TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR
POLITÉCNICO SETÚBAL

NICOLE ALEXANDRA MARTINS VIEIRA

CREATION OF DYNAMIC VIRTUAL TOURS IN MULTIMEDIA SPACES

Relatório de Dissertação do Mestrado em Engenharia Informática

ORIENTADOR
Doutor, Fausto José da Silva Valentim Mourato

CO-ORIENTADOR
Doutor, Cláudio Miguel Garcia Loureiro dos Santos Sapateiro

NICOLE
ALEXANDRA
MARTINS VIEIRA

CREATION OF DYNAMIC VIRTUAL TOURS IN MULTIMEDIA SPACES

Dezembro 2023

JÚRI

Presidente: Doutor, José António Moinhos Cordeiro, ESTSetúbal/IPS
Orientador: Doutor, Fausto José da Silva Valentim Mourato, ESTSetúbal/IPS
Arguente: Especialista, João Pedro de Abreu Morais, ESTSetúbal/IPS

Dezembro 2023

Acknowledgments

I would like to express my gratitude to everyone who contributed to the completion of this dissertation. Firstly, I want to extend my gratitude to my advisor, Professor Fausto Mourato, and Professor Susana Lucas for granting me the opportunity to work on this project. I want to emphasize my appreciation for my advisor's invaluable guidance and his readiness to address any questions that arose throughout the project. With his assistance, I was able to acquire new knowledge and reach appropriate conclusions for specific issues encountered during the project's development.

I also want to extend my gratitude to Bianca Martins, a speech therapist, and Mariana Franco, a risk analyst, for their assistance in overseeing this dissertation. They not only provided valuable support but also accompanied me on this journey, offering encouragement whenever needed.

Finally, but not least, I would like to express my deep appreciation to my family for their unwavering support in all the decisions I made and their constant availability throughout my academic journey. They were an essential source of strength during this period, and I am truly thankful for their endorsement.

Abstract

With the advancement of technology, it has become possible to carry out virtual tours to physical spaces, without being there in person. With the pandemic, this type of technology proved to be of great importance for places like museums due to people only leaving home to go to places of extreme importance, such as the hospital or to buy essential goods, but even that began to be less thanks to the existence of applications like Uber that deliver food at home.

In case of physical tours, with the innovation of technology, it is possible to use augmented reality to make the visits more interesting and thus enrich the experience. Although virtual tours are useful, there are still certain difficulties in their implementation due to the inclusion of dynamic elements, such as interaction with people and the scenery. Throughout this paper, these same difficulties will be addressed and explained how they were resolved.

Build2050 is a European project coordinated by IPS, which aims to build smart and sustainable buildings. This dissertation consists of the implementation of the technologies mentioned above, to provide different ways of visiting buildings and be more impactful for the visitor.

Keywords: Augmented reality, virtual reality, Vuforia, 360tour, virtual tour

Index

List of Figures

List of Tables

List of Acronyms

AR	Augmented Reality
HMD	Head-Mounted Display
IPS	Setúbal Polytechnic Institute
MAR	Mobile Augmented Reality
PADM	Protective Action Decision Model
RAM	Random Access Memory
SDK	Software Development Kit
SSMS	SQL Server Management Studio
TAM	Technology Acceptance Model
UWP	Universal Windows Platform
VR	Virtual reality

Chapter 1
Introduction

The following document constitutes a master's thesis conducted at Polytechnic Institute of Setúbal (IPS), specifically within the field of Informatics Engineering, within the scope of the Internship/Project curricular unit.

The primary objective of this project is to develop an application that uses avatars to communicate in real time with the user to create an engaging 360º virtual tour. This chapter serves as an introductory segment that provides a comprehensive overview of the project undertaken within the ambit of the current dissertation. It aims to delve deeper into the motivation that supports the dissertation's foundation, offering a through exploration of the contextual background. Additionally, it will elucidate all the objectives guiding this study. Finally, the structure of this document will be explained to provide a clear guideline for the reader.

1.1. Motivation

During the global pandemic, businesses across various industries faced unprecedented challenges as people became apprehensive about visiting crowded places and chose to stay home to mitigate the risk of contracting the virus. Industries that relied on direct customer engagement, such as museums, theatres, and concert venues, were among the hardest hit by the pandemic. As noted by Osman El-Said and Heba Aziz (2022), these establishments found themselves in a dire situation struggling with a severe decline in customers and revenue. In response, they sought innovative solutions to address the lack of customers and adapt to the evolving circumstances.

Museums initially hesitated to digitalize their collections for online viewing. However, the reality of the pandemic forced them to reconsider their stance and invest in technology to ensure their survival. One of the groundbreaking solutions they embraced was the concept of virtual tours, a digital recreation of either real or fictional environments that enables users to engage with a simulated world in real-time (LI Na & HU Weihua, 2012). These virtual tours presented a transformative way to share culture, history, and art via smartphones or computers, closing the gap between cultural institutions and a global audience.

As the pandemic situation gradually improved, institutions began to recognize the potential of virtual tours. They observed changes in user behaviour and preferences, revealing that virtual tours did not replace physical tours but served as a complementary experience. Virtual tours opened doors to a broader audience, allowing individuals to "visit" cultural sites from the comfort of their own homes. This accessibility was particularly meaningful for those who, due to physical constraints or geographic distances, were unable to participate on physical tours to these locations. Furthermore, virtual tours kindled curiosity and interest in visiting the physical location

among those who had virtually explored them.

The utilisation of virtual tour technology extended beyond the cultural sector and found applications in the real estate industry. Prospective property buyers could virtually tour homes and properties by navigating through 360-degrees images without the need for physical tours. This innovation not only saved time but also enabled a more informed decision-making process.

Considering these developments, the central challenge addressed by this dissertation is to investigate and develop methods to enhance the appeal of both physical and virtual tours. Concerning physical tours, the objective is to use technology to take the visitor experience to another level. This makes use of interactive elements and immersive technologies, such as the AR to make physical tours more engaging and memorable. In the case of virtual tours, the goal is to enhance user interaction and engagement by incorporating dynamic elements and avatars within the virtual environment. The avatars will be open to start a real-time communication with the user, so that all throughout the tour the user can have support when needed. These functionalities aim to create a more captivating and informative experience for users, whether they choose virtual tours or physical tours.

1.2. Context

This dissertation is based in the Build2050 project, an European initiative coordinated by the IPS, aimed at the construction of intelligent and sustainable buildings. The goal of this project is to contribute to Europe's target of achieving climate neutrality by the year 2050. The Build2050 project officially commenced its activities in February 2022 and is expected to span a duration of three years. It is a collaborative effort involving seven prestigious higher education institutions within Europe.

The focus of this dissertation was initiated in November 2022, with a primary objective of developing two innovative tools.

- The first tool makes use of augmented reality technology, and its creation was motivated by the aspiration to enhance on-site presentations and create more engaging interactions with users. This augmented reality application allows for a creative and immersive experience, closing the gap between physical and digital environments. In the initial phase, avatars were incorporated into specific appearances during demonstrations. However, the potential of AR applications extends far beyond this, showing a multitude of possibilities.
- The second application is similar to Google's Street View but designed with a simplified interface, makes use of 360-degrees images for navigation. However, what differs this application from Google's Street View is its gamified approach of user interaction. It features an interactive avatar that engages with the user, facilitating dynamic interactions by providing links and other engaging elements, besides that it is possible to communicate with the support behind the avatar using

the chat box functionality. Because the chat box uses interaction in real time, the conversation functionality runs very smoothly. This gamified aspect adds an element of surprise to the application, making it a unique and engaging way to explore spaces.

The development of these applications within the context of the Build2050 project not only aligns with the broader mission of creating sustainable and intelligent buildings but also represents a significant step towards the use of innovative ways for user engagement.

1.3. Objectives

The main objective of this project revolves around the exploration and in-depth study of the current tools and technologies available for generating virtual tours. To achieve these objectives, the project requires the development of two distinct applications, serving as prototypes. The first application is designed for creating virtual tours, emphasizing the incorporation of dynamic elements, and enabling user interaction with virtual avatars. That interaction can be through the manual interaction with the avatar, where basic information is provided, or through a real-time communication between the user and the support behind the avatar. The second application focuses on physical tours, integrating virtual elements to facilitate user engagement and interaction, such as the use of avatars that will provide basic information. The physical tour application must be accessible through the web browser, ensuring widespread availability and ease of access for users.

1.4. Contributions

The main contribution of this work encompasses the following key aspects:

- **Interactive Spatial Interface:** An application has been developed, allowing users to engage with physical spaces through the use of avatars. These avatars serve as informative guides, enhancing the user's experience by providing real-time assistance and contextual information within the physical environment.
- **Virtual Space Exploration:** Another significant aspect of this project is the creation of an application that enables virtual exploration of physical spaces. Within this virtual space, users are accompanied by avatars that facilitate their exploration by offering insightful information, historical contexts, and even guidance through real-time chat, closing the gap between the physical and virtual worlds.
- **Intuitive chat interface**: To enhance user experience, a user-friendly mechanism has been implemented for allowing user support on their virtual visits. This mechanism is conveniently accessible through standard web browsers.

1.5. Document structure

This dissertation commences with an acknowledgements section, where appropriate gratitude is extended to those who played a crucial role in the realization of the project and dissertation. The dissertation is divided into three chapters, in addition to the bibliography and acknowledgements. The first chapter serves as an introduction to the presented dissertation. It outlines the motivation behind its inception, the context within which it was developed, the objectives aimed to be achieved through the dissertation, its contributions, and finally, the structure of the document.

The second chapter delves into the state of art, it addresses specifically the theoretical framework, where the development methodology that was applied during the project's development can be found. Furthermore, it introduces applications that share certain similarities with the one under development, providing a brief description of each application and its functionalities.

The third chapter explains all the details about the project's development. Initially, it mentions all the used technologies, along with some similar technologies and a comparison with them. Subsequently, the development process is discussed, encompassing both functional and non-functional requirements.

Still within the third chapter, the development of the solution is detailed. This section elucidates how the database containing user profiles and messages between them was developed. It also mentions the development of the server, specifically the routes it contains, which are crucial for interaction between support users and visitors, as well as the website where support users can log in to communicate with visitors.

Finally, the chapter concludes with the presentation of prototypes developed for virtual and physical tours. The virtual tours prototypes cover the creation of avatars and the application's design, the authentication process visitors must undergo to utilize the application, and how the interaction functionality works on desktop and Android.

The dissertation concludes with the final section, the conclusion, which discusses the challenges and issues encountered throughout the development and how they were successfully addressed. This section provides a comprehensive overall project's progression and outcomes. Additionally, there is a bibliography included, which lists all the documents consulted and referenced during the creation of this document. This serves as a valuable resource for readers, offering further insights and references for those engaging with the dissertation.

Chapter 2

State of art

This chapter serves as a comprehensive foundation for the forthcoming work by establishing a theoretical framework. Within this framework, not only the core concepts and principles supporting the project are explained, but also concrete examples of existing applications that bear relevance to the project developed are demonstrated. In doing so, it is expected to provide an in-depth understanding of the theoretical foundations and practical context that will inform and guide the development of the project.

2.1. Theoretical framework

2.1.1. Development methodology

The prototypes under development employ a human-centred design approach, which is a methodology firmly rooted in prioritizing people. This technique places the preferences and needs of individuals considerably at the forefront, shaping the entire product development process around the desires and requirements of the intended audience.

As expressed by Lauren Landry (2020), this approach to design unfolds through four distinct phases. The first phase requires meticulous observation and the acquisition of information. During this phase, the main objective is to accurately define the underlying problem and create strategies to resolve it. This often involves engaging with users through questioning to comprehensively grasp why and how they interact with specific products. For example, understating how they intend to use the product reveals potential customer frustrations.

The second phase is dedicated to idealizing, where the creative process takes centre stage. It is in this phase that brainstorming sessions begin, generating countless ideas related to product development. These ideas serve as the foundation for the subsequent stages.

Following idealizing, the product development phase takes stage. This stage involves a critical examination of the generated ideas to determine whether they can effectively meet the users' needs, if they are technically feasible, and if they align with sustainable practices.

Once these initial three phases are completed, the focus shifts to the product implementation stage. Effective communication becomes important, ensuring that all stakeholders involved in the project are well-informed and aligned for success. Crucial, this phase has no concrete endpoint, as user needs continually evolve as they are addressed. Therefore, ongoing project improvement is essential, adapting the design and development to meet the ever-changing requirements of the users. This iterative approach acknowledges that the dynamic nature of user preferences requires ongoing adjustments to create a truly user-centred product.

In Figure1, it is possible to observe the phases of the mentioned methodology and how they interact with each other.

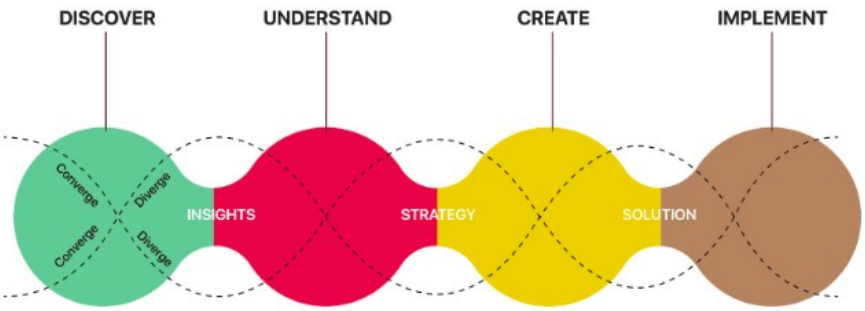

Figure 1 – Phases of the Human Centred Design[1]

2.1.2. Before the implementation of virtual tours

According to the research conducted by Osman El-Said and Heba Aziz (2022), a comprehensive model was created to gauge individuals' intentions to use a computer-base technology. This model, known as the Technology Acceptance Model (TAM), was initially conceived by Fred D.Davis (1985). The TAM was designed to compress the various factors that play a crucial role in influencing a user's decision regarding the adoption of technology. These factors revolve around the perceived ease of use, which assesses the extent to which the technology is user-friendly, free of intricate and overly complex mechanisms, and the perceived usefulness of the technology, which evaluates whether it enhances task performance or not. Over time, as technology evolved and user preferences changed, additional factors emerged that were proven to be influential in determining user technology adoption decision, necessitating revisions and enhancements to the original TAM.

Following this, another notable model, the Protective Action Decision Model (PADM), was developed by Michael K. Lindell and Ronald W.Perry (1992). PADM specifically delved into how individuals react and make decisions in the context of disasters. Given the appearance of virtual tours as a new technology, especially in the environment of the COVID-19 pandemic, it became important to consider both the TAM and PADM frameworks. These models provided valuable insights into understanding user behaviour and decision-making processes, which were particularly relevant in the context of virtual tours.

