonPause.interactable = true;
            buttonStop.interactable = true;
            SetTimer();


        }

    }
    catch (System.Exception ex)
    {

        throw ex;
    }
}


/// <summary>
/// User is trying to filter by a specific value
```

```
    /// </summary>
    public void BtnSearchAuthor_Click()
    {
        //This search is only active when presentation is not in progress
        if (!started)
        {
            //Find the container os values to be filter
            GameObject authorsContent =
GameObject.FindGameObjectWithTag("AuthorsContent");


            //Find the filter field
            GameObject searchAuthors =
GameObject.FindGameObjectWithTag("SearchAuthors");
            TMPro.TMP_InputField filterValue =
searchAuthors.GetComponent<TMPro.TMP_InputField>();


            //Iterate through values of container to apply the filter
            foreach (Transform child in authorsContent.transform)
            {
                AuthorComponent author = child.GetComponent<AuthorComponent>();


                if (filterValue.text.Equals(""))
                {
                    //The filter value is empty meaning the user wants to clear all previous
selection
                    child.gameObject.SetActive(true);
                }
                else if (author != null && author.AuthorName != null)
                {
```

```
                //The checkbox is set to active/inactive according to the filter criteria
                child.gameObject.SetActive(
                    author.AuthorName.ToUpper().Contains(filterValue.text.ToUpper())
                );
            }
        }
    }
}

public void BtnSearchInstitution_Click()
{
    if (!started)
    {
        GameObject institutionsContent =
GameObject.FindGameObjectWithTag("InstitutionsContent");
        GameObject searchInstitutions =
GameObject.FindGameObjectWithTag("SearchInstitutions");
        TMPro.TMP_InputField val =
searchInstitutions.GetComponent<TMPro.TMP_InputField>();
        //Consider only selected Institutions
        foreach (Transform child in institutionsContent.transform)
        {
            InstitutionComponent institution =
child.GetComponent<InstitutionComponent>();

            if (val.text.Equals(""))
            {
                child.gameObject.SetActive(true);
            }
```

```
            else if (institution != null && institution.InstitutionName != null)
            {

child.gameObject.SetActive(institution.InstitutionName.ToUpper().Contains(val.text.
ToUpper()));
            }


        }
    }
}


public void BtnPause_Click()
{
    paused = true;
    buttonPlay.interactable = true;
    buttonPause.interactable = false;
    buttonStop.interactable = true;
}


public void BtnStop_Click()
{
    started = false;
    paused = false;


    buttonPlay.interactable = true;
    buttonPause.interactable = false;
    buttonStop.interactable = false;
```

```
    GameObject graph = GameObject.FindGameObjectWithTag("Graph");
    GameObject yearLable = GameObject.FindGameObjectWithTag("YearText");

    var lable = yearLable.GetComponent<TMPro.TextMeshProUGUI>();
    lable.text = "";

    foreach (Transform child in graph.transform)
    {
        Destroy(child.gameObject);
    }

    if(functionTimer != null)
    {
        functionTimer.Destroy();
        functionTimer = null;
    }

    timeLinePanel.gameObject.SetActive(false);
    speedPanel.gameObject.SetActive(false);
    scalePanel.gameObject.SetActive(false);
    var sliderComponent = slider.GetComponent<Slider>();
    sliderComponent.value = 0;
    years.Clear();
    ControlTimer.Clear();
}

// Update is called once per frame
void Update()
```

```
  {


    if (started)
    {
        if (updatedIndex != currIndexShow)
        {
            currIndexShow = updatedIndex;
            slider.GetComponent<Slider>().value = currIndexShow;
        }


        GameObject graph = GameObject.FindGameObjectWithTag("Graph");
        GameObject yearLable = GameObject.FindGameObjectWithTag("YearText");


        var currIndex = slider.GetComponent<Slider>().value;


        if (paused)
            updatedIndex = int.Parse(currIndex.ToString());


        var yearControl = ((int index, int year, GameObject yearObject))
ControlTimer.Where(o => o.index == currIndex).FirstOrDefault();


        if (yearControl.year != currYearToShow)
        {
            currYearToShow = yearControl.year;
            foreach (var val in ControlTimer)
            {
                val.yearObject.gameObject.SetActive(false);
            }
```

```
            yearLable.GetComponent<TMPro.TextMeshProUGUI>().text =
currYearToShow.ToString();
            yearControl.yearObject.gameObject.SetActive(true);
        }
    }
}

public void BtnAllYears_Click()
{
    GameObject yearsContent =
GameObject.FindGameObjectWithTag("YearsContent");
    foreach (Transform child in yearsContent.transform)
    {
        Toggle toggle = child.GetComponent<Toggle>();
        toggle.isOn = true;
    }
}

public void BtnNoneYears_Click()
{
    GameObject yearsContent =
GameObject.FindGameObjectWithTag("YearsContent");
    foreach (Transform child in yearsContent.transform)
    {
        Toggle toggle = child.GetComponent<Toggle>();
        toggle.isOn = false;
    }
```

```
    }


    public void BtnAllAuthors_Click()
    {
        GameObject authorsContent =
GameObject.FindGameObjectWithTag("AuthorsContent");
        foreach (Transform child in authorsContent.transform)
        {
            Toggle toggle = child.GetComponent<Toggle>();
            toggle.isOn = true;
        }
    }


    public void BtnNoneAuthors_Click()
    {
        GameObject authorsContent =
GameObject.FindGameObjectWithTag("AuthorsContent");
        foreach (Transform child in authorsContent.transform)
        {
            Toggle toggle = child.GetComponent<Toggle>();
            toggle.isOn = false;
        }
    }


    public void BtnAllInstitutions_Click()
    {
        GameObject institutionsContent =
GameObject.FindGameObjectWithTag("InstitutionsContent");
        foreach (Transform child in institutionsContent.transform)
```

```
        {
            Toggle toggle = child.GetComponent<Toggle>();
            toggle.isOn = true;
        }
    }

    public void BtnNoneInstitutions_Click()
    {
        GameObject institutionsContent =
GameObject.FindGameObjectWithTag("InstitutionsContent");
        foreach (Transform child in institutionsContent.transform)
        {
            Toggle toggle = child.GetComponent<Toggle>();
            toggle.isOn = false;
        }
    }

    private void SetTimer()
    {
        functionTimer = FunctionTimer.Create(UpdateUi, currentAnimationSpeed);
    }

    private void UpdateUi()
    {
        if (started && !paused)
        {
            updatedIndex = (currIndexShow + 1) % ControlTimer.Count;
        }
```

```
}

private void OnTimedEvent(object source, ElapsedEventArgs e)
{
    if (started && !paused)
    {
        updatedIndex = (currIndexShow + 1) % ControlTimer.Count;
    }
}

public void SetAnimationSpeed(float speed)
{
    //initial animationSpeed = 2000 => speed = 4
    currentAnimationSpeed = (float)(speed * initialAnimationSpeed) / 4;

    if(functionTimer != null)
        functionTimer.UpdateSpeed(currentAnimationSpeed);



    float times = (float)(8 - speed) / 4;
    var lable = spreedTimesLabel.GetComponent<TMPro.TextMeshProUGUI>();
    lable.text = $"{times.ToString("0.0")}x";
}

public void SetScale()
{
    var dimX = dimensionX.GetComponent<LineRenderer>();
    dimX.SetPosition(1, new Vector3(scaleSlider.value, 0, 0));
```

```
        canvasDimensionX.GetComponent<RectTransform>().localPosition = new
Vector3(scaleSlider.value, 0, 0);



        var dimY = dimensionY.GetComponent<LineRenderer>();

        dimY.SetPosition(1, new Vector3(0, scaleSlider.value, 0));

        canvasDimensionY.GetComponent<RectTransform>().localPosition = new
Vector3(0, scaleSlider.value, 0);



        var dimZ = dimensionZ.GetComponent<LineRenderer>();

        dimZ.SetPosition(1, new Vector3(0, 0, -scaleSlider.value));

        canvasDimensionZ.GetComponent<RectTransform>().localPosition = new
Vector3(0, 0, -scaleSlider.value);



        SetScaleModels(scaleSlider.value);

    }

    private void SetScaleModels(float scale)

    {

        foreach (var val in ControlTimer)

        {

            foreach(Transform child in val.yearObject.transform)

            {

                var useComponent = child.GetComponent<UseComponent>();

                var originalPosition = new Vector3(useComponent.NarrativeImmersion,
useComponent.SystemImmersion, useComponent.ChallengeImmersion);

                child.transform.localPosition = originalPosition * scale;

            }

        }

    }

}
```

FunctionTimer.cs

```
using System;
using UnityEngine;

namespace Assets.Scripts
{
    public class FunctionTimer
    {
        public static FunctionTimer Create(Action action, float timer)
        {
            GameObject gameObject = new GameObject("FunctionTimer",
typeof(MonoBehaviourHook));

            FunctionTimer functionTimer = new FunctionTimer(action, timer, gameObject);

            gameObject.GetComponent<MonoBehaviourHook>().onUpdate = functionTimer.Update;

            return functionTimer;
        }
        public class MonoBehaviourHook : MonoBehaviour
        {
            public Action onUpdate;

            private void Update()
            {
                if(onUpdate != null) onUpdate();
            }
        }

        private Action action;
        private float timer;
        private float initialTimer;
        private GameObject gameObject;
        private bool isDestroyed;

        private FunctionTimer(Action action, float timer, GameObject gameObject)
        {
            this.action = action;
            this.initialTimer = timer;
            this.timer = timer;
            this.gameObject = gameObject;
            isDestroyed = false;
        }

        public void Update()
        {
            if(!isDestroyed)
            {
                timer -= Time.deltaTime;
                if (timer < 0)
                {
                    //Trigger the action
                    action();
```

```
                timer = initialTimer;
            }
        }
    }

    public void UpdateSpeed(float newSpeed)
    {
        initialTimer = newSpeed;
    }

    public void Destroy()
    {
        isDestroyed = true;
        UnityEngine.Object.Destroy(gameObject);
    }
  }
}
```

LoadingAuthors.cs

```
using Assets.Scripts;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadingAuthors : MonoBehaviour
{
  private RectTransform rectComponent;
  private float rotateSpeed = 200f;

  private void Start()
  {
    rectComponent = GetComponent<RectTransform>();
  }

  private void Update()
  {
    if (Entities.LoadingAuthors)
    {
      rectComponent.Rotate(0f, 0f, rotateSpeed * Time.deltaTime);
    }
    else
    {
      this.gameObject.transform.parent.gameObject.SetActive(false);
    }
  }
}
```

LoadingInstitutions.cs