As the development of this project process, it is important to keep the TAM model in the head. This model will serve as a critical rule to assess whether our prototype is indeed user-friendly, if it fulfils a practical purpose, and whether it enhances both physical and virtual tour experiences. By adhering to the principles of the TAM, we can ensure that our technology aligns with user expectations and preferences, ultimately leading to its successful adoption and utilisation.

[1] https://scopeimpact.fi/core-human-centered-design-process

2.2. Similar applications

2.2.1. ShowMeAround

ShowMeAround is an innovative video platform designed specifically for conducting 360º video tours, targeting a small group of people. This solution has certain limitations commonly found in traditional video conferencing platforms. In video conferencing tools, users often face the obstacle of not being able to fully immerse themselves in the virtual environment, which can hinder their sense of presence within the shared space.

ShowMeAround aims to overcome these limitations by providing an immersive 360º video experience. It accomplishes this by leveraging a technology similar to that of JackInHead, another existing platform. JackInHead employs multiple cameras to capture an entire location and then seamlessly merges these captures to create a comprehensive 360º video. This technology ensures that users can explore and engage with the virtual environment in a more immersive and dynamic manner.

According to a study conducted by Alaeddin Nassani et al. (2021), existing applications for 360º videos typically fall short in two key aspects. First, they are often adapted towards recording videos for later viewing, which may not be conducive to real-time interactions. Second, when real-time interactions are supported, there is usually a limitation on the number of users who can simultaneously access the content often capped at just five participants.

To address these shortcomings, ShowMeAround is built upon the foundation of the open-source Jitsi video conference platform. In addition to Jitsi's capabilities, the application also incorporates the Jitsi API and uses react-native technology to enhance its functionality and user experience. Unlike traditional video conferencing apps that require downloads, ShowMeAround offers accessibility through a simple URL link. This means that users can access the platform from any device with a web browser, eliminating the need for bothersome installations.

One of the relevant features of ShowMeAround is its versatile viewing options. Users can choose between viewing a static image or experiencing the environment in real-time, enhancing the platform's adaptability to various scenarios and preferences.

ShowMeAround is specifically designed to accommodate audiences of 20 or more participants, stimulating a collaborative and interactive environment. Within the platform, two distinct roles exist: users and the host presenter. The host presenter possesses the 360º video feed, while all the other users engaging with the content have access to a conventional 2D video stream. This arrangement allows the host to take control of the virtual conference, including the ability to restrict free navigation and manipulate each user's point of view, ensuring that all participants share the same perspective when needed.

Furthermore, ShowMeAround empowers users by enabling them to mark points of interest within the 360º video feed. When a user marks a point of interest, other participants can observe it and gain insight into the user's perspective, enhancing engagement and facilitating discussions within the virtual tour. This feature allows a more immersive and interactive experience, making ShowMeAround a valuable tool for various educational, business, and recreational purposes.

The functionalities mentioned earlier can be observed in Figure 2, where the participant's video is displayed alongside their respective viewpoints. It becomes evident that these viewpoints are distinguishable through small, coloured dots.

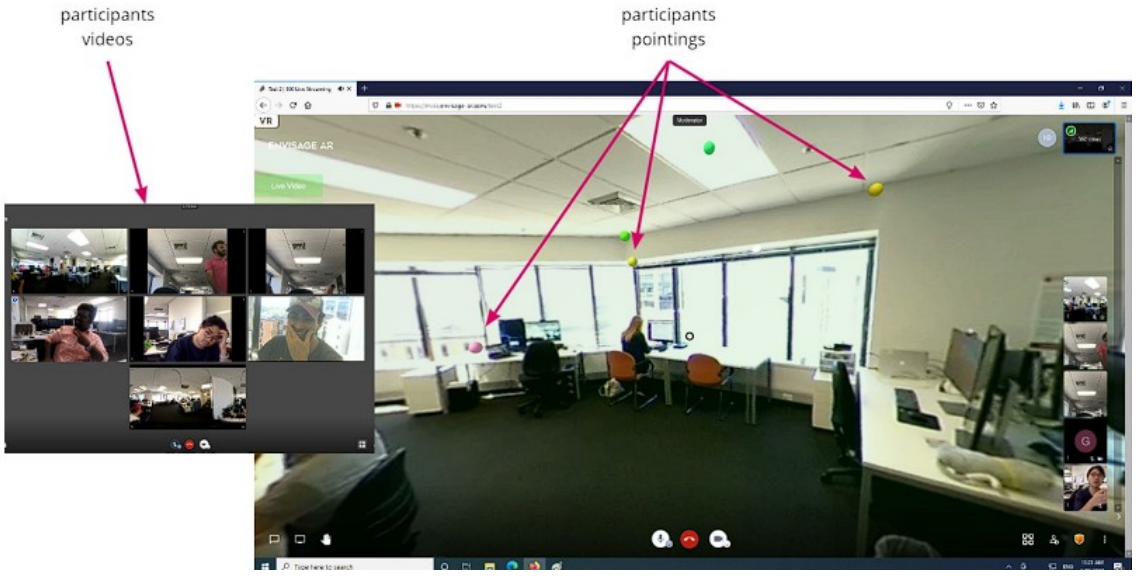

Figure 2 – Example from the app ShowMeAround (Nessani et al., 2021)

2.2.2. Pokemon Go

According to the research conducted by Alvin Yong Chie Yew et al. (2020), the launch of the popular mobile game Pokemon Go (Niantic, 2016) served as a crucial event that showcased the enormous potential of augmented reality in transforming the way users engage with multimedia content.

Pokemon Go, which was a huge gaming phenomenon, utilizes the power of augmented reality to immerse its players in a captivating virtual world. This game was collaboratively developed by Nintendo and the Pokemon company, and upon its release, it accumulated a massive number of players globally. One of the defining features of Pokemon Go is its dependency on the player's GPS, which is important for locating "Pokemoins" scattered throughout the real world, as well as identifying strategic locations referred to as "gyms".

In Pokemon Go, players engage in an all-around gaming experience. They can engage in battles with other players, seize control of gyms through combat, and, when victorious of those combats, the players earn the privilege of placing their own Pokemon in the gym in order to defend them. It is possible to collaborate with players of the same team, allowing them to place their own Pokemons to help in the defence tactic.

An Important aspect of Pokemon Go is its emphasis on physical activity. Players are required to walk to discover wild Pokemons around their surroundings or incubate Pokemon eggs by walking specific distances. Upon encountering a Pokemon, players can use Pokeballs to capture and add it to their Pokédex, building a personal collection of pokemons this way. Engaging in a battle with a captured Pokemon is an option, and players can choose between two modes: battling with or without

augmented reality. Opting for the latter, players can witness the Pokemon superimposed onto their mobile device's camera view, creating an illusion that the Pokemon is physically present in their environment. This innovative use of augmented reality technology enhances the immersion and excitement of the Pokemon Go experience, making it an example of AR's potential in multimedia consumption.

Figure 3 serves as a brief demonstration of how players interact with Pokémon through AR. In this depiction, it is possible to witness the presence of a Pokémon in a backyard setting, while a Pokéball appears in the lower central portion of the mobile screen, which is used for capturing these Pokémons.

Figure 3 – Pokemon Go[2]

2.2.3. Marvel Comics

Marvel developed a mobile application named "Marvel AR", making use of AR technology to raise the comic book reading experience to a whole new level of immersion and information embellishment. The main objective of this app was to provide readers with an enhanced and interactive experience through their favourite Marvel Comics.

To access this augmented experience, users first need to purchase a compatible comic book, which can be easily identified by the presence of the Marvel AR logo. This logo server as a marker for the application to recognize and synchronize with the physical comic book.

Once the user scans the comic's panels using the Marvel AR app, an abundance of additional media content is unlocked and integrated into the reading experience. This extra content can take various forms, such as animated segments that relate to the specific panel being scanned or a

[2] https://pokemongolive.com/pt_br/

commentary from the comic's authors and creators (So-mi Yoon, 2014).

In Figure 4, a depiction of an individual using their mobile phone's camera to view augmented reality content displayed on a poster is observed. In this specific instance, the poster features the iconic Iron Man character, and their avatar emerges from it.

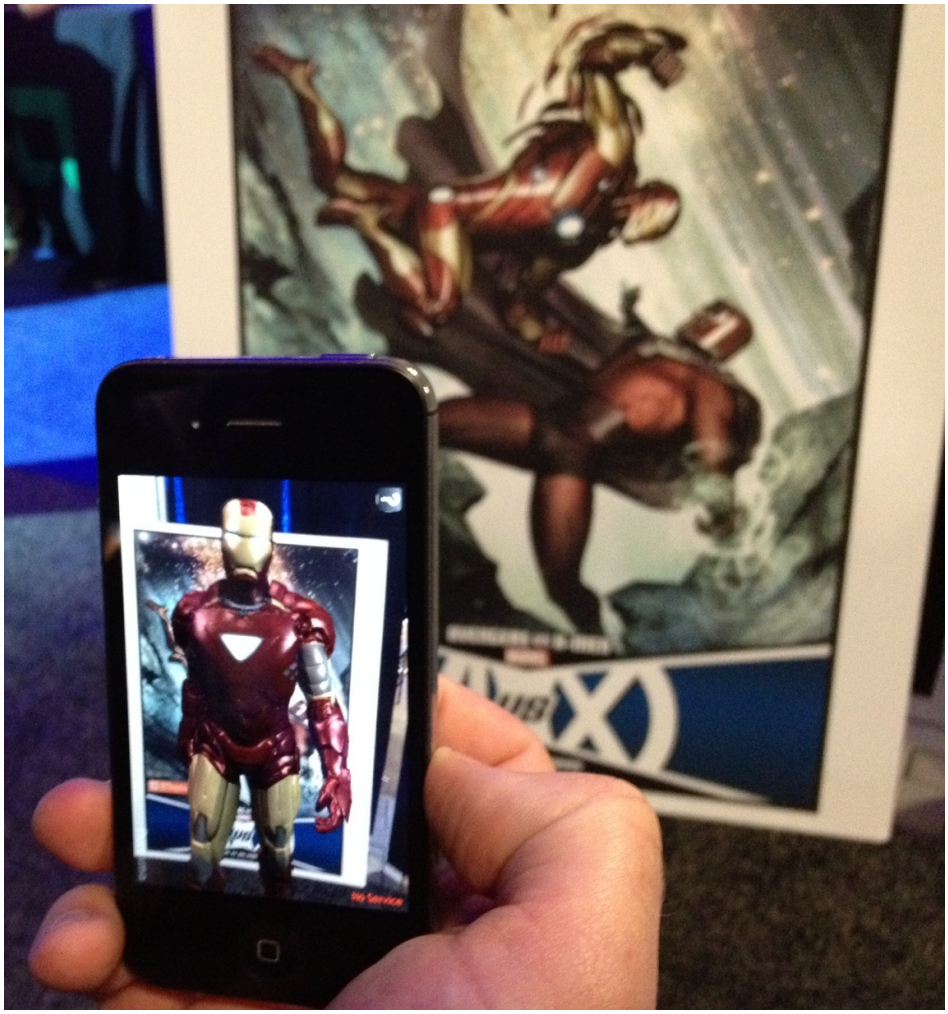

Figure 4 – Marvel Comic AR[3]

2.2.4. Second Life

Second Life (Linden Lab, 2003) is a social game, that had a breakthrough with the norms of social games at the time. Second Life seeks to replicate reality while granting its users unlimited or an unmatched freedom to express themselves. Within this immersive virtual world, the users have the ability of creating personalized avatars, which act as their virtual personas, facilitating communication with others while embodying their chosen appearance and identity. When styling their avatars, users have the liberty to assume any persona they desire, breaking the traditional boundaries of gender and even race, be allowed to create non-humanoid personas. In this dynamic virtual world, it is not uncommon for people to maintain multiple avatars, allowing them to explore a wide range of identities

[3] https://techcrunch.com/2012/03/11/marvel-augmented-reality/

and experiences.

As a social game, Second Life encourages the development of meaningful relationships among its users. Specially for those who may encounter challenges when interaction in the real world, social games like Second Life provide a comforting and supportive environment where users can be themselves, connect and create connections with others.

As can be observed in Figure 5, the ability to establish a relationship with other players serves as a form of advertisement for the possibilities that the game offers.

Figure 5 – Second Life[4]

Avatars is Second Life display the traits and personalities of their respective users, not only through their physical appearance but also in their mannerism and clothing. Many individuals invest a considerable amount of time and effort in the creation and customization of their avatars, much like the way players engage with The Sims. The Sims is a life simulation game where players create their virtual characters (Sim) and need to manage different aspects of the Sim's life, such as their needs, relationships, jobs, house, etc. The player starts the game by creating a family and introducing them to the world.

The dedication of character creation in Second Life extends to the creation of personalized content, as community members grant items and experiences that vibrate with their unique styles. Moreover, real-world currency is often used to purchase in-game items and enhance the virtual experience.

Paul Hemp (2006) highlights that Second Life strives to offer users a sense of realism. This realism is exemplified by the ability to acquire virtual real estate, such as home and businesses, imitating activities in the real world. Users can establish and operate businesses within the game, selling products and services to fellow participants. In some instances, individuals have achieved

[4] https://www.bbc.com/news/technology-59180273

financial success within Second Life to the extent that they've converted from their job in the real world to work exclusively in the game. Even major corporations like Nike and Levi Strauss have ventured into Second Life, creating, and selling products within the game's virtual economy.

Simon Gottschalk (2011) mentions that educational institutions have also embraced Second Life as a platform for virtual education. This virtual world offers an abundance of diverse environments, starting from hospitals and museums to aquariums and planetariums. Some institutions made use of the platform for professional training, with healthcare professionals practicing surgical procedures on virtual patients. The all-around nature of Second Life has transformed it into a versatile educational tool, enabling innovative approached to teaching and learning.

Chapter 3

Technology Overview

This chapter provides a comprehensive overview of the technologies that will play a crucial role in the project's development, emphasizing the distinctions between these various technologies, thereby providing an understanding of their contributions.

3.1. Technology

3.1.1. Unity vs. Unreal Engine vs. Godot

For the development of the dissertation's solution, it is important to employ a development engine. There are three prominent players in this domain, each offering unique capabilities and features. In the subsequent sections, a comprehensive overview of these development engines will be presented, providing insights for the decision taken.

UNITY

Unity is a versatile game engine that's built in native C++ and works on multiple platforms, such as desktop, mobile, web and console. It allows his users to create both 2D and 3D games and is often referred to as Unity3D. As stated by John K. Haas (2014), Unity was first release on June 6, 2005, with the goal of providing affordable yet professional tools for aspiring game developers, while also making game development accessible to a wider audience.

Unity supports various programming languages such as C#, JavaScript, and Boo for development. It is designed to be user-friendly, especially for newcomers, and comes with extensive documentation to assist users. Due to its popularity, Unity has a large and active community that contributes by creating assets available on Unity's asset store, which users can either access or purchase. Unity was used for the creation of Pokemon Go.

In addition to Pokemon Go, there are numerous other popular games developed using the Unity platform[5]. One such example is Hollow Knight (Team Cherry, 2017), a game following a metroidvania style, where players engage in the tale of a knight battling monsters to combat an infection.

Figure 6 provides a quick overview of the game interface. As we can see, the game is presented in a 2D format, and in the upper-left corner, the player's remaining lives are depicted in the form of heads.

[5] https://www.thegamer.com/unity-game-engine-great-games/

Figure 6 – Hollow Knight[6]

Subnautica (Unknown Worlds Entertainment, 2018) offers an underwater world for players to explore, with the option of using VR technology (as seen in Figure 7).

Figure 7 – Subnautica[7]

Rust (Facepunch Studios, 2018), on the other hand, is an open-world survival game where players begin with only a rock and torch, needing to gather resources to construct a base while protecting it from other players. As the game progresses, players can fortify their base, search for weapons and blueprints, and construct traps to enhance its security.

[6] https://www.ign.com/articles/2018/06/22/hollow-knight-review

[7] http://subnauticagame.com

In Figure 8, we can observe the interface of Rust, where in the bottom right corner the player's status is displayed, including their hunger, health, and thirst levels. When any of these bars drops to a significant level the player is notified, as indicated by the red bar displaying "staring". In the central bottom part of the figure, it is possible to observe what the user has in their quick inventory. They have a rock and a torch, which are the initial kits, along with a guitar and a machete. The game is set in a world where radiation is intense in certain areas, requiring the use of radiation suits, as seen on the player in the centre of the figure.

Figure 8 – Rust[8]

The platform I will be using in this project is Unity, due to having previous knowledge of this platform. The Unity interface can be seen in Figure 9.

[8] https://www.pcgamer.com/rust-review/

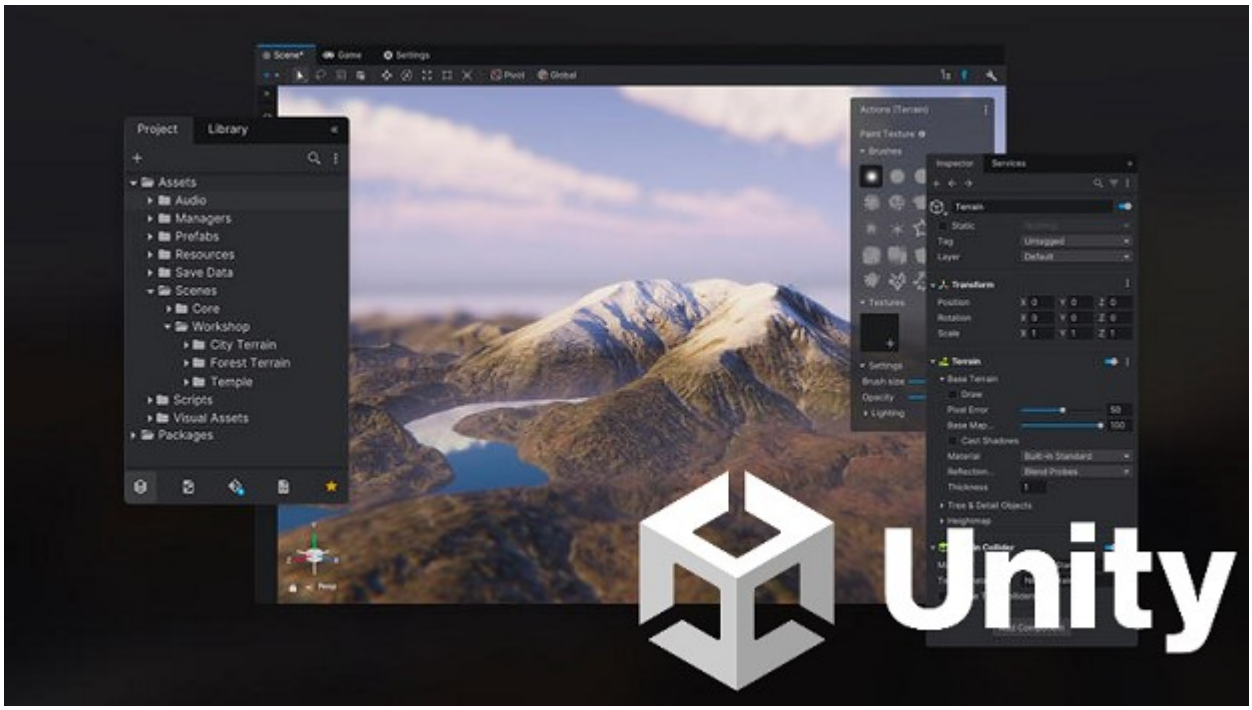

Figure 9– Unity[9]

UNREAL ENGINE

As stated by Antonín Šmíd (2017), the Unreal Engine is a platform used to develop 3D games, developed by Epic Games, which is a video game company. It is a platform created to develop games with the best graphics, to work with Unreal Engine[10] you need to know C++, however the platform provides a visual scripting interface that allows you to design games without programming.

Unlike Unity where you can program for free and only join the premium if you want, Unreal Engine is free, however depending on the value that the game makes, it is necessary to pay a 5% royalty fee.

How the Unreal Engine interface is structured can be seen in Figure 10.

[9] https://dotnet.microsoft.com/en-us/apps/games/unity

[10] https://www.unrealengine.com/en-US/features

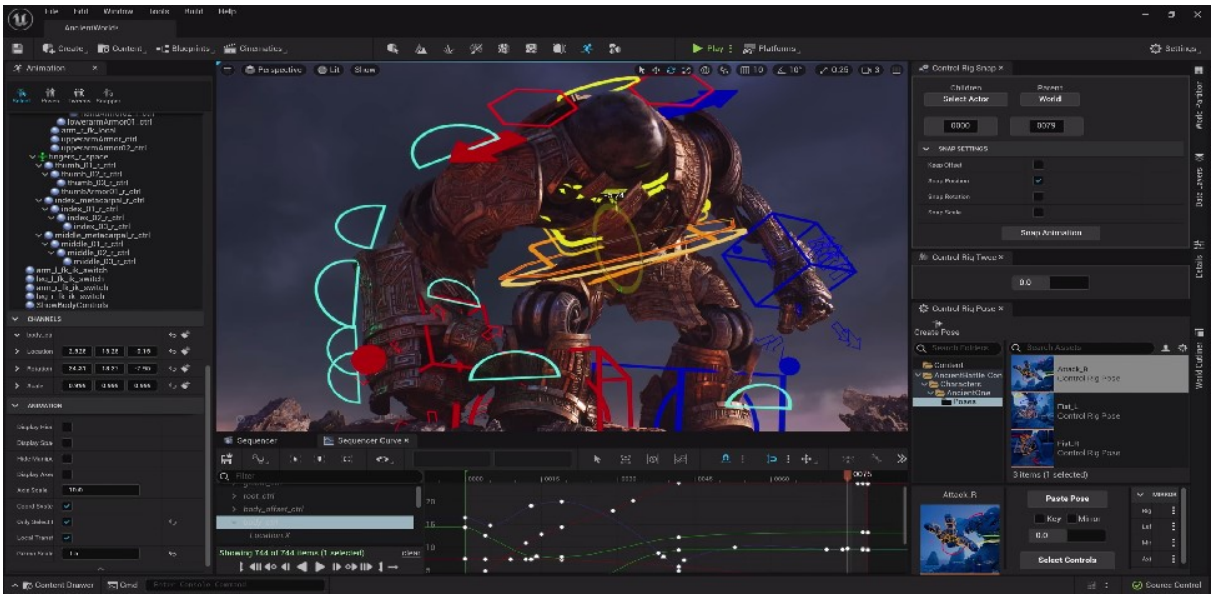

Figure 10 – Unreal Engine[11]

One of the games developed using the Unreal Engine[12] is Fortnite (Epic Games, 2017). This game features three distinct gameplay modes. First is the creative mode, where players can construct or lean to build arenas. Then there's the battle royale mode, where a set number of players start on a flying bus and must parachute down to choose a location on the map, the same can be seen in Figure 11. Each player must search for weapons and resources to survive against others while avoiding the Storm. Additionally, there's a save the world mode, a cooperative option to team up and defeat monsters.

[11] https://beforesandafters.com/2021/05/27/you-can-get-early-access-to-unreal-engine-5-now/

[12] https://www.thegamer.com/great-games-use-unreal-4-game-engine/

Figure 11– Fortnite[13]

Another Unreal Engine game is Hellblade: Senua's Sacrifice (Ninja Theory, 2017), a game praised for its narrative and visuals. In this game, players hear voices that offer guidance, blurring the line between reality and illusion, which can cause confusion.

In Figure 12, it is possible to observe the appearance of the main character.

Figure 12 – Hellblade Senua's Sacrifice[14]

Sea of Thieves (Rare, 2018) is also built on the Unreal Engine. It is a pirate-themed game where players embark on treasure hunts or engage in battles with other pirates, that are other players.

[13] https://twitter.com/FNCompetitive

[14] https://www.gog.com/en/game/hellblade_senuas_sacrifice_pack

In Figure 13, it is possible to gain a comprehensive insight into the graphical elements of the game. The figure depicts the game's aesthetics, featuring a rich and immersive world. At the forefront of this figure are the boats, which serve as the primary means of navigation within the game. A skeleton can be seen, he symbolizes the monsters that guard the islands together, in the quest for hidden treasures the player must confront these monsters. The treasure can go from treasure chests to precious artifacts.

Figure 13 – Sea of Thieves[15]

GODOT

Of the three platforms shown, Godot was the most recent in terms of release. Godot is a free and open-source cross-platform, it's designed to work in multiple platforms such as computers or smartphones, released under the MIT license[16]. Initially the platform used its own language to program, called GDScript, which was based on Python, however through updates it is now possible to program in CSharp and C++. Like Unity, Godot supports desktop, web, and mobile platforms (Chris Bradfield, 2018). Everything that the user creates on Godot belongs to the user, there are no fees or royalties to pay.

In Figure 14, the interface of Godot can be observed, which appears to be quite intricate and exhibits certain resemblances to Blender with the edges.

[15] https://www.seaofthieves.com

[16] https://godotengine.org/license/

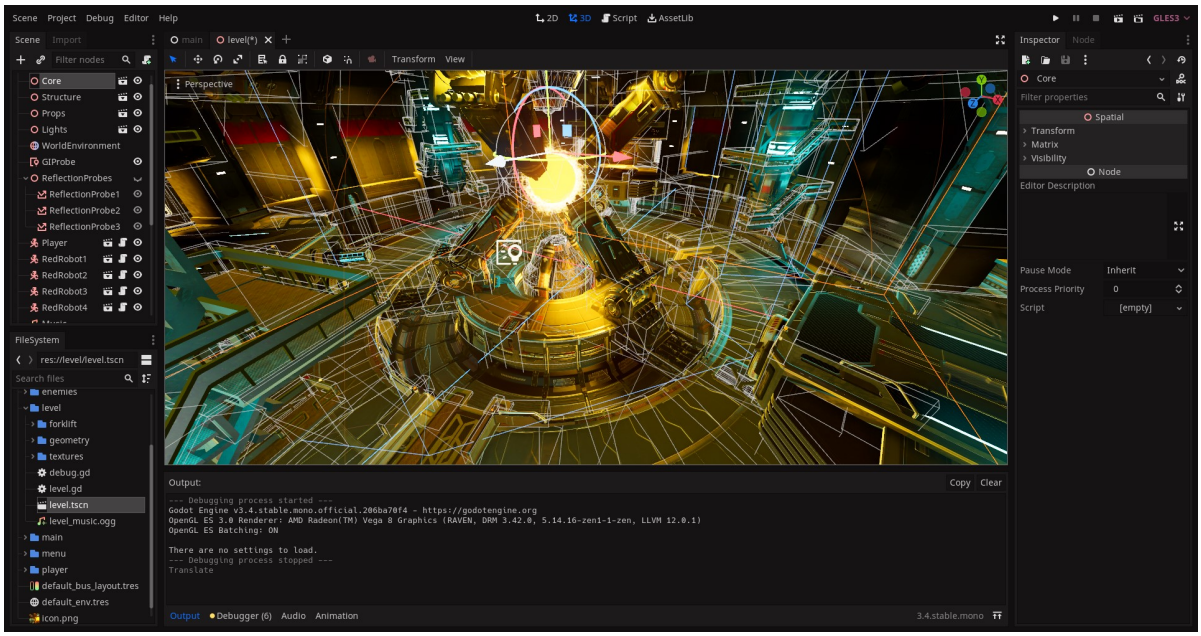

Figure 14– Godot[17]

Some of the games developed in Godot[18] include Nightmare in Squidville (Stephen Hillenburg, 2023), which is based on the TV series SpongeBob SquarePants. In this horror game, Squidward moves to a new neighbourhood with like-minded individuals, and the player (controlling SpongeBob) visits Squidward in this dark and silent place.

Figure 15 depicts Squidward. The fact that the character has red eyes and is in a nighttime setting, which is darker and more ominous, conveys a sense of horror and unease.

Figure 15 – Nightmare in Squidville[19]

[17] https://en.wikipedia.org/wiki/Godot_%28game_engine%29

[18] https://www.thegamer.com/godot-engine-popular-games-best/

[19] https://videogaming.fandom.com/wiki/Nightmare_in_Squidville

Another game is The Legend of Lumina (Wizbane, 2022), where the player takes on the role of a saviour in a dying, infected world. This game involves solving various puzzles scattered throughout the world.

Figure 16 depicts the interface of The Legend of Lumina. It can be observed that is a 2D game that incorporates a story.

Figure 16 – The Legend of Lumina[20]

COMPARISONS

Table 1 provides a more in-depth comparison of the development engines, allowing for a more comprehensive understanding of the differences among these engines.