```
using Assets.Scripts;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadingInstitutions : MonoBehaviour
{
    private RectTransform rectComponent;
    private float rotateSpeed = 200f;

    private void Start()
    {
        rectComponent = GetComponent<RectTransform>();
    }

    private void Update()
    {
        if (Entities.LoadingInstitutions)
        {
            rectComponent.Rotate(0f, 0f, rotateSpeed * Time.deltaTime);
        }
        else
        {
            this.gameObject.transform.parent.gameObject.SetActive(false);
        }
    }
}
```

LoadingYears.cs

```
using Assets.Scripts;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadingYears : MonoBehaviour
{
    private RectTransform rectComponent;
    private float rotateSpeed = 200f;

    private void Start()
    {
        rectComponent = GetComponent<RectTransform>();
    }

    private void Update()
    {
        if(Entities.LoadingYears)
        {
            rectComponent.Rotate(0f, 0f, rotateSpeed * Time.deltaTime);
```

```
        }
        else
        {
            this.gameObject.transform.parent.gameObject.SetActive(false);
        }
    }
}
```

TooltipManager.cs

```
using UnityEngine;
using TMPro;

public class TooltipManager : MonoBehaviour
{
    public static TooltipManager _instance;
    public TextMeshProUGUI textComponent;

    private void Awake()
    {
        if (_instance != null && _instance != this)
        {
            Destroy(this.gameObject);
        }
        else
        {
            _instance = this;
        }
    }

    // Start is called before the first frame update
    void Start()
    {
        Cursor.visible = true;
        gameObject.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        transform.position = Input.mousePosition;
    }

    public void SetAndShowTooltip(string message)
    {
        gameObject.SetActive(true);
        textComponent.text = message;
    }

    public void HideTooltip()
    {
        gameObject.SetActive(false);
```

```
        textComponent.text = string.Empty;
    }
}
```

UseEvents.cs

```
using Assets.Scripts;
using Assets.Scripts.CustomComponents;
using Assets.Scripts.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using UnityEngine;
using UnityEngine.EventSystems;

public class UseEvents : MonoBehaviour
{

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnMouseDown()
    {
        try
        {
            if (!IsMouseOverUIIgnore())
            {
                UseComponent useComponent = transform.GetComponent<UseComponent>();

                var content = JsonConvert.SerializeObject(GenerateUseDetails(useComponent));
                var id = Guid.NewGuid();
                JSFunctionsHelper.SetUseDetails($"{id}", content);
                var url = Path.Combine(Application.streamingAssetsPath, $"usedetails.html?id={id}");
                Application.OpenURL(url);
            }

        }
        catch (System.Exception ex)
        {

            throw ex;
```

```
    }

}

private UseDetails GenerateUseDetails(UseComponent useComponent)
{
    var useDetails = new UseDetails
    {
        Id = useComponent.UseId,
        Name = useComponent.Name,
        PapersCount = useComponent.PapersCount,
        AuthorsCount = useComponent.AuthorsCount,
        InstitutionsCount = useComponent.InstitutionsCount,
        Papers = new List<PaperDetails>()
    };

    foreach (var pap in useComponent.Papers)
    {
        var paper = Entities.Papers.Where(p => p.Id.Equals(pap)).FirstOrDefault();

        var newPaper = new PaperDetails
        {
            Id = paper.Id,
            Title = paper.Title,
            Date = paper.Date,
            PubYear = paper.PubYear,
            OpenAlexId = paper.OpenAlexId,
            Authors = new List<AuthorDetails>()
        };

        var authorsPaper = (from a in Entities.Authors
                            join ap in Entities.AuthorsPapers on a.Id equals ap.AuthorId
                            where ap.PaperId.Equals(pap)
                            select a).ToList();

        foreach (var author in authorsPaper)
        {
            var newAuthor = new AuthorDetails
            {
                Id = author.Id,
                Name = author.Name,
                OpenAlexId = author.OpenAlexId
            };
            newAuthor.Institutions = (from i in Entities.Institutions
                                      join ai in Entities.AuthorsInstitutions on i.Id equals ai.InstitutionId
                                      where ai.AuthorId.Equals(author.Id)
                                      select i).ToList();

            newPaper.Authors.Add(newAuthor);
        }
        useDetails.Papers.Add(newPaper);
    }

    var years = useDetails.Papers.Select(p => p.PubYear).ToList().Distinct();
```

```
        useDetails.StartYear = years.OrderBy(year => year).First();
        useDetails.FinalYear = years.OrderBy(year => year).Last();

        return useDetails;
    }

    private void OnMouseEnter()
    {
        if(!IsMouseOverUIIgnore())
        {
            UseComponent useComponent = transform.GetComponent<UseComponent>();
            //UseModel use = new UseModel(); // = useComponent.Use;

            if (useComponent != null)
            {
                string description = $"Use: {useComponent.Name}" +
                                     $"\n\nPapers: {useComponent.PapersCount}" +
                                     $"\nAuthors: {useComponent.AuthorsCount}" +
                                     $"\nInstitutions: {useComponent.InstitutionsCount}";
                TooltipManager._instance.SetAndShowTooltip(description);
            }
        }
    }

    private void OnMouseExit()
    {
        TooltipManager._instance.HideTooltip();
    }

    private bool IsMouseOverUIIgnore()
    {
        var restrictedControls = new List<string> { "SpeedPanel", "ScalePanel", "TimeLinePanel",
"Panel" };
        PointerEventData pointerEventData = new PointerEventData(EventSystem.current);
        pointerEventData.position = Input.mousePosition;

        List<RaycastResult> raycastResultsList = new List<RaycastResult>();
        EventSystem.current.RaycastAll(pointerEventData, raycastResultsList);
        for (int i = 0; i < raycastResultsList.Count; i++)
        {
            if (restrictedControls.Contains(raycastResultsList[i].gameObject.name))
            {
                return true;
            }
        }

        return false;
    }
}
```

JSFunctionsHelper.cs

```
using System.Runtime.InteropServices;
using UnityEngine;

namespace Assets.Scripts
{
    public static class JSFunctionsHelper
    {
#if UNITY_WEBGL && !UNITY_EDITOR
        [DllImport("__Internal")]
        private static extern void downloadToFile(string content, string filename);

        [DllImport("__Internal")]
        private static extern void setUseDetails(string index, string content);
#endif

        public static void DownloadToFile(string content, string filename)
        {
#if UNITY_WEBGL && !UNITY_EDITOR
            downloadToFile(content, filename);
#endif
        }

        public static void SetUseDetails(string index, string content)
        {
#if UNITY_WEBGL && !UNITY_EDITOR
            setUseDetails(index, content);
#endif
        }
    }
}
```

Rotate/CenterPosition.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class CenterPosition : MonoBehaviour, IPointerDownHandler
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
```

```
    public void OnPointerDown(PointerEventData eventData)
    {
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
        axis3D.transform.rotation = Quaternion.Euler(0, 0, 0);

        Camera.main.transform.position = new Vector3(19.27f, 10.5f, -488.9f);
        RotateUseLabels();
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Rotate/RotateDown.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class RotateDown : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        float rotateXSpeed = 1f;
```

```
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
        axis3D.transform.rotation = Quaternion.Euler(axis3D.transform.localEulerAngles + new
Vector3(-1 * rotateXSpeed, 0, 0));
        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Rotate/RotateLeft.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class RotateLeft : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;

    // Start is called before the first frame update
    void Start()
    {

    }
```

```
    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        float rotateXSpeed = 1f;
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
        axis3D.transform.rotation = Quaternion.Euler(axis3D.transform.localEulerAngles + new
Vector3(0, 1 * rotateXSpeed, 0));
        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Rotate/RotateRight.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class RotateRight : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;
```

```
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        float rotateXSpeed = 1f;
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
        axis3D.transform.rotation = Quaternion.Euler(axis3D.transform.localEulerAngles + new
Vector3(0, -1 * rotateXSpeed, 0));
        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Rotate/RotateUp.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class RotateUp : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        float rotateXSpeed = 1f;
        GameObject axis3D = GameObject.FindGameObjectWithTag("3DAxis");
        axis3D.transform.rotation = Quaternion.Euler(axis3D.transform.localEulerAngles + new
Vector3(1 * rotateXSpeed, 0, 0));
        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
```

```
        }
    }
  }
}
```

Rotate/ZoomIn.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class ZoomIn : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;
    private float scrollSpeed = 1f;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        if (Input.GetKey(KeyCode.LeftControl))
        {
            Camera.main.transform.position += Camera.main.transform.forward * 0.1f * 1000f *
Time.deltaTime * scrollSpeed * 4f;
        }
        else
        {
            Camera.main.transform.position += Camera.main.transform.forward * 0.1f * 1000f *
Time.deltaTime * scrollSpeed;
        }

        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
```

```
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Rotate/ZoomOut.cs

```
using UnityEngine;
using UnityEngine.EventSystems;

public class ZoomOut : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    private bool ispressed = false;
    private float scrollSpeed = 1f;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (!ispressed)
            return;

        if (Input.GetKey(KeyCode.LeftControl))
        {
            Camera.main.transform.position += Camera.main.transform.forward * -0.1f * 1000f *
Time.deltaTime * scrollSpeed * 4f;
        }
        else
        {
            Camera.main.transform.position += Camera.main.transform.forward * -0.1f * 1000f *
Time.deltaTime * scrollSpeed;
        }
```

```
        RotateUseLabels();
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        ispressed = true;
    }

    public void OnPointerUp(PointerEventData eventData)
    {
        ispressed = false;
    }

    private void RotateUseLabels()
    {
        GameObject graph = GameObject.FindGameObjectWithTag("Graph");

        if (graph != null)
        {
            foreach (Transform year in graph.transform)
            {
                foreach (Transform model in year.transform)
                {
                    foreach (Transform label in model.transform)
                    {
                        label.transform.rotation = Quaternion.LookRotation(-
Camera.main.transform.position);
                    }
                }
            }
        }
    }
}
```

Anexo II – WebAPI Swagger

O serviço disponibiliza uma página de documentação, gerada automaticamente em Swagger [Swagger 2022], que no momento de envio deste relatório está disponível acedendo ao URL https://immersivelearningapi.azurewebsites.net/swagger