	Unity	Unreal Engine	Godot
Open source	No	Yes	Yes
Free	It can be free but has premium	It is free but has a 5% royalty fee	It is completely free
3D	Yes	Yes	Yes
2D	Yes	Yes	Yes
Language	C#, JavaScript	C++	GDScript, C# and

[20] https://wizbane.itch.io/the-legend-of-lumina

	and Boo		C++
Support	Desktop, mobile, web, console	Desktop, console, mobile	Desktop, web, mobile

Table 1 – Unity vs. Unreal Engine vs. Godot

1. **Open-Source and free:**
    a. **Unity:** Unity is not open source but provides a free version. It offers premium features through paid packages, with pricing that can vary depending on the selected package.
    b. **Unreal Engine:** Unreal Engine is partially open source, with the source code available for use. However, there is a 5% royalty fee that users need to pay on gross revenue after a certain threshold, typically for commercial projects.
    c. **Godot:** Godot is fully open source and completely free to use. There are no royalties or fees, allowing users to develop games without any financial constraints.
2. **3D and 2D support**:
    a. **Unit**y: Unity offers broad support for both 3D and 2D game development, making it versatile for various game types.
    b. **Unreal Engin**e: Unreal Engine is known for its 3D capabilities but also provides strong support for 2D game development, making it suitable for a wide range of projects.
    c. **Godo**t: Godot, like the other engines, can handle both 3D and 2D game development, offering flexibility for game creators.
3. **Programming language**s:
    a. **Unit**y: Unity supports various programming languages, including C#, JavaScript, and Boo. This versatility allows developers to choose the language they are most comfortable with.
    b. **Unreal Engin**e: Unreal Engine uses C++ for scripting and programming, providing powerful control over game logic and performance.
    c. **Godo**t: Godot supports both C# and C++, and it also has its own scripting language called GDScript. This gives developers options for coding in familiar languages or using GDScript for rapid development.
4. **Platform support:**
    a. **Unity:** Unity consists of an extensive list of supported platforms, including Windows, macOS, Linux, iOS, Android, web, and several console platforms. This broad range of compatibility makes it suitable for a variety of projects.
    b. **Unreal Engine:** Unreal Engine provides support for Windows, macOS, Linux, console platforms, and iOS, Android. It is particularly known for its high-quality console and PC game development capabilities.
    c. **Godot:** Godot supports desktop platforms, web, and mobile platforms. It offers a

strong foundation for cross-platform development.

3.1.2. Vuforia vs. Arcore vs. ARKit

To develop an AR application, it is important to consider the appropriate software to use. In this section, three AR software options, representing major players in the industry, will be presented.

VUFORIA

In accordance with Cheng Xiao & Zhang Lifeng (2014), Vuforia was developed by Qualcomm company, later bought by PTC Inc., and supports both Android and iOS, in addition to these two, it works with Unity Editor and UWP (Universal Windows Platform). Compared to the ARKit, Vuforia allows you to run your AR platforms on iOS devices older than iPhone 6, in addition, if the device where the application is running supports both ARCore and ARKit SDK, Vuforia can use them, otherwise it uses its own platform.

With Vuforia it is possible to scan a QR code or create an image target to make a virtual object appear on the location, besides of providing lots of virtual elements for interaction, also being recognized for its ability to track images and objects, and upload models.

Figure 17 illustrates an example of how Vuforia can be utilized, in this case for an augmented reality shopping application, enabling consumers to visually inspect a product before making a purchase.

Figure 17– Vuforia[21]

[21] https://library.vuforia.com/objects/image-targets

ARCore

In line with Oufqir, Z. et al. (2020), ARCore is an SDK that has been developed by Google for the purpose of creating AR applications. This SDK supports both iOS and Android systems and is free to use.

ARCore, like ARKit, is also known for its plane detection, face and image tracking and other functionalities. With their motion tracking functionality, it is possible to detect the current position of the mobile phone in relation to the surroundings. It is also a good application for e-commerce due to this functionality and the way it integrates virtual objects in the real world.

Figure 18 illustrates a character being displayed in augmented reality through a mobile phone.

Figure 18– ARCore[22]

ARKit

In accordance with Oufqir, Z. et al. (2020), ARKit is an SDK that was developed by Apple, aiming to help build AR platforms. ARKit is targeted at Apple devices such as iPhones and iPads, supporting only the iPhone 6 or higher and iPad pro models.

ARKit is known for its plane detection, face tracking or even motion capture. It also supports several features, such as image detection and tracking, and the placement of virtual objects on surfaces. In addition to that it also supports models and scenes created in Reality Composer. For annual Developer Program account, you must pay $99.

Figure 19 illustrates the option for ARKit to be employed across various devices seamlessly, enabling the presentation of augmented reality elements without any issues.

22 https://www.theverge.com/2017/8/29/16219696/google-arcore-augmented-reality-platform-announce-release-pixel-samsung

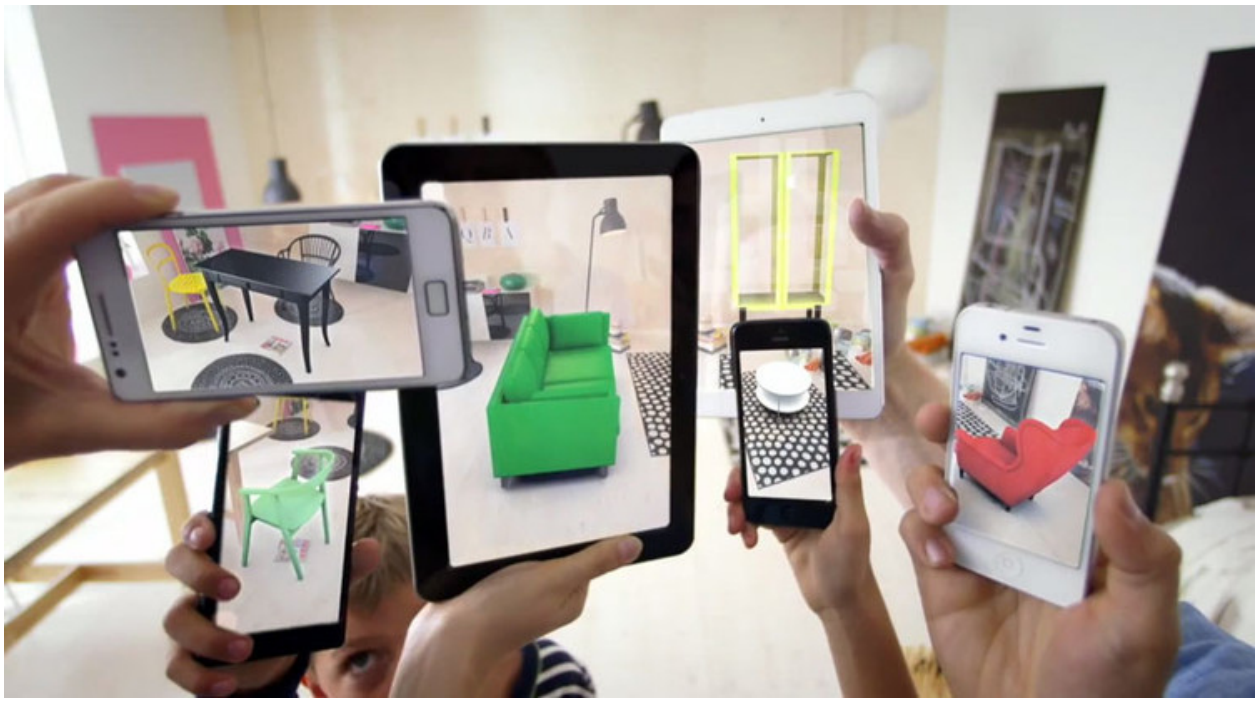

Figure 19– ARKit[23]

COMPARISONS

Similar to the preceding table, Table 2 provides a comprehensive overview of the software comparisons in a manner that is easy to grasp.

	Vuforia	ARCore	ARKit
Support	Android, iOS, Unity Editor and UWP	Android and iOS	iOS
Free	It can be free but has premium	Yes	Yes but free of $99 for store distribution
3D recognition	Yes	Yes	Yes
2D recognition	Yes	Yes	Yes

Table 2 – Vuforia vs, ARCore vs. ARKit

1. **Platform support:**
    a. **Vuforia:** Vuforia is a versatile AR development platform with support for multiple platforms, including Android, iOS, Unity Editor, and UWP. This versatile platform

[23] https://www.archdaily.com/879403/the-real-star-of-the-apple-keynote-arkit-augmented-reality-technology

compatibility ensures that AR applications created with Vuforia can reach a wide audience across various devices and operating systems.

   b. **ARCore:** ARCore is Google's AR development platform, offering support for both Android and iOS devices.
   c. **ARKit:** ARKit is Apple's AR development framework, exclusively available for iOS devices. When compared to ARCore, ARKit becomes the preferred choice for projects focused on iOS, while ARCore remains the top choice for Android AR applications.

2. **Free:**
   a. **Vuforia:** Vuforia provides developers with flexibility by offering both a free tier and a premium option. This allows developers to explore and utilize the platform for free, making it an ideal choice for those who want to experiment with AR development before committing to a purchase.
   b. **ARCore:** ARCore is entirely free to use, meaning developers can access its full range of features without incurring any additional costs, making it an attractive option for budget-conscious developers.
   c. **ARKit:** ARKit is also free for development purposes. However, there is a $99 fee for store distribution on the Apple App Store.
3. **3D and 2D recognition:**
   a. **Vuforia:** Vuforia offers support for both 2D and 3D object recognition, making it a robust choice for applications that require an extensive range of tracking and recognition capabilities.
   b. **ARCore:** ARCore similarly supports both 2D and 3D recognition.
   c. **ARKit:** ARKit, like the others, offers support for both 2D and 3D recognition.

3.1.3. VR headsets (PC VR vs. standalone VR vs. smartphone VR)

In order to utilize the potential of VR, it is important to make use of hardware that enable the full experience of this technology. We need to delve into the realm of the VR market to select the three prominent players in the industry and observe which will be the most suitable for the project to be developed.

PC VR

VR is a digital way to experience 3D to the fullest through a VR headset or a HDM (head-mounted display).

PC VR is widely used in gaming but can also be used in education and tourism[24]. As statted by Angelov et al. (2020), one of the best well-known glasses of the PC VR type are the Oculus Rift. This type of glasses in turn offers more immersion than other types of VR, but they are more limited in

[24] https://www.aniwaa.com/guide/vr-ar/types-of-vr-headsets/

terms of movements due to needing a certain space for its setup and in terms of hardware it requires a more powerful computer.

In Figure 20 it is possible to observe the design of the Oculus Rift, consisting of a head-mounted device and two hand-held controllers.

Figure 20 – Oculus Rift[25]

STANDALONE VR

Standalone VR (Angelov et al., 2020), as the name mentions, does not need a connection to a computer or a mobile phone due to being wireless and having already all the computing hardware components required, so it is not limited to one location only. This type of VR only needs to be charged and perhaps creating an account on some VR platforms, apart from that it provides sensors, built-in processors, battery and even storage memory. But compared to PC VR, this one offers a lower quality in terms of graphics.

Standalone VR can be used for casual games, for example those games for socializing with people, or it can be used for watching movies. One of the most used brands is Pico Neo.

In Figure 21, it can be observed the design of the Pico Neo, which, in terms of the number of components doesn't differ from the oculus Rift, except for aesthetic purposes.

25 https://www.backmarket.pt/pt-pt/p/oculus-rift-oculos-vr-realidade-virtual/b4d764f0-fdaa-4d34-849b-274101e559e8

Figure 21 – Pico Neo[26]

SMARTPHONE VR

The Smartphone VR only asks the user to place their mobile phone inside the headsets, it provides various lenses to create a sense of depth for the user. Like PC VR, the quality of the smartphone VR experience varies depending on the hardware being used, meaning the more powerful the mobile phone the better the experience. As stated by Hillmann & C. (2019), this type of VR has the same uses as Standalone VR, it is user-friendly and perhaps more affordable than the other options, however it has a rather limited immersion compared to the VRs mentioned earlier but his greatest advantage is the social media discovery. One of the most used brands is Samsung Gear VR.

Figure 22 illustrates that VR for mobile devices involves the use of a head-mounted device exclusively.

[26] https://arpost.co/2021/12/01/review-pico-neo-3-pro-vr-headset/

Figure 22 – Samsung Gear VR[27]

COMPARISONS

The various types of VR are compared in Table 3 for enhanced clarity and understanding. This method becomes more accessible and perceptible to the reader.

	PC VR	Standalone VR	Smartphone VR
Graphics Quality	High	Low	Low
Wireless	No	Yes	Yes
Cables	Yes	Can be used without cables	No
Price	Pricey	Medium	Cheap

Table 3 – PC VR vs. Standalone VR vs. Smartphone VR

1. **Graphics quality:**
    a. **PC VR:** The PC VR systems offer high-quality graphics because they have the support of computers that can deliver detailed and realistic visuals.
    b. **Standalone VR:** The standalone VR have a built-in hardware, making it less powerful than the PCs, offering a lower graphic quality, with less detail and realism compared to PC VR.
    c. **Smartphone VR:** Smartphone VR relies on the power of the connected smartphone, which in comparison to a computer, has lower graphics quality. This

[27] https://www.samsung.com/pt/mobile-accessories/gear-vr-r325-sm-r325nzvctph/

results in a graphics quality that, while enjoyable, may not immerse users to the same extent.

2. **Wireless:**
    a. **PC VR:** PC VR requires wired connections between the headset and the computer. This limitation can restrict user's movements and affect the overall immersive experience.
    b. **Standalone VR:** In contrast, standalone VR systems are designed with wireless freedom in mind. This design allows users to move more freely within their virtual environments, enhancing the sense of immersion.
    c. **Smartphone VR:** Like standalone VR, smartphone VR also benefits from wireless connectivity. Relying on the smartphone for both sensors and interface, users can enjoy a wire-free experience.
3. **Cables:**
    a. **PC VR:** The PC VR often involves a web of cables, potentially complicating setup, and movement.
    b. **Standalone VR:** While primarily wireless, standalone VR systems may incorporate optional cables to augment the user experience.
    c. **Smartphone VR:** The smartphone VR is completely cableless, relies on the smartphone alone, eliminating the need for any additional cables.
4. **Price:**
    a. **PC VR:** In comparison to the other's VRs, these setups tend to be more expensive due to the cost of the computer and the headsets.
    b. **Standalone VR:** The standalone VRs are in the between in terms of price, it is more affordable than the computer VR but is pricier than the smartphone VR:
    c. **Smartphone VR:** This is the most budget-friendly option for VR.

3.1.4. MAR systems (optical see-through vs. video see-through)

Within the domain of MAR systems, there exists two distinct types which will be presented in the subsequent sections.

OPTICAL SEE-THROUGH

According to Cheng Xiao & Zhang Lifeng (2014), the optical see-through systems allow users to observe the real world and the virtual objects they have added to it by combining the two through a semi-transparent slanted mirror.

Figure 23 shows how an optical see-through system works in a detailed manner. This technology enables users to see a real image displayed on a monitor. It achieves this by using a special mirror called a half-silvered mirror and computer-generated graphics.