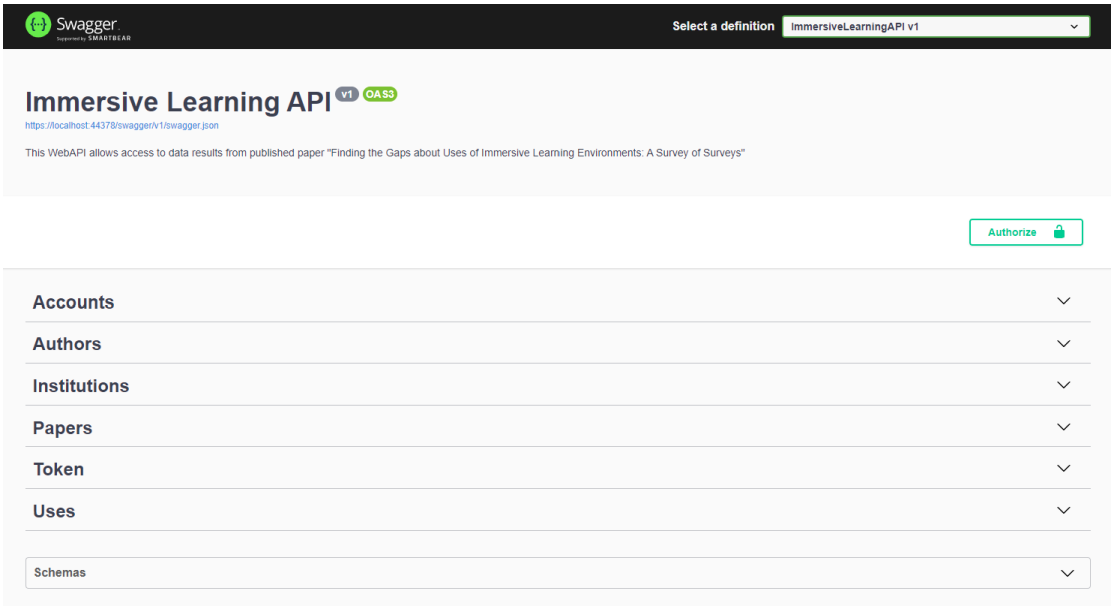


Figura 51: Documentação gerada pelo Swagger para o serviço WebAPI

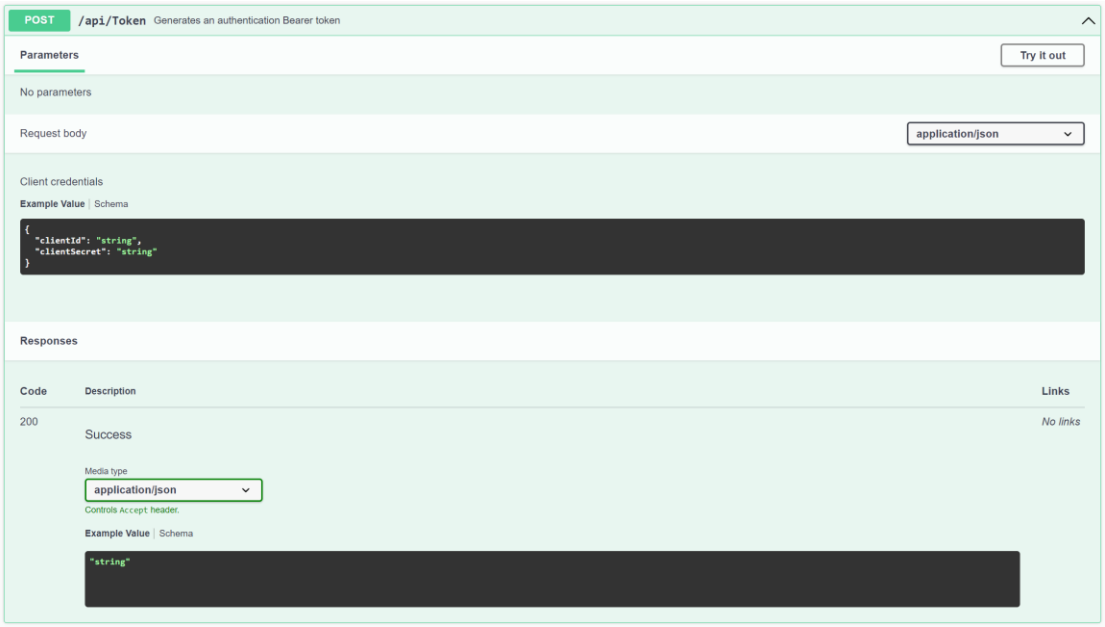


Figura 52: api/token

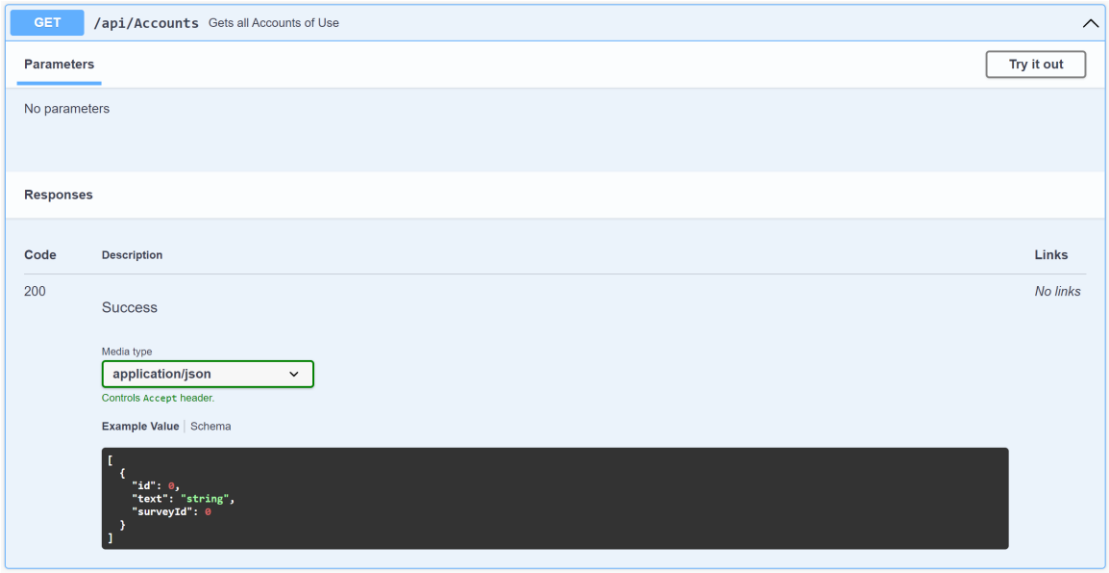


Figura 53: api/accounts

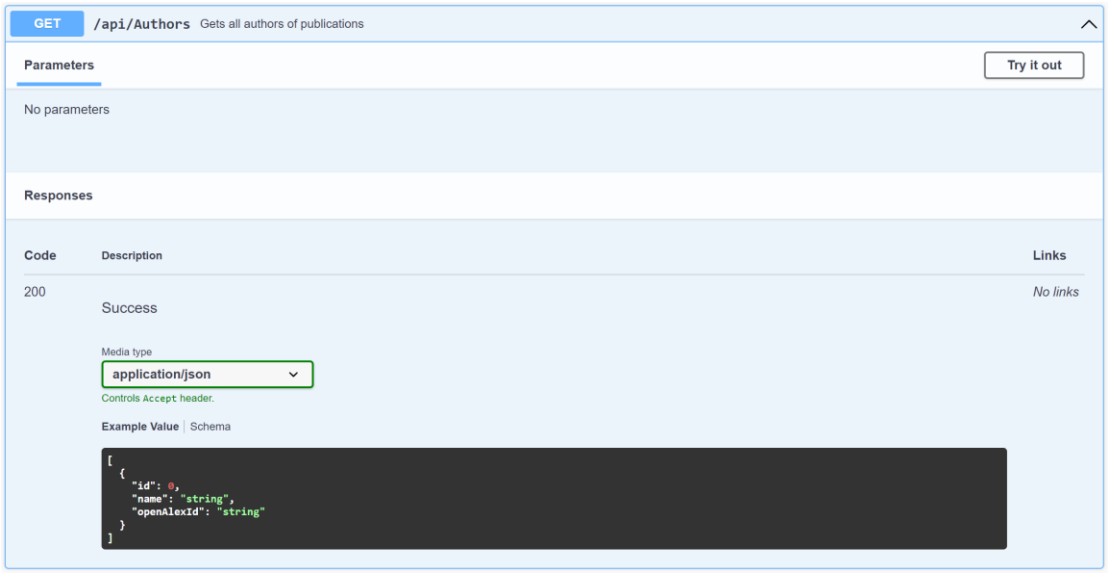


Figura 54: api/authors

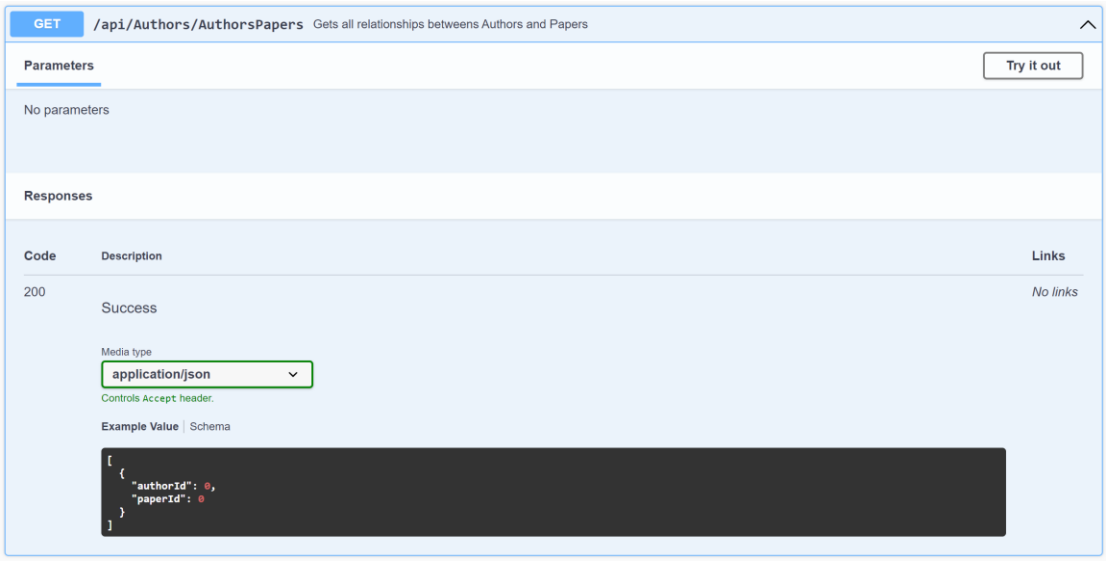


Figura 55: api/authorspapers

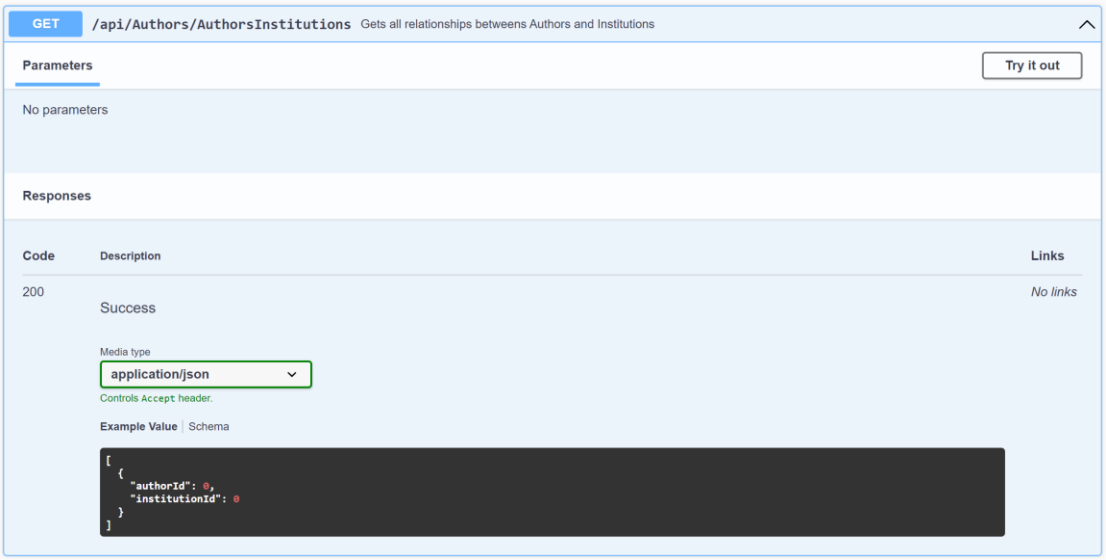


Figura 56: api/authorsinstitutions

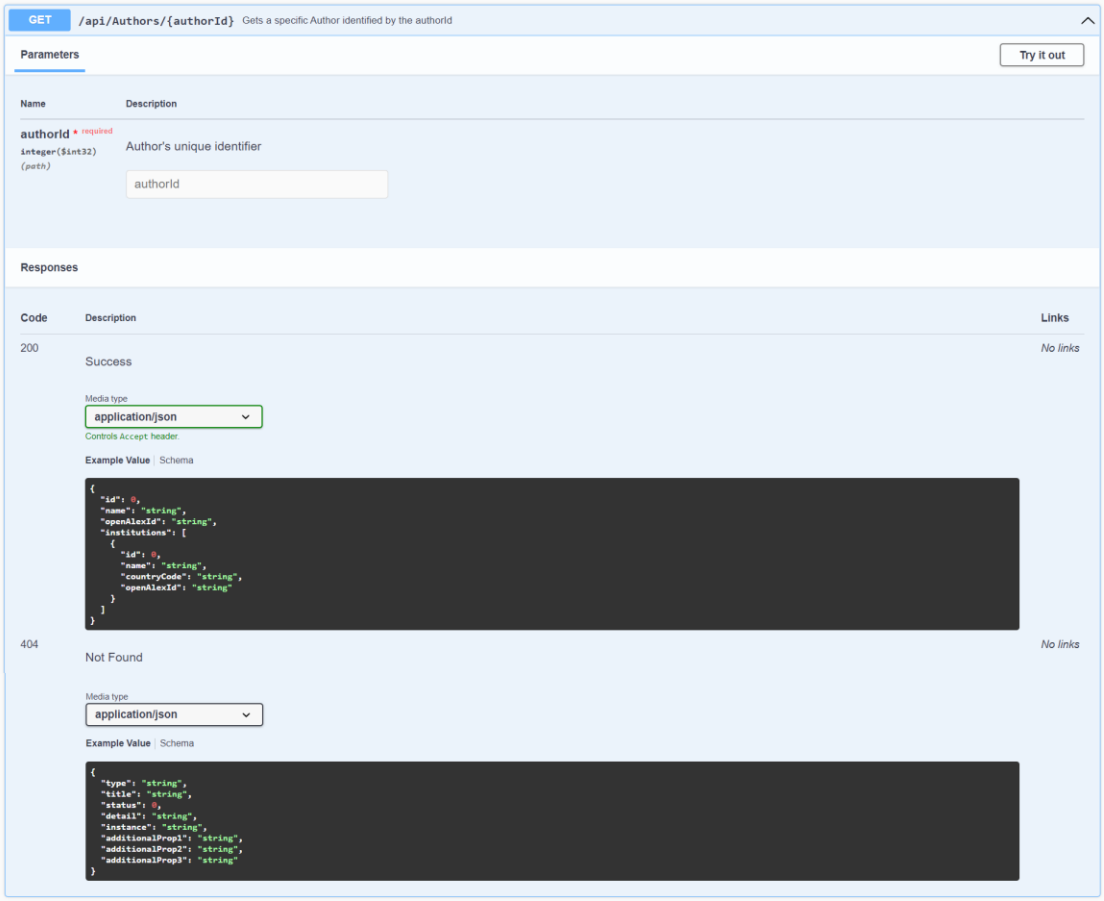


Figura 57: api/authors/{authorId}

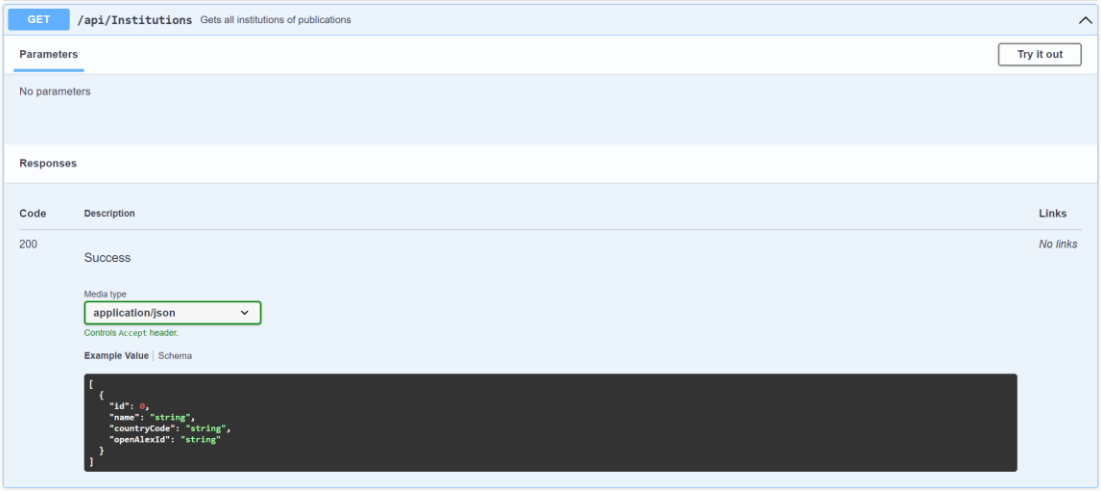


Figura 58: api/institutions

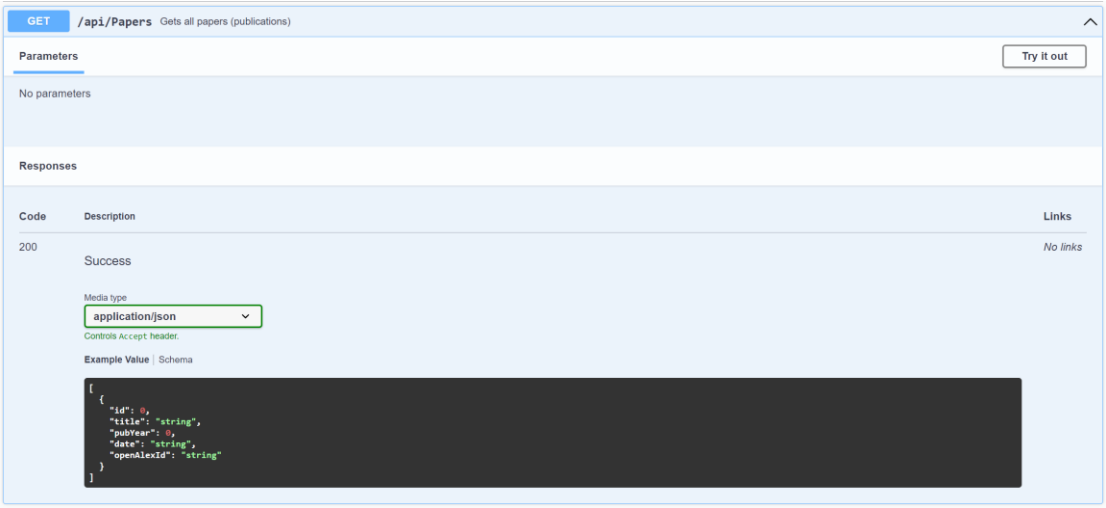


Figura 59: api/papers

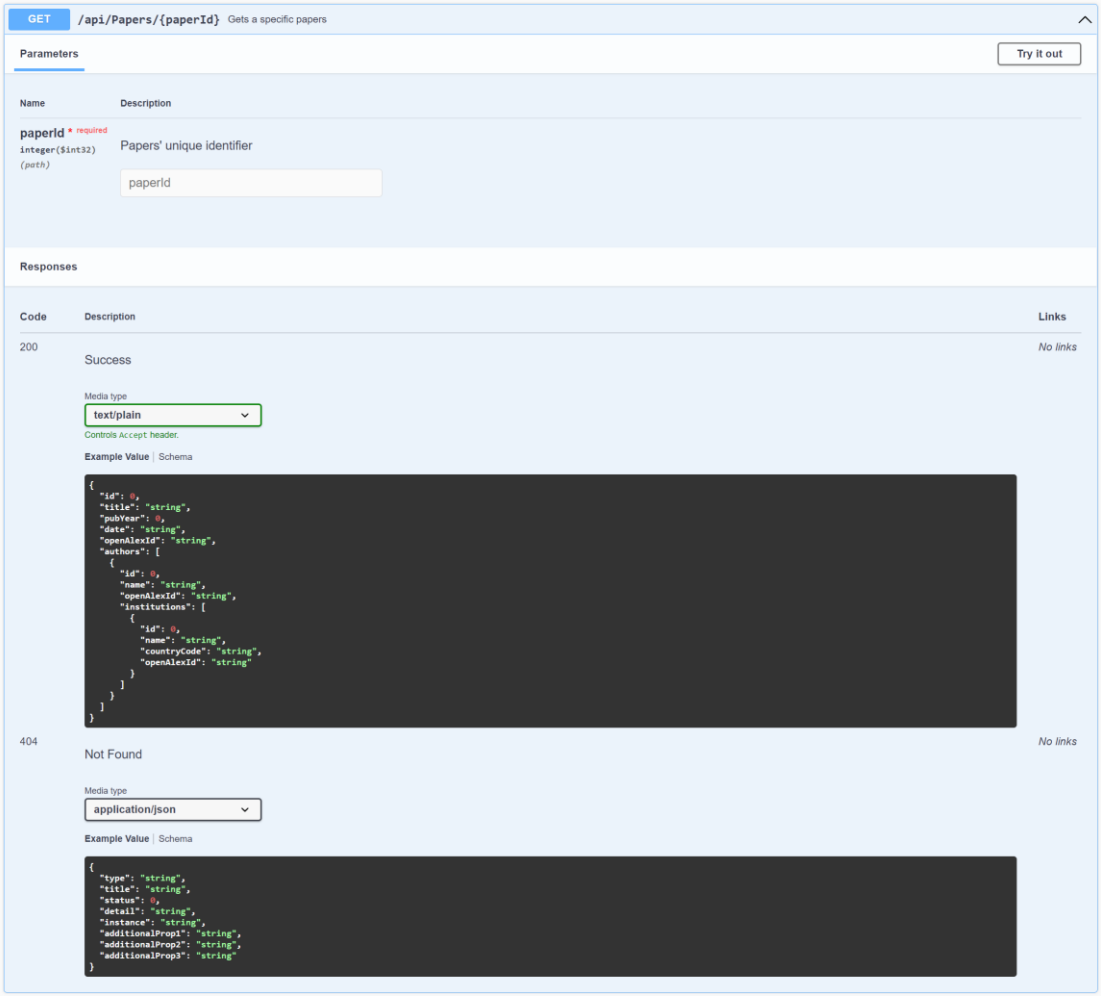


Figura 60: api/papers/{paperId}

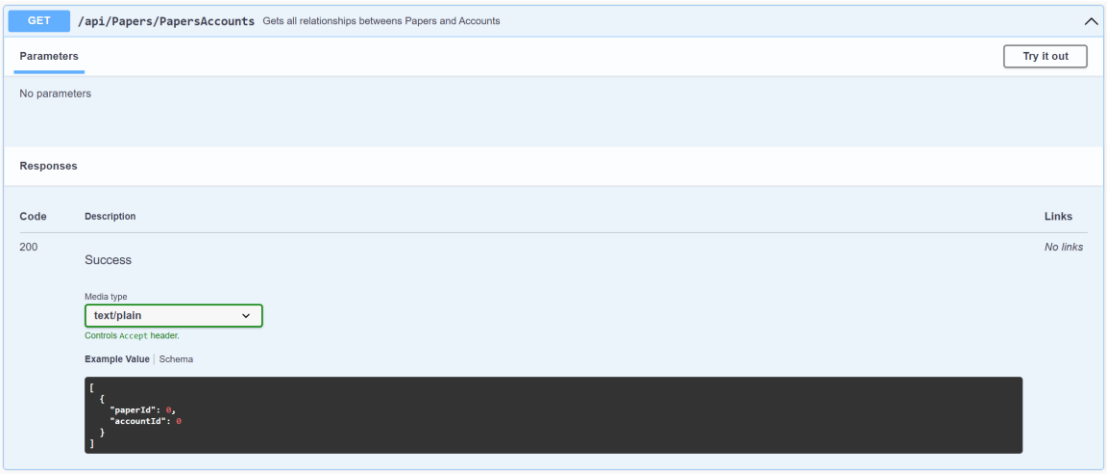


Figura 61: api/papers/papersaccounts

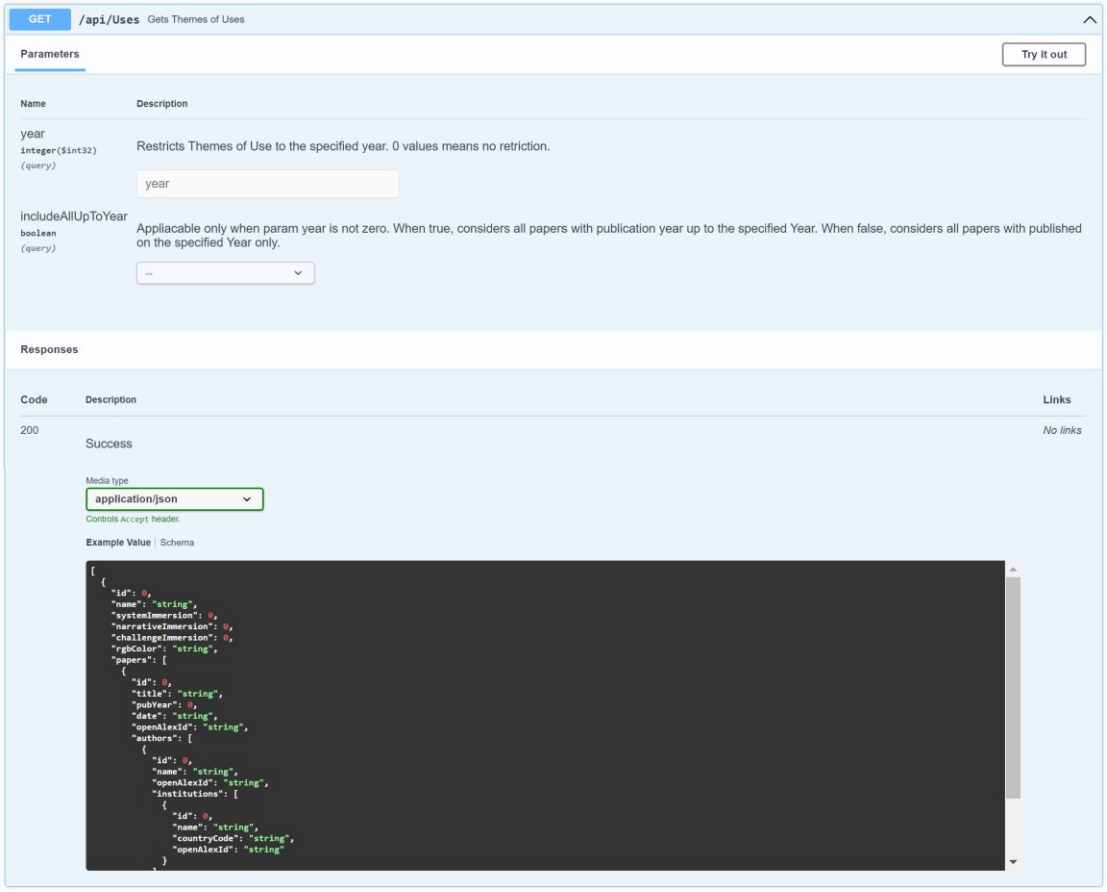


Figura 62: api/uses

Figura 63: api/uses/usesaccounts

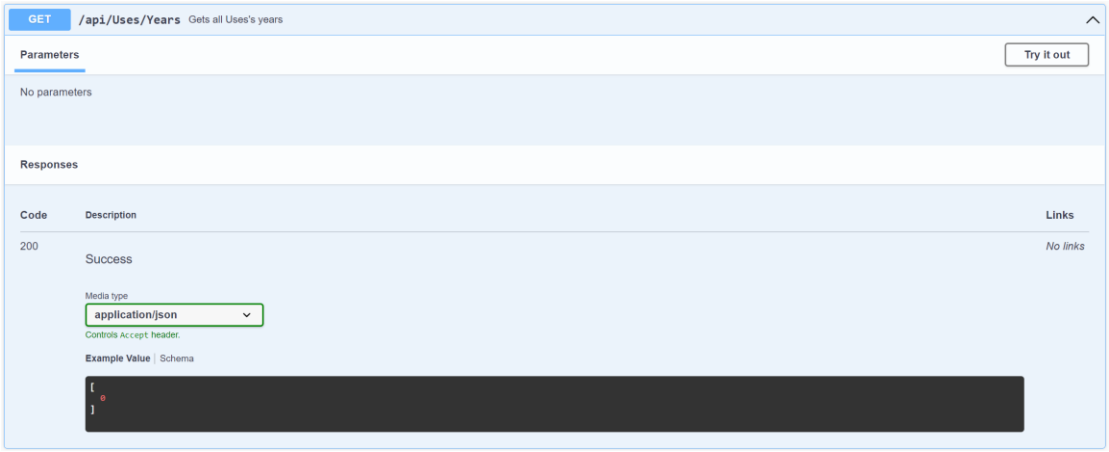


Figura 64: api/uses/years

Anexo III – Serviço WebAPI

Neste anexo, pode-se encontrar todo o código implementado para o desenvolvimento do serviço WebAPI. À data de envio deste relatório, este serviço está disponível em https://immersivelearningapi.azurewebsites.net

ImmersiveLearningAPI.DataModels

AccountModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("account")]
    public class AccountModel
    {
        public int Id { get; set; }
        public string Text { get; set; }