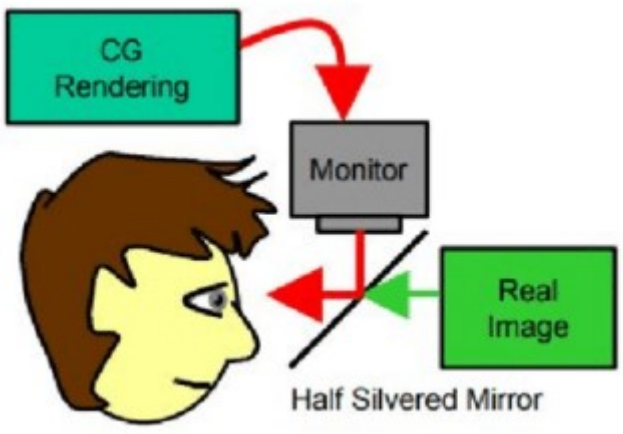

Figure 23 – Optical see-through system[28]

VIDEO SEE-THROUGH

According to Cheng Xiao & Zhang Lifeng (2014), the video see-through systems provide video through the cameras that are inside the device, thus allowing the user to see virtual objects that are superimposed in the real world. It is used if it is necessary to experience something remotely.

Figure 24 illustrates how a video see-through system works. In this system, a real image is captured by a camera. This captured image is then processed to combine it with a computer-generated image. After this composition process, the final image, which is a blend of the real-world view and the computer-generated elements, is shown to the user.

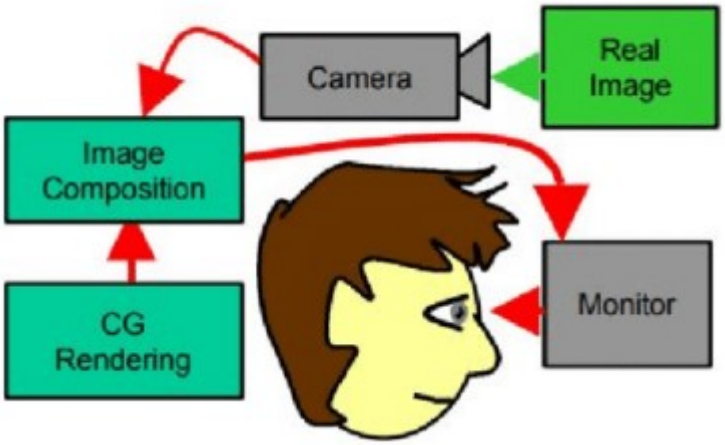

Figure 24 – Video see-through system[27]

[28] https://www.researchgate.net/figure/Figure-2-Video-versus-optical-see-through-augmented-reality_fig1_320868121

COMPARISONS

Table 4 consolidates the information pertaining to each MAR System, allowing for a more comprehensive and easily understandable comparison.

	Optical see-through	Video see-through
Graphics Quality	High	Low
Wireless	No	Yes
Cables	Yes	Can be used without cables
View of the world	Through semi-transparent mirrors	Two miniature video cameras

Table 4 – Optical see-through vs. Video see-through

1. **Graphics quality:**
    a. **Optical see-through:** The graphics quality in optical see-through augmented reality is high. Virtual objects seamlessly blend with the physical environment, creating a remarkably realistic and immersive experience.
    b. **Video see-through:** In contrast, the graphics quality in video see-through AR tends to be lower than optical see-through. This is because the virtual objects cover or obscure the live video feed from the real world, making it more challenging to achieve the same level of realism and integration.
2. **Wireless:**
    a. **Optical see-through:** Optical see-through AR often necessitates a connection to external computing devices or power sources. This reliance on external components can limit mobility to some extent.
    b. **Video see-through:** Video see-through AR, similar to smartphone-based AR, typically does not require direct connections to external devices. This wireless nature enhances mobility and convenience.
3. **Cables:**
    a. **Optical see-through:** In the optical see-through setup, cables are commonly employed for both power supply and data transmission.
    b. **Video see-through:** Video see-through AR systems, depending on their design, can often eliminate the need for cables altogether.
4. **View of the world:**
    a. **Optical see-through:** Optical see-through AR offers users a view of the world through semi-transparent mirrors. that allows digital graphics to be overlapped on the field of view.
    b. **Video see-through:** In video see-through AR, the user's view of the world is captured

through two miniature video cameras. These cameras provide a live video feed of the real-world environment.

3.1.5. GitHub vs. GitLab vs. Bitbucket

Regardless of the project's scale, it is advisable to maintain version control in case something goes wrong. To achieve this, version control software is recommended. Below are listen some of the well-known version control software options.

GitHub

In conformity with Cosentino et al. (2016), GitHub is a highly popular online platform that plays a crucial role in software development. It serves as a virtual hub were developers from all around the word collaborate on various coding projects. This platform is known for being an open-source software, where projects are developed by a community of contributors. Figure 25 displays the GitHub logo.

In conformity with Jarczyk et al. (2014), GitHub provides an environment for virtual teams of developers to work together. Every individual who uses GitHub can create their own repository, which is essentially a digital workspace where they can store their code, documentation, and other project-related resources. Users can invite other users to join their repositories, making it possible to work collectively on projects. Unlike some platforms that suggest projects to users based on skills, GitHub leaves this decision entirely up to the user, this means, each user decides which project to work on and how to allocate their time. This process makes GitHub organic and self-organizing.

Figure 25– GitHub[29]

[29] https://codeburst.io/why-you-should-start-using-github-right-now-e817d213c6ff

GITLAB

In accordance with Adam O'Grady (2018), GitLab is a web-based tool used for DevOps purposes. It follows an open-core business model and is licensed under the MIT license. It serves as a platform for hosting Git repositories, like GitHub.

GitLab is centred around the Git version control system, enabling users to store code and manage issues related to their projects. It operates on top of Git, ensuring that project contributors have a local copy of the project on their computers. The platform offers a web interface that facilitates advanced Git workflows, suggesting a productive and efficient approach for working with Git to enhance productivity and user-friendliness.

The GitLab interface is shown in Figure 26, where the issues that the developers have marked as important to resolve in the project can be seen.

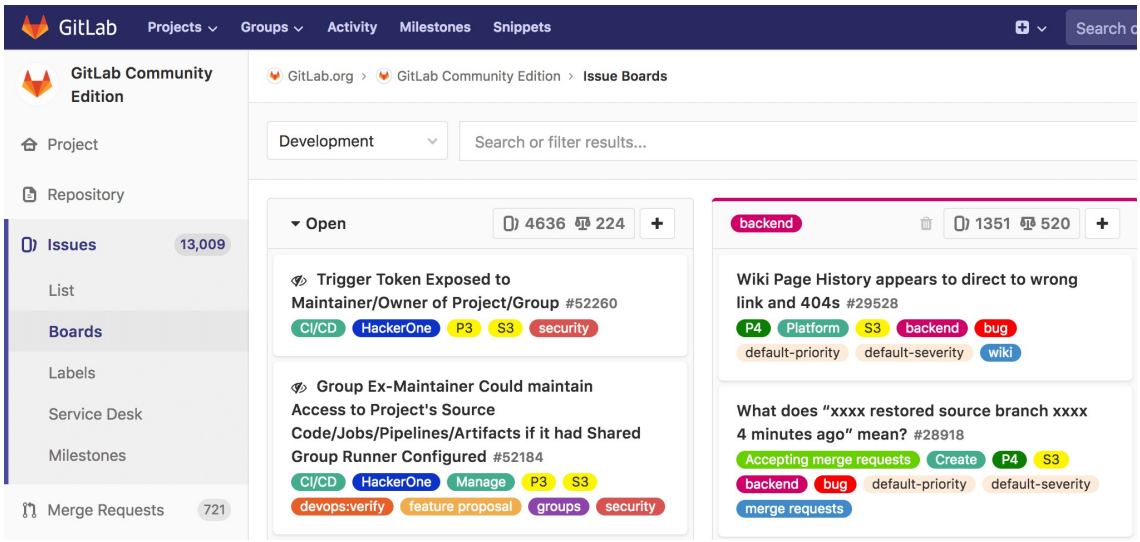

Figure 26– GitLab[30]

BITBUCKET

As stated by Abi Tyas Tungga (2023), Bitbucket, a repository hosting platform, initially set itself apart from its competitors like GitHub by supporting both Git and Mercurial, which later was discontinued focusing only on Git. Bitbucket's integration with JIRA, Atlassian's renowned issue and project tracking software, provides a smooth experience for users. JIRA, originally designed as a bug tracker, has evolved into a versatile tool capable of handling various aspects of project management, including bug tracking, issue management, service desk operations, and project oversight. Bitbucket allows developers to store, track changes to, an collaborate on their codebases using version control. It supports features like pull requests, code reviews, branch comparison, commit history and more.

The BitBucket interface shown in Figure 27 looks a bit different from the GitLab interface showed earlier in Figure 26. In Figure 27 the use of pipelines is being displayed.

[30] https://www.theregister.com/2020/05/22/gitlab_130_released_devsecops_in/

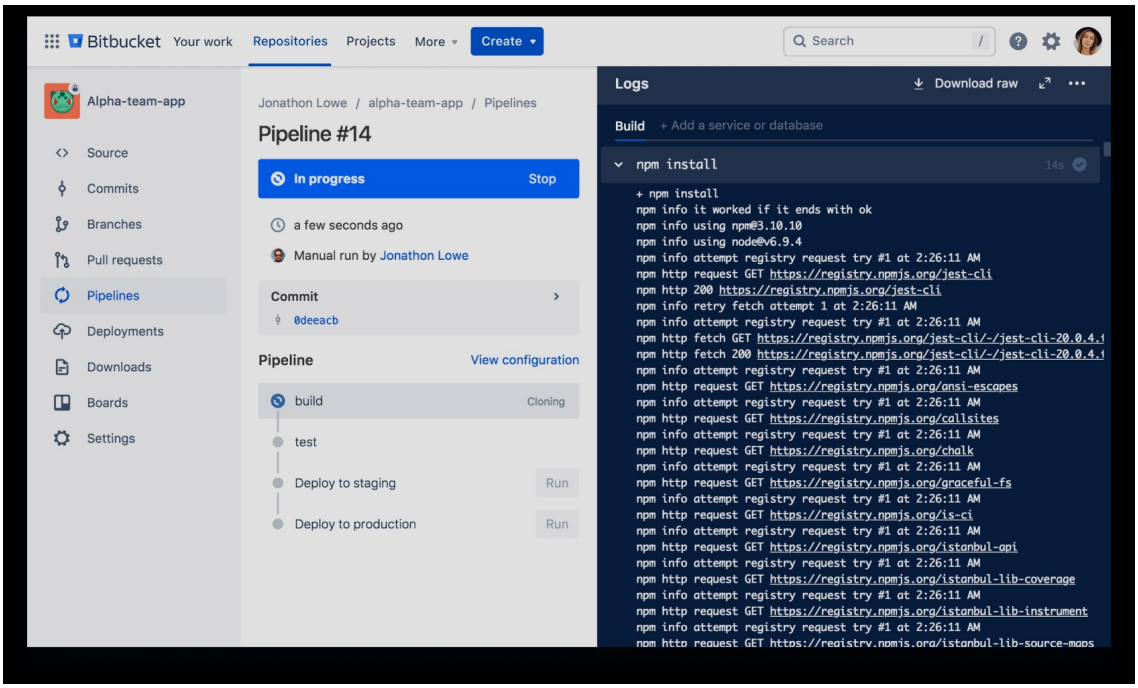

Figure 27– Bitbucket[31]

COMPARISONS

A concise comparison of the three mentioned version control software is presented in Table 5.

	GitHub	GitLab	Bitbucket
Open source	No	Yes	No
Nº of private repositories	unlimited	Unlimited	unlimited
Nº of contributors	unlimited	unlimited (paid) 5 (free)	unlimited (paid) 5 (free)
Storage limit	2Gb (free) 5GB (paid)	10GB	4GB

Table 5 – GitHub vs. GitLab vs. Bitbucket

1. **Open source:**
   a. **Github:** GitHub's source code is not publicly available, which means it is not open source.
   b. **GitLab:** GitLab's source code is publicly accessible and can be modified, making it an open source platform.
   c. **BitBucket:** Similar to GitHub, BitBucket is not open source as its source code is not publicly available.

[31] https://bitbucket.org

2. **Nº of private repositories:**
    a. **Github:** GitHub provides an unlimited number of private repositories, allowing users to create as many as they need.
    b. **GitLab:** GitLab also offers an unlimited number of private repositories, on par with GitHub's offering.
    c. **BitBucket:** BitBucket aligns with GitHub and GitLab, offering users an unlimited number of private repositories.
3. **Nº of contributors:**
    a. **Github:** GitHub allows an unlimited number of contributors for both free and paid versions, promoting collaboration without restrictions.
    b. **GitLab:** The free version of GitLab allows up to 5 contributors per project without requiring payment. For more contributors, users need to subscribe to the premium version.
    c. **BitBucket:** Similar to GitLab, BitBucket offers the same limitation. The free version allows up to 5 contributors per project, while additional contributors require a premium subscription.
4. **Storage limit:**
    a. **Github:** GitHub's free version provides up to 2GB of storage for repositories, while the paid version offers up to 5GB.
    b. **GitLab:** GitLab is generous with storage, offering 10GB of storage for both free and paid accounts, ensuring enough space for projects.
    c. **BitBucket:** BitBucket provides 4GB of storage for both free and paid accounts, offering a reasonable amount of storage capacity for version control.

3.1.6. Node.js vs. PHP vs. Python

There are several programming languages that can be used to create a server for handling HTTP requests. In the following discussion, we will explore some of the well-known programming languages commonly used for server development.

NODE.JS

As stated by S. Tilkov & S. Vinoski (2010), Node.js, often referred as Node, is a runtime environment that enables developers to use JavaScript for server-side programming. This makes it possible to create a web application that can handle server-side tasks such as data processing, file manipulation and interaction with the database. Node.js is primarily implemented using C and C++, which provides the advantage of being closer to the system hardware, that results in significant improved performance and efficient memory utilisation.

One feature of Node.js is that instead of waiting for each task to complete before moving to the next one, it uses event-driven, non-blocking I/O model, that allows it to efficiently manage multiple tasks consecutively without relying on traditional multithreading. As stated by S. et al. (2016), Node.js

handles well a large number of connections simultaneously, making it ideal for application that require real-time interaction and responsiveness.

This technology is especially effective in scenarios where fast data exchange between a web server and clients is crucial. Comparing it to other server-side languages, it demonstrates impressive performance benchmarks. One example of that is the execution speed being two times faster than PHP and six to seven times faster than Python.

Figure 28 illustrates how to write code in Node.js using the Visual Studio Code program.

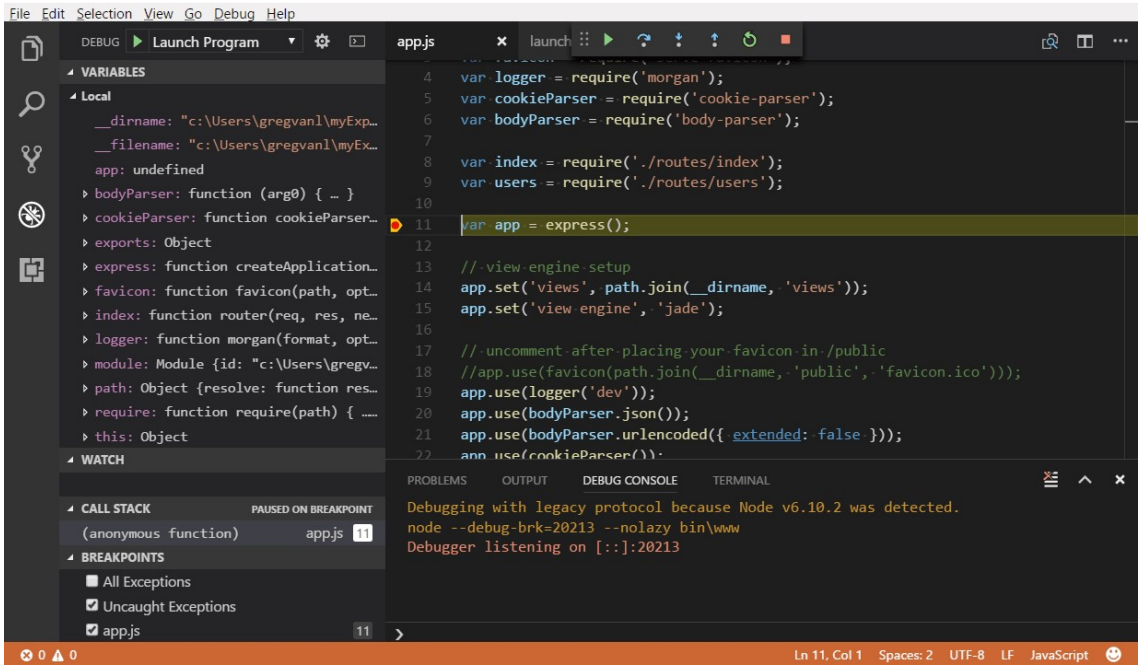

Figure 28 – Node.js[32]

PHP

As stated by Luke Welling & Laura Thomson (2001), PHP is a programming language used on the server side to create dynamic content for websites. It can be inserted into HTML pages, and when a page is accessed, the web server processed the PHP code to generate HTML or other outputs visible to users. PHP is open source, which means that the user can access, modify, and share its source code as he wishes.

In accordance with Rasmus Lerdorf & Kevin Tatroe (2002), PHP has various functionalities, such as server-side scripting, command-line scripting and client-side GUI applications. It works on most operating systems such as Unix, Windows, and Mac OS X. It has extensive database support and comes with a library of pre-written code for common tasks.

PHP was initially created to fulfil a developer's needs and evolved over time to meet the community's requirements.

In Figure 29, it is possible to see how the PHP language is structured. The figure depicts a piece

[32] https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

of code that handles access to a specific section.

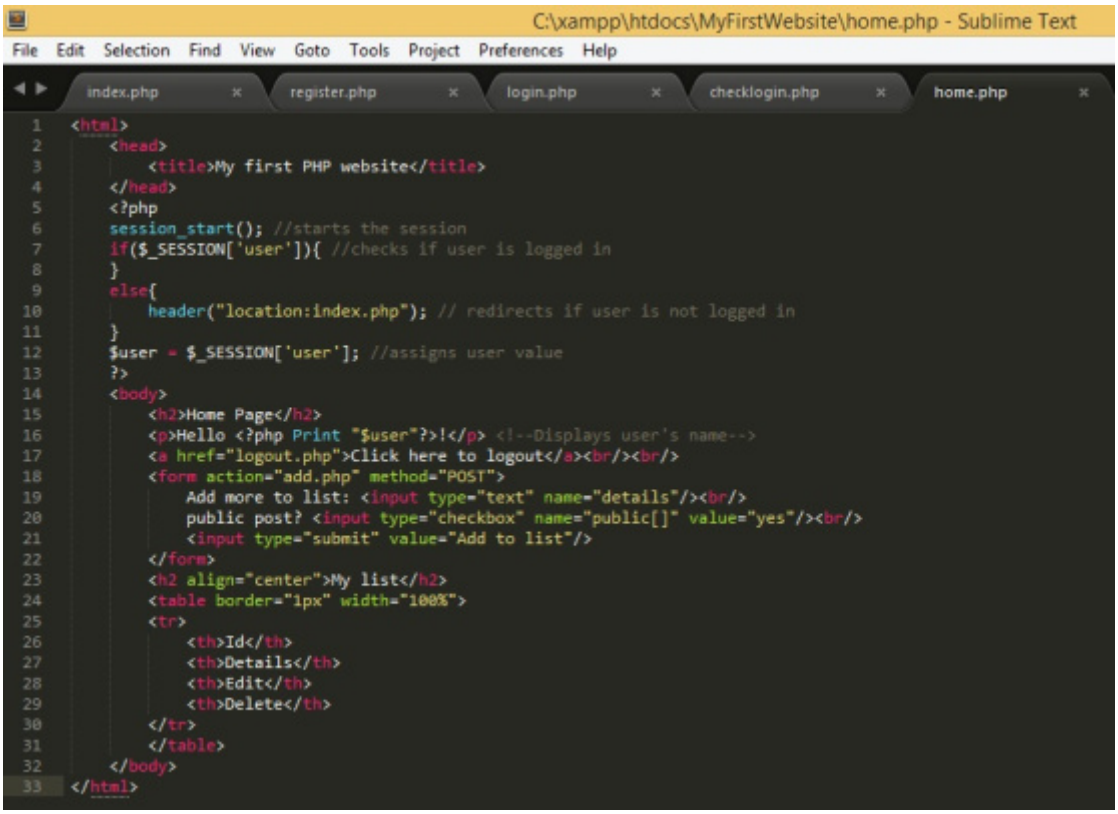

Figure 29 - PHP[33]

PYTHON

In accordance with Simon Yuill & Harry Halpin (2006), Python is a robust and elegant programming language known for its readability and ease of comprehension. It is also open source, has a single main version and it is supported by a large and welcoming community of programmers.

As stated by Guido van Rossum (1995), Python is both beginner-friendly and powerful, featuring efficient high-level data structures and a straightforward approach to the object-oriented programming.

Its syntax, dynamic typing, and interpreted nature make it particularly suitable for scripting and quickly creating applications across various platforms. While Python is user-friendly, it is a genuine programming language, providing more organization and assistance for complex programs compared to shell scripts or batch files. It also offers more error checking than languages like C.

Python incorporates advanced data types like adaptable arrays and dictionaries due to its high-level nature. Additionally, it also supports modular programming, allowing the user to divide his code into reusable components that can be shared with other Python projects.

As per Guido van Rossum (2011), furthermore, Python is categorized as free software, granting the user the liberty to run, copy, distribute, analyse, modify, and enhance the software.

In Figure 30, a representation of the structure Python code file is observed. In this figure, a

[33] https://www.codeproject.com/Articles/759094/Step-by-Step-PHP-Tutorials-for-Beginners-Creating

"while" loop is depicted, designed to check for a game over scenario based on various ongoing events. The program responds to each event accordingly.

```
57          #Gameplay.
58          while not is_game_over:
59
60              #gets all the events occuring at any given time
61              for event in pygame.event.get():
62              #If there is a quit type event - exit out of loop
63                  if event.type == pygame.QUIT:
64                      is_game_over = True
65                  # Detect when key is pressed down
66                  elif event.type == pygame.KEYDOWN:
67                      # Move up if up key pressed
68                      if event.key == pygame.K_UP:
69                          direction = 1
70                      # Move down if down key is pressed
71                      elif event.key == pygame.K_DOWN:
72                          direction = -1
73                  # Detect when key is released
74                  elif event.type == pygame.KEYUP:
75                      # Stop movement when key no longer pressed
76                      if event.key == pygame.K_UP or event.key == pygame.K_DOWN:
77                          direction = 0
78                  print(event)
79
```

Figure 30 - Python[34]

COMPARISONS

In Table 6, a comparison is made among the three programming languages, based on the detailed texts provided for each one above.

	Node.js	PHP	Python
Ease of learning	Easy and quick to learn	Easy to learn	Easy and quick to learn
Speed	Faster than PHP and Python	3x faster than python	Fast
Library support	Multiples libraries	Great number of extensions	Offers support for all kind of applications
Community	Less community support compared to PHP	Large community support	Good community support

[34] https://www.jpl.nasa.gov/edu/learn/project/code-a-mars-rover-driving-game/

Scalability	High	Quite scalable	Not the best

Table 6 – Node.js vs. PHP vs. Python

1. **Ease of learning:**
    a. **Node.js:** Node.js is known for its beginner-friendly nature, offering a smooth and rapid learning curve. This makes it highly appealing to beginners to web development.
    b. **PHP:** PHP is also relatively easy to learn, making it accessible for individuals new to programming.
    c. **Python:** Python, like Node.js, is known for its ease of learning and quick grasp. It is particularly attractive to beginners due to its clean and readable syntax.
2. **Speed:**
    a. **Node.js:** Among these three languages, Node.js stands out as the fastest. Its exceptional speed is especially noticeable when handling I/O-bound tasks.
    b. **PHP:** PHP is generally faster than Python, particularly in web development scenarios.
    c. **Python:** Python is fast, but it doesn't achieve the same level of speed as Node.js.
3. **Library support:**
    a. **Node.js:** Node.js has an extensive library, offering numerous modules and packages for various purposes.
    b. **PHP:** PHP provides a wide range of extensions and libraries that prove useful in web development projects.
    c. **Python:** Both Python and Node.js have large libraries available for diverse use cases, making them versatile choices for developers.
4. **Community:**
    a. **Node.js:** Node.js enjoys a good community, although it tends to be smaller in comparison to PHP.
    b. **PHP:** PHP has a large and vibrant community that actively contributes to the language's growth and development.
    c. **Python:** Python, like PHP, has a sizable and supportive community, which means generous resources and assistance for developers.
5. **Scalability:**
    a. **Node.js:** Node.js is highly regarded for its scalability, particularly in real-time applications.
    b. **PHP:** PHP is scalable, but it requires careful architectural planning to ensure optimal performance in larger projects.
    c. **Python:** Python may not be the best choice for highly scalable applications due to limitations in its execution model.

3.1.7. MySQL vs. Microsoft SQL Server vs. Firebase

There are several databases that can be utilized in the project development, with three specific

ones that were selected to be presented in the text below.

MySQL

As stated by Steve Suehring (2002), the Microsoft SQL Server reflects Microsoft's traditional strengths, strategies, and limitations, offering good stability, user-friendliness, and accessible support.

On the other hand, MySQL provides a comprehensive solution: it operates on various platforms, has a low overall cost, and remains stable. Its documentation is strong, supported by a comprehensive website from MySQL AB that contains reference materials. They also provide high-quality support, including a service allowing developers to remotely access servers to address issues and offer optimization assistance.

In accordance with Paul DuBois (2000), database software has become more attainable, particularly open-source options like MySQL. MySQL is a SQL client/server relational database management system that features an SQL server, client tools, administration utilities, and a programming interface for creating custom programs. Originally recognized for its speed and simplicity, MySQL grew in popularity but faced criticism for lacking features such as transactions and foreign key support. However, it evolved to cater the enterprise applications. MySQL's portability extends to commercial operating systems like Mac OS X, HP-UX, and Windows, scaling from regular hardware to enterprise servers.

In Figure 31, it is possible to see the MySQL Workbench interface. On the left side, there is a list of existing schemas, information about a table, and the databases. In the middle, there is more detailed information about a specific database, and on the right side, it is possible to find details about whether the server is running and other performance-related information.

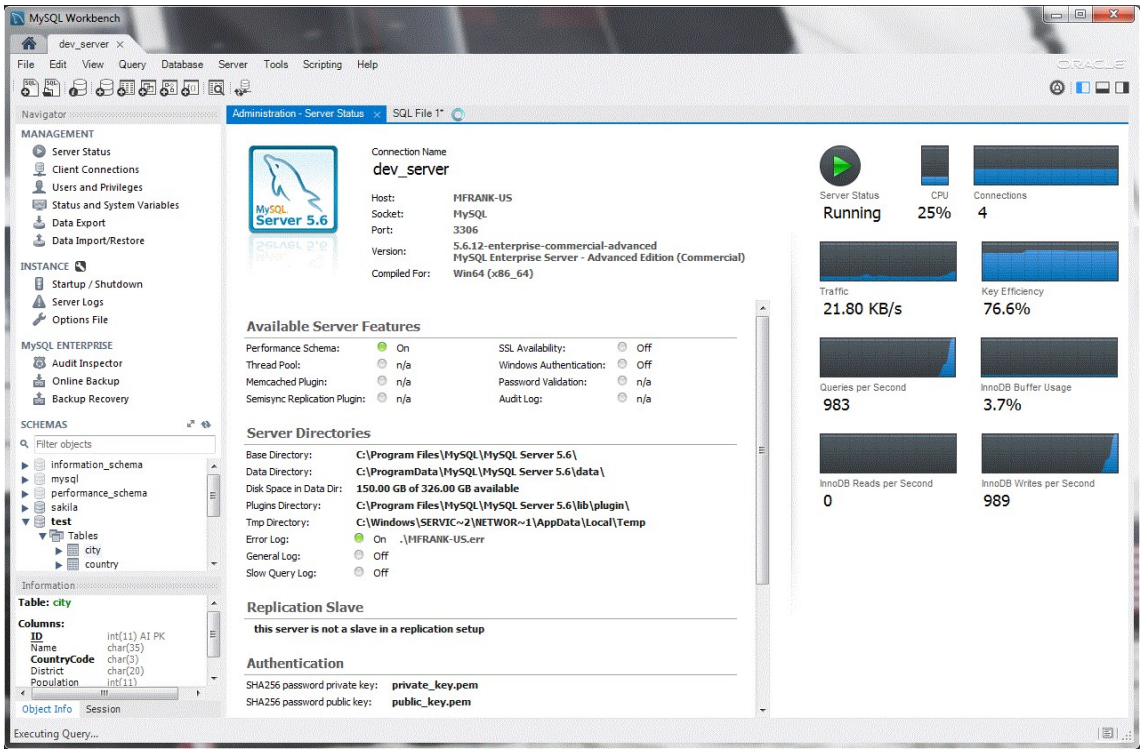

Figure 31– MySQL[35]

MICROSOFT SQL SERVER MANAGEMENT STUDIO

SQL Server Management Studio (SSMS)[36] is an integrated platform designed for overseeing various aspects of SQL infrastructure. It offers the capability to connect to, set up, control, oversee, and create components within SQL Server, Azure SQL Database… SSMS serves as an inclusive toolset, blending a wide array of graphical utilities with script editors, catering to individuals of varying expertise levels including developers and database administrators.