        [Column("survey_id")]
        public int SurveyId { get; set; }
    }
}
```

AuthorInstitutionModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("author_institution")]
    public class AuthorInstitutionModel
    {
        [Key, Column("author_id", Order = 0)]
        public int AuthorId { get; set; }

        [Key, Column("institution_id", Order = 1)]
        public int InstitutionId { get; set; }
    }
}
```

AuthorModel.cs

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("author")]
    public class AuthorModel
    {
        [Key]
        public int Id { get; set; }
        public string Name { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

AuthorPaperModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("author_paper")]
    public class AuthorPaperModel
    {
        [Key, Column("author_id", Order = 0)]
        public int AuthorId { get; set; }

        [Key, Column("paper_id", Order = 1)]
        public int PaperId { get; set; }
    }
}
```

InstitutionModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("institution")]
    public class InstitutionModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string CountryCode { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

PaperAccountModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("paper_account")]
    public class PaperAccountModel
    {
        [Column("paper_id")]
        public int PaperId { get; set; }
        [Column("account_id")]
        public int AccountId { get; set; }
    }
}
```

PaperModel.cs

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("paper")]
    public class PaperModel
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int PubYear { get; set; }
        public string Date { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

UseAccountModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("uses_account")]
    public class UseAccountModel
    {
        [Key, Column("uses_id", Order = 0)]
        public int UseId { get; set; }
        [Key, Column("account_id", Order = 1)]
        public int AccountId { get; set; }
```

```
    }
}
```

UseModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("uses")]
    public class UseModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

UsesImmersionClassificationModel.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace ImmersiveLearningAPI.DataModels
{
    [Table("uses_immersion_classification")]
    public class UsesImmersionClassificationModel
    {
        [Key]
        public int Id { get; set; }
        [Column("uses_id")]
        public int UseId { get; set; }
        [Column("system_immersion")]
        public float SystemImmersion { get; set; }
        [Column("narrative_immersion")]
        public float NarrativeImmersion { get; set; }
        [Column("challenge_immersion")]
        public float ChallengeImmersion { get; set; }
        [Column("rgb_color")]
        public string RGBColor { get; set; }
    }
}
```