SSMS[37] equips users with resources to configure, manage, and administer Microsoft SQL Server instances. It combines diverse visual design tools, graphical interfaces, and script editors to simplify interactions with SQL Server. Key features of SSMS encompass the Object Explorer for managing all SQL Server objects, the Template Explorer for efficient query and script development through reusable text files, and the Solution Explorer for structuring administration items like queries and scripts within projects.

In the SSMS interface, as shown in Figure 32, you can see the existing databases on the left side, as well as tables, views, and other information related to the specific database the user is working on.

---

35 https://www.mysql.com/products/workbench/

36 https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16

37 https://www.techtarget.com/searchdatamanagement/definition/Microsoft-SQL-Server-Management-Studio-SSMS

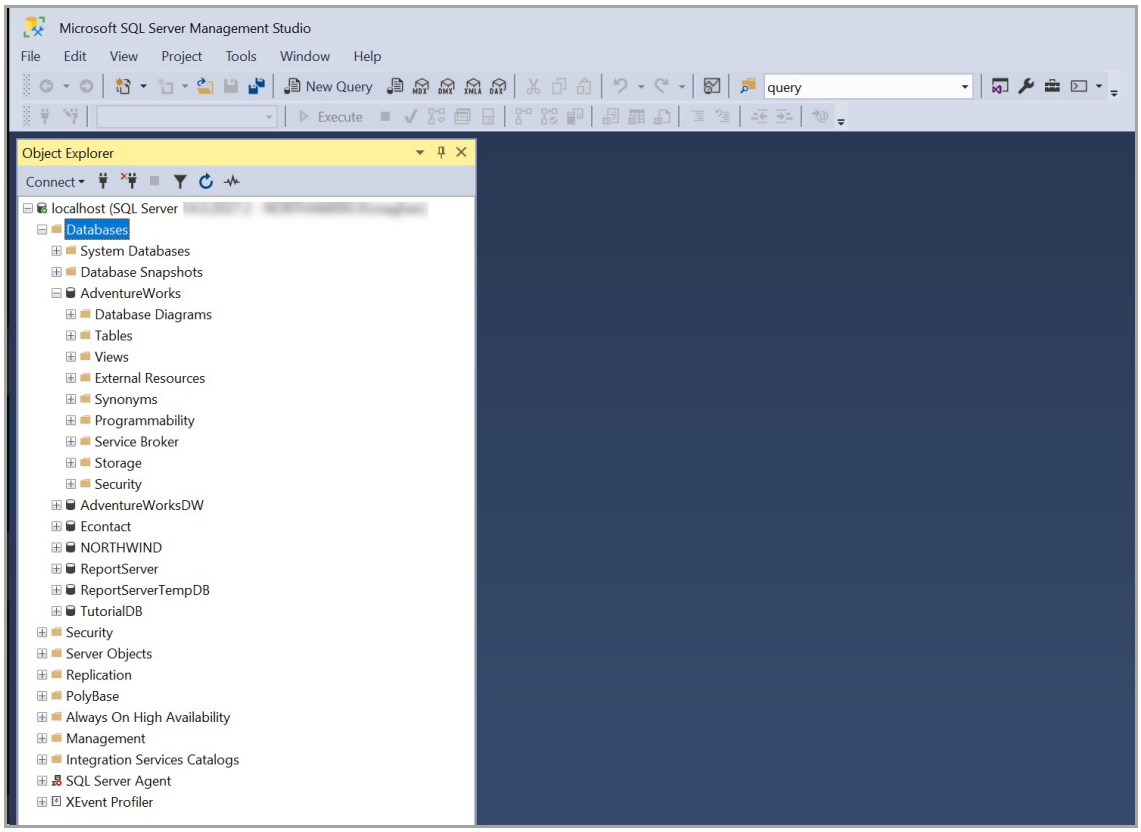

Figure 32– SSMS[38]

FIREBASE REALTIME DATABASE

In accordance with Moroney & L. (2017), the Firebase Realtime Database is a cloud-based database system that stores data in JSON format, enabling real-time synchronization and offline functionalities across various clients. This means that data is automatically updated for all connected users whenever changes occur.

As per Sonam Khedkar & Swapnil Thube (2017), Firebase, create by Google, is an API that allows the user to incorporate database storage and synchronization into the user's Android, iOS, or web application. It includes features like authentication to control authorized user access, serves as a host for deploying web content in a quick manner, it is possible to cross-platform messaging at no cost, it offers analytics for understanding user behaviour, and storage for managing media content such as images, videos, and audio directly from client SDKs. Because Firebase is a cloud-based database it doesn't require SQL queries for data storage and retrieval, being highly reliable and maintains data even when the connection is lost.

Figure 33 illustrates how Firebase operates, enabling collaboration among multiple devices in a straightforward manner.

38 https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16

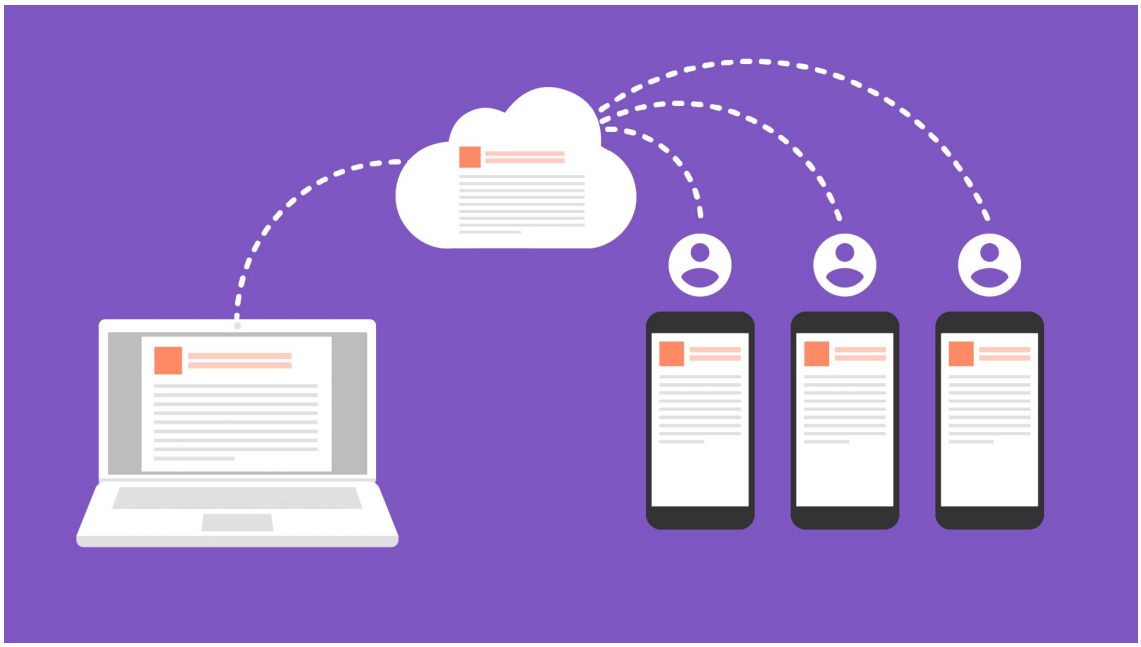

Figure 33 – Firebase Realtime Database[39]

COMPARISONS

Table 7 provides a more in-depth comparison selected databases, allowing for a more comprehensive understanding of the differences among these databases.

	MySQL	Microsoft SQL Server	Firebase
Database type	Relational Database	Relational Database	Non-Relational Database
Scalability	Difficult to scale	Difficult to scale	Easily scalable
Learning Curve	Medium	Hard	Medium
Developer	Oracle	Microsoft	Google
Data scheme	Yes	yes	Schema-free

Table 7 – MySQL vs. Microsoft SQL Server vs. Firebase

1. **Database type:**
    a. **MySQL:** MySQL is a robust relational database management system that employs structured tables with predefined schemas to store data. This structured approach ensures data integrity and consistency.
    b. **MSS:** MSS, like MySQL, is a relational database system. It relies on predefined

[39] https://firebase.google.com/products/realtime-database

schemas and key-defined relationships to organize and manage data efficiently.

c. **Firebase:** Firebase takes a different approach as a non-relational database, offering greater flexibility in data storage. It allows for data to be stored without the constraints of a rigid schema, making it adaptable to changing data needs.

2. **Scalability:**
   a. **MySQL:** Scaling MySQL can be challenging, particularly when dealing with large datasets and high traffic. Achieving scalability often requires complex strategies and can be cost-intensive.
   b. **MSS:** Similar to MySQL, scaling MSS databases can be complex and costly, especially when faced with significant data growth and high demand.
   c. **Firebase:** Firebase excels in scalability, simplifying the management of large volumes of data and high traffic. It offers easy horizontal scaling options, making it well-suited for dynamic workloads.
3. **Learning curve:**
   a. **MySQL:** MySQL presents a moderate learning curve, requiring a fundamental understanding of SQL and database principles to work effectively with it.
   b. **MSS:** Compared to MySQL, MSS can be more complex, posing greater challenges for beginners and those new to relational database systems.
   c. **Firebase:** Firebase shares a similar learning curve with MySQL but may be more accessible to individuals unfamiliar with relational database concepts due to its schema-less nature.
4. **Developer:**
   a. **MySQL:** Developed by Oracle Corporation.
   b. **MSS:** Microsoft is the driving force behind MSS.
   c. **Firebase:** Firebase is developed by Google.
5. **Data scheme:**
   a. **MySQL:** MySQL commands a predefined schema, enforcing data integrity and consistency through structured tables and relationships.
   b. **MSS:** Like MySQL, MSS enforces data integrity and consistency through predefined schemas and key-defined relationships.
   c. **Firebase:** Firebase takes a schema-less approach, allowing data to be stored without predefined structures. This flexibility is advantageous for applications with evolving data models.

3.1.8. Render vs. Heroku

When it comes to testing an application without the hassle of passing the code for every test user, utilising online platforms is recommended. These platforms provide a convenient way to deploy and test applications without the need for complex setup procedures. In this dissertation, we will explore two existing platforms that facilitate the process of hosting applications online.

RENDER

Render[40] is a comprehensive cloud platform that allows you to create and manage your applications and websites effortlessly. It comes with a big variety of features.

The user needs to choose a service type and the application is set up within seconds and maintains automatic updates. Render centralizes all services, going from web services and static sites to PostgreSQL databases, private services, and background workers.

One of Render's features is the ability to handle HTTP request with response times of up to 100 minutes. It offers native support for static sites through a worldwide content delivery network and comprehensive cron job capabilities.

Render is user-friendly, it is easy to link the user's GitHub or GitLab repository to the render dashboard, where the platform suggests commands to initiate building and launching the app.

Figure 34 shows a replica of the user's production environment.

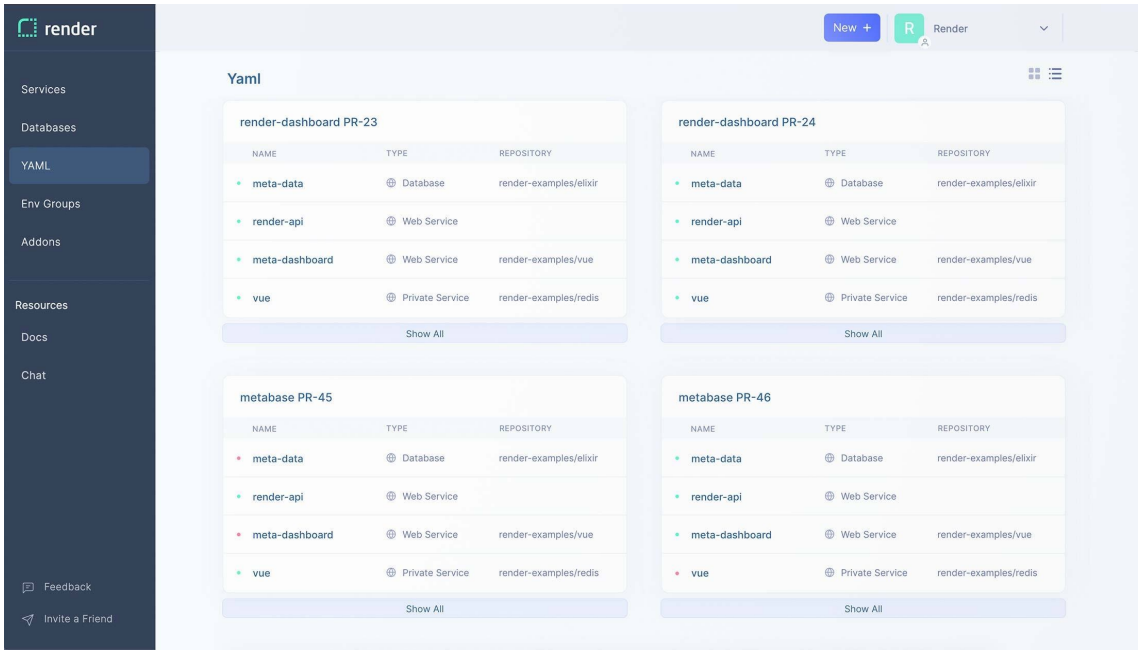

Figure 34– Render[41]

HEROKU

As stated by Chris Kemp & Brad Gyger (2013), Heroku is a versatile cloud-based platform designed for development and delivering applications. It alleviates developers from managing servers, system administration, and technology stacks, allowing them to concentrate on application development. This platform is built on virtual machines, accommodating multiple users. Management and scaling of application are facilitated through a command-line interface, online dashboard, and mobile devices, accessible from anywhere via the API. Heroku enables swift development for both

[40] https://render.com

[41] https://render.com/preview-environments

development and production apps, aligning well with agile development methods and supporting seamless continuous deployment.

In accordance with Neil Middleton & Richard Schneeman (2013), think of Heroku as a kind of web-based operating system. It allows developers to build applications like a game designer, without the need to delve into the complexities of configuring database servers or maintaining server updates. Heroku handles these technical aspects, granting users the freedom to integrate with it according to their preferences.

Figure 35 shows how Heroku works, allowing the user to monitor the metrics of a blog and receive corresponding alerts.