ImmersiveLearningAPI.Data

IServiceContext.cs

```
using ImmersiveLearningAPI.DataModels;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Text;

namespace ImmersiveLearningAPI.Data
{
    public interface IServiceContext
    {
        //Base Entities
        DbSet<AccountModel> Accounts { get; set; }
        DbSet<AuthorModel> Authors { get; set; }
        DbSet<InstitutionModel> Institutions { get; set; }
        DbSet<PaperModel> Papers { get; set; }
        DbSet<UseModel> Uses { get; set; }
        DbSet<UsesImmersionClassificationModel> UsesImmersionClassification { get; set; }

        //Relatioships
        DbSet<AuthorInstitutionModel> AuthorsInstitutions { get; set; }
        DbSet<AuthorPaperModel> AuthorsPapers { get; set; }
        DbSet<UseAccountModel> UsesAccounts { get; set; }
        DbSet<PaperAccountModel> PapersAccounts { get; set; }
    }
}
```

ServiceContext.cs

```
using ImmersiveLearningAPI.DataModels;
using Microsoft.EntityFrameworkCore;
using System;

namespace ImmersiveLearningAPI.Data
{
    public class ServiceContext : DbContext, IServiceContext
    {
        public ServiceContext(DbContextOptions<ServiceContext> options)
            : base(options)
        {

        }

        //Entities
        public DbSet<AccountModel> Accounts { get; set; }
        public DbSet<AuthorModel> Authors { get; set; }
        public DbSet<InstitutionModel> Institutions { get; set; }
        public DbSet<PaperModel> Papers { get; set; }
        public DbSet<UseModel> Uses { get; set; }

        //Relationships
```

```
        public DbSet<AuthorInstitutionModel> AuthorsInstitutions { get; set; }
        public DbSet<AuthorPaperModel> AuthorsPapers { get; set; }
        public DbSet<UseAccountModel> UsesAccounts { get; set; }
        public DbSet<UsesImmersionClassificationModel> UsesImmersionClassification { get; set; }
        public DbSet<PaperAccountModel> PapersAccounts { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<AuthorInstitutionModel>()
                .HasKey(o => new { o.AuthorId, o.InstitutionId });
            modelBuilder.Entity<AuthorPaperModel>()
                .HasKey(o => new { o.AuthorId, o.PaperId });
            modelBuilder.Entity<UseAccountModel>()
                .HasKey(o => new { o.UseId, o.AccountId });
            modelBuilder.Entity<PaperAccountModel>()
                .HasKey(o => new { o.PaperId, o.AccountId });
        }
    }
}
```

ImmersiveLearningAPI.Contracts

Account.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class Account
    {
        public int Id { get; set; }
        public string Text { get; set; }
        public int SurveyId { get; set; }
    }
}
```

Author.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class Author
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

AuthorDetails.cs

```
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Contracts
{
    public class AuthorDetails : Author
    {
```

```
        public List<Institution> Institutions { get; set; }
    }
}
```

AuthorInstitution.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class AuthorInstitution
    {
        public int AuthorId { get; set; }
        public int InstitutionId { get; set; }
    }
}
```

AuthorPaper.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class AuthorPaper
    {
        public int AuthorId { get; set; }
        public int PaperId { get; set; }
    }
}
```

Institution.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class Institution
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string CountryCode { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

Paper.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class Paper
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int PubYear { get; set; }
        public string Date { get; set; }
        public string OpenAlexId { get; set; }
    }
}
```

PaperAccount.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class PaperAccount
    {
        public int PaperId { get; set; }
        public int AccountId { get; set; }
    }
}
```

PaperDetails.cs

```
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Contracts
{
    public class PaperDetails : Paper
    {
        public List<AuthorDetails> Authors { get; set; }
    }
}
```

TokenRequest.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class TokenRequest
    {
        public string clientId { get; set; }
        public string clientSecret { get; set; }
    }
}
```

TokenResponse.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class TokenResponse
    {
        public string Token { get; set; }
    }
}
```

Use.cs

```
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Contracts
{
    public class Use
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public float SystemImmersion { get; set; }
        public float NarrativeImmersion { get; set; }
        public float ChallengeImmersion { get; set; }
```

```
        public string RGBColor { get; set; }
        public List<PaperDetails> Papers { get; set; }
    }
}
```

UseAccount.cs

```
namespace ImmersiveLearningAPI.Contracts
{
    public class UseAccount
    {
        public int UseId { get; set; }
        public int AccountId { get; set; }
    }
}
```

ImmersiveLearningAPI.Domain

IServiceDomain.cs

```
using ImmersiveLearningAPI.Contracts;
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Domain.interfaces
{
    public interface IServiceDomain
    {
        /// <summary>
        /// Finds and returns all accounts from database
        /// </summary>
        /// <returns>Collection of Accounts</returns>
        public IEnumerable<Account> GetAllAccounts();
        /// <summary>
        /// Finds and returns all Papers from database
        /// </summary>
        /// <returns>Collection of Papers</returns>
        IEnumerable<Paper> GetAllPapers();
        /// <summary>
        /// Finds and returns all relationships between Papers and Accounts
        /// </summary>
        /// <returns>Collection of relationships between Papers and Accounts</returns>
        IEnumerable<PaperAccount> GetAllPapersAccounts();
        /// <summary>
        /// Gets paper details
        /// </summary>
        /// <param name="paperId">Unique identifier for Paper</param>
        /// <returns>Paper details</returns>
        PaperDetails GetPaperDetailsById(int paperId);
        /// <summary>
        /// Finds and returns all Institutions
        /// </summary>
        /// <returns>Collection of Institutions</returns>
        IEnumerable<Institution> GetAllInstitutions();
        /// <summary>
```

```
        /// Finds and returns all Authors
        /// </summary>
        /// <returns>Collection of Authors</returns>
        IEnumerable<Author> GetAllAuthors();
        /// <summary>
        /// Finds and returns all relationships between Authors and Papers
        /// </summary>
        /// <returns>Collection of relationships between Authors and Papers</returns>
        IEnumerable<AuthorPaper> GetAllAuthorsPapers();
        /// <summary>
        /// Finds and returns all relationships between Authors and Institutions
        /// </summary>
        /// <returns>Collection of relationships between Authors and Institutions</returns>
        IEnumerable<AuthorInstitution> GetAllAuthorsInstitutions();
        /// <summary>
        /// Gets Author details
        /// </summary>
        /// <param name="authorId">Unique identifier for Author</param>
        /// <returns>Author details</returns>
        AuthorDetails GetAuthorDetailsById(int authorId);
        /// <summary>
        /// This method finds all returns all Theme of Uses, according
        /// to defined Params.
        /// </summary>
        /// <param name="year">
        /// This method returns all uses with publications within the specified
        /// year. When 0, it returns all uses with publication, regardless of the
        /// publication Year..
        /// </param>
        /// <param name="includeAllUpToYear">
        /// When true, this method considers all publications up until the
        /// specified Year (when a year was specified).
        /// Otherwise, considers only publication within the specified
        /// Year (when a yeas was specified)
        /// </param>
        /// <returns>
        /// Uses, including details such as Papers) according
        /// to specified Params.
        /// </returns>
        IEnumerable<Use> GetAllUses(int year, bool includeAllUpToYear);
        /// <summary>
        /// Finds and returns all relationships between Themes of Use and Accounts
        /// </summary>
        /// <returns>Collection of relationships between Themes of Use and Accounts</returns>
        IEnumerable<UseAccount> GetAllUsesAccounts();
        /// <summary>
        /// Finds and return all Years for Themes of Use
        /// </summary>
        /// <returns>Collection of years</returns>
        IEnumerable<int> GetUsesYears();
    }
}
```

ServiceDomain.cs

```
using AutoMapper;
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Data;
using ImmersiveLearningAPI.DataModels;
using ImmersiveLearningAPI.Domain.interfaces;
using System;
using System.Collections.Generic;
using System.Linq;

namespace ImmersiveLearningAPI.Domain
{
    public class ServiceDomain : IServiceDomain
    {
        private readonly IServiceContext _serviceContext;
        private readonly IMapper _mapper;
        public ServiceDomain(IServiceContext serviceContext, IMapper mapper)
        {
            _serviceContext = serviceContext;
            _mapper = mapper;
        }

        public IEnumerable<Account> GetAllAccounts()
        {
            try
            {
                return _mapper.Map<List<Account>>(_serviceContext.Accounts.ToList());
            }
            catch (Exception ex)
            {

                throw ex;
            }
        }

        public IEnumerable<Paper> GetAllPapers()
        {
            try
            {
                return _mapper.Map<List<Paper>>(_serviceContext.Papers.ToList());
            }
            catch (Exception ex)
            {

                throw ex;
            }
        }

        public IEnumerable<PaperAccount> GetAllPapersAccounts()
        {
            try
            {
                return _mapper.Map<List<PaperAccount>>(_serviceContext.PapersAccounts.ToList());
```

```
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }

    public PaperDetails GetPaperDetailsById(int paperId)
    {
        try
        {
            PaperDetails paper = _mapper.Map<PaperDetails>(_serviceContext.Papers.Where(p =>
p.Id == paperId).FirstOrDefault());

            if (paper != null)
            {
                paper.Authors = new List<AuthorDetails>();

                var authorsPapers = _serviceContext.AuthorsPapers.Where(p => p.PaperId ==
paperId);
                if (authorsPapers != null)
                {
                    foreach (var val in authorsPapers)
                    {
                        paper.Authors.Add(GetAuthorDetailsById(val.AuthorId));
                    }
                }
            }

            return paper;
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public IEnumerable<Institution> GetAllInstitutions()
    {
        try
        {
            return _mapper.Map<List<Institution>>(_serviceContext.Institutions.ToList());
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }

    public IEnumerable<Author> GetAllAuthors()
    {
        try
```

```
        {
            return _mapper.Map<List<Author>>(_serviceContext.Authors.ToList());
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public IEnumerable<AuthorPaper> GetAllAuthorsPapers()
    {
        try
        {
            return _mapper.Map<List<AuthorPaper>>(_serviceContext.AuthorsPapers.ToList());
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public IEnumerable<AuthorInstitution> GetAllAuthorsInstitutions()
    {
        try
        {
            return
_mapper.Map<List<AuthorInstitution>>(_serviceContext.AuthorsInstitutions.ToList());
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }


    public AuthorDetails GetAuthorDetailsById(int authorId)
    {
        try
        {
            AuthorDetails author = _mapper.Map<AuthorDetails>(_serviceContext.Authors.Where(a
=> a.Id == authorId).FirstOrDefault());

            if(author != null)
            {
                author.Institutions = new List<Institution>();

                var authorInstitutions = _serviceContext.AuthorsInstitutions.Where(a => a.AuthorId ==
authorId);
                if(authorInstitutions != null)
                {
                    foreach(var val in authorInstitutions)
                    {
```

```
author.Institutions.Add(_mapper.Map<Institution>(_serviceContext.Institutions.Where(i => i.Id ==
val.InstitutionId).FirstOrDefault()));
                }
            }
        }

        return author;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}

public IEnumerable<Use> GetAllUses(int year, bool includeAllUpToYear)
{
    try
    {
        List<Use> uses = null;
        if (year == 0)
        {
            //Year = 0 means all themes of use should be consider, regardless
            //of the publication year
            uses = _mapper.Map<List<Use>>(_serviceContext.Uses.ToList());
        }
        else
        {
            //Year != 0 means we should restrict to the Themes of Use with
            //publications on/up to the specified year.
            IEnumerable<UseModel> usesModel;

            usesModel = from u in _serviceContext.Uses
                        join ua in _serviceContext.UsesAccounts on u.Id equals ua.UseId
                        join pa in _serviceContext.PapersAccounts on ua.AccountId equals
pa.AccountId
                        join p in _serviceContext.Papers on pa.PaperId equals p.Id
                        where (includeAllUpToYear && p.PubYear <= year) || (!includeAllUpToYear &&
p.PubYear == year)
                        select u;

            uses = _mapper.Map<List<Use>>(usesModel.Distinct());

            if(uses != null)
            {
                //lets get the details for each Theme of Use found
                uses.ForEach(u => {
                    IEnumerable<PaperModel> papersModel;

                    papersModel = from p in _serviceContext.Papers
                                  join pa in _serviceContext.PapersAccounts on p.Id equals pa.PaperId
                                  join ua in _serviceContext.UsesAccounts on pa.AccountId equals
ua.AccountId
                                  where
```

```
                                (
                                    (includeAllUpToYear && p.PubYear <= year) ||
                                    (!includeAllUpToYear && p.PubYear == year)
                                ) && ua.UseId == u.Id
                                select p;

                    if (papersModel != null)
                    {
                        u.Papers = new List<PaperDetails>();
                        papersModel.ToList().ForEach(p =>
                        {
                            //this will cascade getting the paper details, authors and institutions
                            u.Papers.Add(GetPaperDetailsById(p.Id));
                        });
                    }
                });
            }
        }

        if (uses != null)
        {
            uses.ForEach(u =>
            {
                //for each theme of use, we get its immersion classification
                var usesClassification = _serviceContext.UsesImmersionClassification
                                        .Where(c => c.UseId == u.Id)
                                        .FirstOrDefault();

                if (usesClassification != null)
                {
                    u.SystemImmersion = usesClassification.SystemImmersion;
                    u.NarrativeImmersion = usesClassification.NarrativeImmersion;
                    u.ChallengeImmersion = usesClassification.ChallengeImmersion;
                    u.RGBColor = usesClassification.RGBColor;
                }
            });
        }
        return uses;
    }
    catch (Exception ex)
    {

        throw ex;
    }
}

public IEnumerable<UseAccount> GetAllUsesAccounts()
{
    try
    {
        var usesAccounts =
_mapper.Map<List<UseAccount>>(_serviceContext.UsesAccounts.ToList());
        return usesAccounts;
    }
```

```
            catch (Exception ex)
            {

                throw ex;
            }
        }

        public IEnumerable<int> GetUsesYears()
        {
            var years = new List<int>();

            try
            {
                years = (from u in _serviceContext.Uses
                         join ua in _serviceContext.UsesAccounts on u.Id equals ua.UseId
                         join pa in _serviceContext.PapersAccounts on ua.AccountId equals pa.AccountId
                         join p in _serviceContext.Papers on pa.PaperId equals p.Id
                         where p.PubYear > 0
                         select p.PubYear).Distinct().ToList();
                years.Sort();
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return years;
        }
    }
}
```

ImmersiveLearningAPI

appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "ImmersiveLearning.db"
  },
  "SecurityKey": "NÃO INCLUÍDA NA DISPONIBILIZAÇÃO PÚBLICA",
  "clientId": "NÃO INCLUÍDA NA DISPONIBILIZAÇÃO PÚBLICA ",
  "clientSecret": "NÃO INCLUÍDA NA DISPONIBILIZAÇÃO PÚBLICA",
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"

}
```

AutoMapperServiceProfile.cs

```
using AutoMapper;
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.DataModels;

namespace ImmersiveLearningAPI
{
    public class AutoMapperServiceProfile : Profile
    {
        public AutoMapperServiceProfile()
        {
            CreateMap<AccountModel, Account>();
            CreateMap<PaperModel, Paper>();
            CreateMap<PaperModel, PaperDetails>();
            CreateMap<InstitutionModel, Institution>();
            CreateMap<AuthorModel, Author>();
            CreateMap<AuthorPaperModel, AuthorPaper>();
            CreateMap<AuthorInstitutionModel, AuthorInstitution>();
            CreateMap<AuthorModel, AuthorDetails>();
            CreateMap<UseModel, Use>();
            CreateMap<UseAccountModel, UseAccount>();
            CreateMap<PaperAccountModel, PaperAccount>();
        }
    }
}
```

Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;


namespace ImmersiveLearningAPI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }

}
```

Startup.cs

```
using ImmersiveLearningAPI.Data;
using ImmersiveLearningAPI.Domain;
```

```
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.IO;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using Microsoft.OpenApi.Models;
using System.Reflection;

namespace ImmersiveLearningAPI
{
    public class Startup
    {
        private IWebHostEnvironment _appHost;

        public Startup(IConfiguration configuration, IWebHostEnvironment appHost)
        {
            Configuration = configuration;
            _appHost = appHost;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

            //Setting up JSON Token validation
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {
                    options.TokenValidationParameters = new TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidateAudience = true,
                        ValidateLifetime = true,
                        RequireExpirationTime = true,
                        ValidateIssuerSigningKey = true,
                        ValidIssuer = "UAb",
                        ValidAudience = "UAb",
                        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["SecurityKey"])),
                        ClockSkew = TimeSpan.FromSeconds(1),
                    };
                }
            );
```

```
        //Setting up dependency invection
        services.AddDbContext<ServiceContext>(x =>
            x.UseSqlite($"Data
Source={_appHost.ContentRootPath}/{Configuration.GetConnectionString("DefaultConnection")}")
        );
        services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
        services.AddScoped<IServiceContext, ServiceContext>();
        services.AddScoped<IServiceDomain, ServiceDomain>();
        services.AddApplicationInsightsTelemetry();

        //Making sure the browser allows the web app talking to this service
        services.AddCors(options =>
        {
            options.AddDefaultPolicy(
                builder =>
                {
                    builder.AllowAnyOrigin()
                           .AllowAnyHeader()
                           .AllowAnyMethod();
                });
        });

        services.AddControllers()
            .AddJsonOptions(options => options.JsonSerializerOptions.IgnoreNullValues = true);

        services.AddSwaggerGen(options =>
        {
            options.SwaggerDoc("v1", new OpenApiInfo
            {
                Version = "v1",
                Title = "Immersive Learning API",
                Description = "This WebAPI allows access to data results from published paper
\"Finding the Gaps about Uses of Immersive Learning Environments: A Survey of Surveys\" "
            });

            options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
            {
                Description = @"JWT Authorization header using the Bearer scheme. \r\n\r\n
                  Enter 'Bearer' [space] and then your token in the text input below.
                  \r\n\r\nExample: 'Bearer 12345abcdef'",
                Name = "Authorization",
                In = ParameterLocation.Header,
                Type = SecuritySchemeType.ApiKey,
                Scheme = "Bearer"
            });

            options.AddSecurityRequirement(new OpenApiSecurityRequirement()
              {
                {
                  new OpenApiSecurityScheme
                  {
                    Reference = new OpenApiReference
                      {
                        Type = ReferenceType.SecurityScheme,
```

```
                        Id = "Bearer"
                    },
                    Scheme = "oauth2",
                    Name = "Bearer",
                    In = ParameterLocation.Header,

                },
                new List<string>()
            }
        });

        var xmlFilename = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
        options.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, xmlFilename));
    });
}

// This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseSwagger();

    app.UseSwaggerUI();

    app.UseAuthentication();

    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseCors();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
}
}
```

Controllers/AccountsController.cs

```
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
```

```
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Controllers
{
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    [ApiController]
    public class AccountsController : ControllerBase
    {
        private static IServiceDomain _serviceDomain;

        public AccountsController(IServiceDomain serviceDomain)
        {
            _serviceDomain = serviceDomain;
        }

        /// <summary>
        /// Gets all Accounts of Use
        /// </summary>
        /// <returns>List of all Accounts of Use</returns>
        [HttpGet()]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Account>))]
        public IActionResult GetAll()
        {
            try
            {
                return Ok(_serviceDomain.GetAllAccounts());
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }
    }
}
```

Controllers/AuthorsController.cs

```
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Controllers
{
    /// <summary>
    /// Authors of publications
    /// </summary>

    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    [ApiController]
```

```
public class AuthorsController : Controller
{
    private static IServiceDomain _serviceDomain;

    public AuthorsController(IServiceDomain serviceDomain)
    {
        _serviceDomain = serviceDomain;
    }

    /// <summary>
    /// Gets all authors of publications
    /// </summary>
    /// <returns>List of all Authors of publications</returns>
    [HttpGet()]
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Author>))]
    public IActionResult GetAll()
    {
        try
        {
            return Ok(_serviceDomain.GetAllAuthors());
        }
        catch (System.Exception ex)
        {
            return Problem(ex.Message);
        }
    }

    /// <summary>
    /// Gets all relationships betweens Authors and Papers
    /// </summary>
    /// <returns>List of all relationships between Authors and Papers</returns>
    [HttpGet("AuthorsPapers")]
    [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IEnumerable<AuthorPaper>))]
    public IActionResult GetAllAuthorsPapers()
    {
        try
        {
            return Ok(_serviceDomain.GetAllAuthorsPapers());
        }
        catch (System.Exception ex)
        {
            return Problem(ex.Message);
        }
    }

    /// <summary>
    /// Gets all relationships betweens Authors and Institutions
    /// </summary>
    /// <returns>List of all relationships between Authors and Institutions</returns>
    [HttpGet("AuthorsInstitutions")]
    [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IEnumerable<AuthorInstitution>))]
    public IActionResult GetAllAuthorsInstitutions()
```

```
        {
            try
            {
                return Ok(_serviceDomain.GetAllAuthorsInstitutions());
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }

        /// <summary>
        /// Gets a specific Author identified by the authorId
        /// </summary>
        /// <param name="authorId">Author's unique identifier</param>
        /// <returns>Author details</returns>
        [HttpGet("{authorId}")]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(AuthorDetails))]
        [ProducesResponseType(StatusCodes.Status404NotFound)]
        public IActionResult GetById([FromRoute] int authorId)
        {
            try
            {
                var response = _serviceDomain.GetAuthorDetailsById(authorId);

                if (response == null)
                    NotFound(authorId);

                return Ok(response);
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }
    }
}
```

Controllers/InstitutionsController.cs

```
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Controllers
{
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    [ApiController]
    public class InstitutionsController : ControllerBase
    {
```

```
        private static IServiceDomain _serviceDomain;

        public InstitutionsController(IServiceDomain serviceDomain)
        {
            _serviceDomain = serviceDomain;
        }

        /// <summary>
        /// Gets all institutions of publications
        /// </summary>
        /// <returns>List of all Institutions of publications</returns>
        [HttpGet()]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IEnumerable<Institution>))]
        public IActionResult GetAll()
        {
            try
            {
                return Ok(_serviceDomain.GetAllInstitutions());
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }
    }
}
```

Controllers/PapersController.cs

```
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Controllers
{
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    [ApiController]
    public class PapersController : ControllerBase
    {
        private static IServiceDomain _serviceDomain;

        public PapersController(IServiceDomain serviceDomain)
        {
            _serviceDomain = serviceDomain;
        }

        /// <summary>
        /// Gets all papers (publications)
        /// </summary>
```

```
        /// <returns>List of all Papers (publications)</returns>
        [HttpGet()]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Paper>))]
        public IActionResult GetAll()
        {
            try
            {
                return Ok(_serviceDomain.GetAllPapers());
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }

        /// <summary>
        /// Gets a specific papers
        /// </summary>
        /// <param name="paperId">Papers' unique identifier</param>
        /// <returns>Paper details</returns>
        [HttpGet("{paperId}")]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(PaperDetails))]
        [ProducesResponseType(StatusCodes.Status404NotFound)]
        public IActionResult GetById([FromRoute] int paperId)
        {
            try
            {
                var response = _serviceDomain.GetPaperDetailsById(paperId);

                if (response == null)
                    NotFound(paperId);

                return Ok(response);
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }

        /// <summary>
        /// Gets all relationships betweens Papers and Accounts
        /// </summary>
        /// <returns>List of all relationships between Papers and Accounts</returns>
        [HttpGet("PapersAccounts")]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IEnumerable<PaperAccount>))]
        public IActionResult GetAllPapersAccounts()
        {
            try
            {
                return Ok(_serviceDomain.GetAllPapersAccounts());
            }
            catch (System.Exception ex)
```

```
        {
            return Problem(ex.Message);
        }
    }
  }
}
```

Controllers/TokenController.cs

```
using ImmersiveLearningAPI.Contracts;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace ImmersiveLearningAPI.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class TokenController : ControllerBase
  {
    private readonly IConfiguration _configuration;

    public TokenController(IConfiguration configuration)
    {
      _configuration = configuration;
    }

    /// <summary>
    /// Generates an authentication Bearer token
    /// </summary>
    /// <param name="tokenRequest">Client credentials</param>
    /// <returns>Bearer token</returns>
    [AllowAnonymous]
    [HttpPost]
    [Produces("application/json", Type = typeof(string))]
    public IActionResult GetToken([FromBody] TokenRequest tokenRequest)
    {
      System.Diagnostics.Trace.TraceInformation("information");
      System.Diagnostics.Trace.TraceWarning("Warning");
      System.Diagnostics.Trace.TraceError("Error");

      var response = new TokenResponse();

      if (!ModelState.IsValid)
        return BadRequest(ModelState);

      if (tokenRequest == null)
        return BadRequest("Could not verify username and password");

      string clientId = _configuration["clientId"];
```

```
            string clientSecret = _configuration["clientSecret"];

            if(!(clientId.Equals(tokenRequest.clientId) &&
clientSecret.Equals(tokenRequest.clientSecret)))
            {
                return Unauthorized();
            }

            var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["SecurityKey"]));
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var claims = new[]
            {
                new Claim("system", "api")
            };

            var token = new JwtSecurityToken(
                issuer: "UAb",
                audience: "UAb",
                claims: claims,
                expires: System.DateTime.UtcNow.AddMinutes(60),
                signingCredentials: creds
            );

            response.Token = new JwtSecurityTokenHandler().WriteToken(token);

            return Ok(response);
        }
    }
}
```

Controllers/UsesController.cs

```
using ImmersiveLearningAPI.Contracts;
using ImmersiveLearningAPI.Domain.interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace ImmersiveLearningAPI.Controllers
{
    [Authorize(AuthenticationSchemes = "Bearer")]
    [Route("api/[controller]")]
    [ApiController]
    public class UsesController : Controller
    {
        private static IServiceDomain _serviceDomain;

        public UsesController(IServiceDomain serviceDomain)
        {
            _serviceDomain = serviceDomain;
        }
```

```
        /// <summary>
        /// Gets Themes of Uses
        /// </summary>
        /// <param name="year">
        /// Restricts Themes of Use to the specified year.
        /// 0 values means no retriction.
        /// </param>
        /// <param name="includeAllUpToYear">
        /// Appliacable only when param year is not zero.
        /// When true, considers all papers with publication year up to the specified Year.
        /// When false, considers all papers with published on the specified Year only.
        /// </param>
        /// <returns></returns>
        [HttpGet()]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<Use>))]
        public IActionResult Get(int year, bool includeAllUpToYear)
        {
            try
            {
                return Ok(_serviceDomain.GetAllUses(year, includeAllUpToYear));
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }

        /// <summary>
        /// Gets all relationships betweens Themes of Use and Accounts
        /// </summary>
        /// <returns>List of all relationships between Themes of Use and Accounts</returns>
        [HttpGet("UsesAccounts")]
        [ProducesResponseType(StatusCodes.Status200OK, Type =
typeof(IEnumerable<UseAccount>))]
        public IActionResult GetAllUsesAccounts()
        {
            try
            {
                return Ok(_serviceDomain.GetAllUsesAccounts());
            }
            catch (System.Exception ex)
            {
                return Problem(ex.Message);
            }
        }

        /// <summary>
        /// Gets all Uses's years
        /// </summary>
        /// <returns>List of years</returns>
        [HttpGet("Years")]
        [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<int>))]
        public IActionResult GetUsesYears()
```

```
    {
        try
        {
            return Ok(_serviceDomain.GetUsesYears());
        }
        catch (System.Exception ex)
        {
            return Problem(ex.Message);
        }
    }
  }
}
```

Anexo IV – Template html para detalhes dos Temas de Uso

Neste anexo, pode-se encontrar o código implementado para desenvolver o template de detalhes de Temas de Uso gerados.

UseDetails.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Use Details</title>
        <style>
            .app-title {
                text-align: center;
                font-weight: normal;
                font-size: 2.6rem;
            }

            body {
                font-family: sans-serif;
                background-color: #dedede;
                color: #333;
                padding: 20px 0 28px 0;
                margin: 0;
            }

            .paper {
                max-width: 650px;
                padding: 20px;
                margin: 30px auto;
                background-color: #fff;
                box-shadow: 3px 3px 3px rgba(0, 0, 0, 0.1);
                overflow: hidden;
            }

            .paper-title {
                font-size: 16px;
                font-weight: normal;
                margin: 0 0 1rem 0;
            }

            .counts {
                max-width: 650px;
```

```
                padding: 20px;
                margin: 30px auto;
            }

            a:link {
                text-decoration: none;
            }

            a:visited {
                text-decoration: none;
            }

            a:hover {
                text-decoration: underline;
            }

            a:active {
                text-decoration: underline;
            }
        </style>
        <script
src='https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js'>
</script>
          <script>
            //Main function called when loading the html page
            //Builds a block of html based on a JSON file
            $(document).ready(function(){
                const id = getUrlParameter('id');
                const useDetails =
JSON.parse(`${sessionStorage.getItem(id)}`)
                $('#title').html(`${useDetails.StartYear ===
useDetails.FinalYear ? 'Year: ' + useDetails.StartYear : 'Years: ' +
useDetails.StartYear + '-' + useDetails.FinalYear }<br>Use:
${useDetails.Name}`);
                $('#summary').html(`Total Papers:
${useDetails.PapersCount}<br>Total Authors:
${useDetails.AuthorsCount}<br>Total Institutions:
${useDetails.InstitutionsCount}`);
                $('#details').html(`${useDetails.Papers.map(paperTempla
te).join('')}`);
            });

            function getInstitutions(institutions) {
                return `${institutions.map(inst => ` (<a
target='_blank' href='${inst.OpenAlexId}'>${inst.Name}</a>)
`).join('')}`;
            }
```

```
            //This function generates a block of html for authors in a
JSON file
            function getAuthors(authors) {
                return `
                <h4>Authors:</h4>
                <ul class='foods-list'>
                    ${authors.map(author => `
                        <li>
                            <a target='_blank' href =
'${author.OpenAlexId}' >${author.Name}</a>
                        ${author.Institutions?
getInstitutions(author.Institutions) : ''}
                        </li>`).join('')}
                </ul>`;
            }

            //This function generates a block of html for papers in a
JSON file
            function paperTemplate(paper) {
                return `
                    <div class='paper'>
                        <h2 class='paper-title'>Paper: ${paper.Title}
</h2>
                        <h2 class='paper-title'>OpenAlex: <a
target='_blank' href='${paper.OpenAlexId}'>${paper.OpenAlexId?
paper.OpenAlexId : ''}</a> </h2>
                        ${paper.Authors ? getAuthors(paper.Authors) :
''}
                    </div>
                `;
            }

            //This function finds and retrieves a param value from URL
            var getUrlParameter = function getUrlParameter(sParam) {
              var sPageURL = window.location.search.substring(1),
                  sURLVariables = sPageURL.split('&'),
                  sParameterName,
                  i;

              for (i = 0; i < sURLVariables.length; i++) {
                  sParameterName = sURLVariables[i].split('=');

                  if (sParameterName[0] === sParam) {
                      return sParameterName[1] === undefined ? true :
decodeURIComponent(sParameterName[1]);
                  }
```

```
            }
            return false;
          };
        </script>
    </head>
    <body>
        <h1 class='app-title' id="title"></h1>
        <div class='counts' id="summary"></div>
        <div id='details'></div>
    </body>
</html>
```