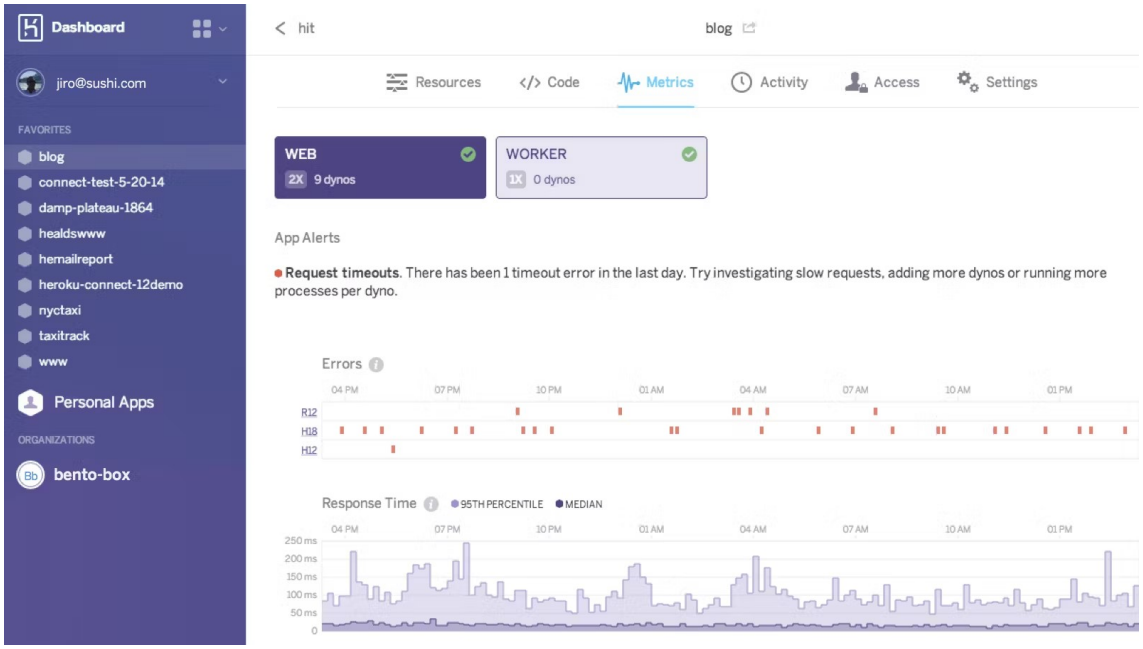

Figure 35– Heroku[42]

COMPARISONS

Table 8 serves as a comparison tool between the two distinct platforms.

	Render	Heroku
Instance RAM	512MB – $7 1GB – $15 2.5GB – $50	512MB – $25 1GB – $50 2.5GB – $250
Free Tier	Yes	Yes
Core Features	Autoscaling Managed Databases	Continuous Delivery Managed Databases

[42] https://www.getapp.pt/software/113509/heroku

	Preview Environments	Container Platform

Table 8 – Render vs. Heroku

1. **Instance RAM:**
    a. **Render:** Render provides a range of options for random access memory (RAM) in virtual instances, provide for to different budget and performance needs. Their offerings start at just $7 for the most affordable package, which includes 512MB of RAM. For users seeking more robust performance, there is a high-end package with 2.5GB of RAM priced at $50. Additionally, Render offers a middle-tier package with 1GB of RAM at a reasonable rate of $15.
    b. **Heroku:** In contrast, Heroku offers a similar range of virtual instance RAM options but at a higher price point compared to Render. The most budget-friendly package with 512MB of RAM starts at $25, which is notably more expensive than Render's equivalent offering. Heroku's top-tier package, boasting 2.5GB of RAM, is priced significantly higher at $250. The intermediate option with 1GB of RAM comes in at $50, making Heroku a pricier choice when compared to Render.
2. **Free Tier:**
    a. **Render:** Render extends the option of a free tier for users, allowing them to access limited features and resources without incurring any costs.
    b. **Heroku: S**imilarly, Heroku also offers a free tier, providing users with access to a limited set of features and resources at no charge. This aligns with Render in terms of their approach to provide to budget-conscious users.
3. **Core Features:**
    a. **Render:** Render stands out by offering robust autoscaling capabilities, empowering users to dynamically adjust the number of instances and allocate resources as needed, ensuring optimal application performance. Moreover, Render simplifies the setup and maintenance of databases, streamlining the database management process. It also facilitates the creation of isolated testing environments, enhancing the development and testing workflow.
    b. **Heroku:** Heroku excels in automating the deployment pipeline, making it a preferred choice for developers looking for efficient DevOps processes. Similar to Render, Heroku offers managed database services, reducing the burden of database administration. Furthermore, Heroku's container platform capabilities streamline the deployment and management of containerized applications, ensuring a consistent and portable deployment method. This containerization approach enhances application scalability and resource efficiency.

3.2. Conclusion and Evaluation of Alternative Technologies

In conclusion, the process of selecting the appropriate technologies for this project involves an

evaluation of various options, considering factors such as prior knowledge, alignment with project requirements, versatility, scalability, and ease of use. The decision to utilize Unity technology for the development of this project came from some factors such as the existence of prior knowledge and experience with the platform. This familiarity with Unity proved to be a significant advantage, as it gave a sense of confidence and ease during the entire application development process. Besides this, the project's objectives and technical demands were carefully assessed, and after an analysis it was concluded that Unity could cater for these needs.

Out of all the other technologies that were evaluated for the AR project, Vuforia became the most suitable choice based on its alignment with the specific requirements of the application being developed. The primary reason for selecting Vuforia over other options was its versatility in accommodating different platforms, allowing the application to reach a bigger audience.

Despite GitHub being somewhat limited compared to the other two technologies, it was chosen as the version repository due to the presence of prior knowledge of this application. Node.js was selected as the programming language of choice for the project's server due to its suitability for real-time applications. This feature holds significant importance in the context of the project being developed. The project demands that the server delivers swift responses to client requests, an important requirement for maintaining a seamless user experience. Additionally, Node.js offers high scalability, allowing it to accommodate growing user demands effortlessly.

Initially, the MSS was used for the development of the database. However, due to its scalability limitations and challenges in getting the database online, it was necessary to switch to Firebase. Firebase is easy to scale and does not impose a strict schema, which in turn simplifies data insertion into the database.

Between Heroku and Render for the cloud platform, Render was the chosen platform despite prior experience with Heroku. This decision was primarily influenced by changes in Heroku's pricing plans. Furthermore, Render offers the convenience of automatic website updates through GitHub integration, making it an ideal choice for the project's development.

The careful evaluation and selection of these technologies played an important role in the successful development of the project. The combination of Unity, Vuforia, GitHub, Node.js, Firebase, and Render provided a foundation to create an application that met its objectives.

Chapter 4

Development

This chapter delves into the complexity of the development process, focusing on functional and non-functional requirements. Furthermore, it provides a comprehensive and detailed explanation of the dissertation's solution development.

4.1. Development stages

4.1.1. Main steps

The project involves the development of an application with a fundamental goal: to enhance the quality and appeal of both virtual and physical tours. The objective is to create an engaging and immersive experience for visitors, whether they are visiting a place remotely or physically. To achieve this, some key aspects must be considered.

First, a comprehensive study must be conducted to identify and select the most suitable technologies for the project's development. This process requires a thoughtful evaluation of various technologies available in the industry. The goal is to choose the right tools and platforms that align with the project's objectives and will facilitate the desired functionalities for both virtual and physical tours.

The technology selection process involves comparing different options within a diverse range of technologies. This comparative analysis is important for justifying the selection of a specific technologies, as it provides clear feedback for why certain tools and approaches were selected over others.

The application to be created must offer dynamic and interactive features, serving the needs of visitors in both virtual and physical environments. There's a need to differentiate between these two contexts in order to choose which dynamic elements should be used.

For virtual tours, the application should be accessible through web browsers and serve as a mean of communication between visitors and support personnel. It is essential that this interface is user-friendly and intuitive, ensuring that even individuals without programming expertise can easily navigate and understand it. The goal is to provide a smooth and efficient channel for communication between visitors and support personnel.

A similar ease of communication should be extended to online visitors. The aim is to make it effortless for them to connect with support personnel, eliminating any unnecessary complexity in the process. This approach simplifies the visitor experience, enhancing overall satisfaction.

Beyond the virtual tours, the project also encompasses the development of an application to enrich physical visits. This can be achieved through the implementation of AR technology. By utilizing AR, avatars and augmented information can be superimposed onto the physical surroundings, offering

visitors additional context and engagement during their tour. This innovative use of technology can significantly enhance the visitor experience.

4.1.2. Non-functional requirements

When talking about non-functional requirements, we are talking about aspects of the project that are concerned with how the features are delivered and experienced by the users. It is important to make the prototypes/mechanisms user-friendly. Being user-friendly implies that the interface should be intuitive and straightforward. Creating something excessively complex can be counterproductive because it may discourage users from engaging with the application. Making the protocols user-friendly is about making sure that users can easily understand and use the application without encountering unnecessary complexities.

About the interaction protocol/mechanism, this refers to how users engage with the application, and it should be accessible through a web browser or any device that can run a browser. It means that users should be able to access and use the application from a wide range of devices, whether it is a desktop, tablet or even smartphone, to guarantee accessibility for every device it is important to have a responsive website.

The dynamic elements are components of the application that can change or be customized by administrators without the user of coding. To be user-friendly, it is essential that these dynamic elements are easily changeable. This accessibility for non-programmers is important because it allows for more flexibility and adaptability in responding to user needs and preferences without requiring extensive technical knowledge.

4.1.3. Functional requirements

The project's functional requirements encompass a series of tasks and objectives that serve as a roadmap for the development of the application, to ensure that the end product meets the desired standards and user expectations.

- The initial step involves a through exploration of the technologies and tools essential for prototype development. This research includes identifying the most suitable tools for prototyping.
- To gain a deep understanding of AR applications development, the student is required to complete an AR course. This course covers the fundamentals of creating AR applications from scratch, providing valuable insights.
- Extensive research into virtual tours is important. Learning how to build immersive 360º tours using images and ensuring seamless transitions between scenes it is crucial to enhance the user experience.
- Creating an AR application from scratch it is important to do before proceeding with the protype, that way it is possible to grasp the practical aspects of AR application creation.
- The same applies for the 360º application that utilizes 360º images to create immersive

tours.

- The project involves creating a prototype for virtual tours. This prototype must include dynamic elements for user interaction, such as avatars, links, images, and more. Users should be able to engage with avatars for information and guidance as they navigate the virtual tour.
- A prototype for physical tours must be created, it should include interactive elements like instructions and guided tour avatars. Users should be able to interact with the avatars for detailed information about specific locations during physical tours.
- The application must offer users the ability to easily modify dynamic elements. This feature should be intuitive, even for non-programmers.
- The application should facilitate communication between visitors and support personnel. It must be accessible via web browsers and designed for user-friendliness, ensuring a good communication between users and support personnel.
- The user interface should also enable a smooth communication between users and support personnel. Ensuring that messages can be sent without problems.
- Avatars should be associated with support users, and when a user logs out, the avatar should no longer be visible, ensuring a coherent user experience.
- A robust user authentication system must be implemented for both the user interface and the web browser, incorporating basic security measures. For this functionality a database must be created in order to ensure that only authorized users can access the application.

4.2. Solution Development

The previous chapter provided an extensive explanation of the technologies used during the development of the project, along with the initial concept and the functional and non-functional requirements. This chapter begins by delving into the modelling of the database needed to establish a connection between the server and the client. Additionally, it elaborates on the development of both the server and the client, offering a more detailed view of the processes involved.

4.2.1. Database

In the context of the virtual tour experience, clients are the primary users who engage with the tour on their preferred computing devices, such as computers or mobile devices. Should they require assistance or guidance during the tour, they have the option to initiate contact with tour guides through a support avatar. This seamless interaction is made possible through the integration of a specialized database that effectively manages various aspects of user interactions, including registrations, logins, and the exchange of messages between clients and the server.

The crucial decision in this project was the selection of an appropriate database system. The chosen database must meet several critical criteria: robust performance, scalability to accommodate potential future growth, and unwavering reliability. This decision carries substantial weight as the

selected database will shoulder the responsibility of securely storing and retrieving client data, ensuring a smooth and responsive user experience.

The choice of database technology depends on the underlying system requirements. For those favouring a structured, tabular approach to data management, relational databases (typically complying to SQL standards) are an ideal choice. Conversely, if the need arises for a more flexible, schema-less approach that is adept at handling unstructured or semi-structured data, NoSQL databases are the preferred solution. Furthermore, the complexity of anticipated queries is an influential factor. SQL databases excel in handling complex, structured queries, whereas NoSQL databases are better suited for simpler, more dynamic queries.

In the initial stages of development, a relational database was created using Microsoft SQL Server Management. However, as the project evolved and the need emerged for an online, real-time database solution that could accommodate potential scalability requirements, a strategic decision was made to transition to the Firebase Realtime Database. This NoSQL database, notable for its real-time capabilities and automatic ID generation, was deemed an ideal fit for the project's evolving needs. This transition also prevented the necessity for a dedicated column in the schema, simplifying the database structure.

Within the database design, a minimalist approach was embraced, aligning with the project's specific requirements. The database is organized around three primary tables, each with distinct roles in managing user data for different user types, including both clients and server users. Additionally, a dedicated table was established to serve as a repository for all message exchanges between users, ensuring efficient storage and retrieval of communication data.

The "guests" table is responsible for storing client-centric data, encompassing essential information such as email addresses, names (used as identifiers in the client’s chat interface), and securely encrypted passwords. These email and password credentials are important for client application logins, emphasizing data security.

In parallel, the “hosts” table is responsible for storing server user information, similarly to the “guests” table, which is responsible for storing the user’s email, name, and password. Just like the guests, hosts would use their password and email to log in. In both tables, passwords are encrypted to ensure security for their clients.

Lastly, the “message" table is a crucial component for documenting message-related data. It meticulously captures sender and recipient email addresses and associated IDs, message content, and precise timestamps. This comprehensive message storage ensures seamless organization and retrieval of chat messages exchanged between clients and guides, enhancing the overall tour support system's efficiency and reliability.

The detailed design that was discussed earlier can be seen in Figure 36.

Figure 36 – Database

4.2.2. Server

The project requires a server for communication between the support personnel and the platform users, as well as for making HTTP requests, data storage, and retrieval. When selecting a server-side programming language, we need to consider the following characteristics that can influence development and maintenance:

1. **Performance:** Different programming languages offer varying performance levels. Some are optimized for high-performance tasks, while others might lead to overhead. We need to choose a language that aligns with our project's speed and load requirements. If we want to respond more quickly to the requests of our users and be able to manage several things at the same time, we must choose a language that provides such functionality.
2. **Scalability:** Continuing the issues of traffic, certain languages support either horizontal (adding more servers) or vertical (improving existing server features) scalability. We need to choose based on our project's traffic demands.
3. **Integration:** In addition, it is advised that exists a compatibility between the server and the client language for a more simplified integration.

For this project, the chosen server-side language is Node.js. Node.js excels at handling numerous connections without resource overload by using a single thread for asynchronous connection handling. This design offers high scalability and efficiency. Additionally, Node.js offers fast execution, resulting in reduced latency and efficient code processing. It is highly recommended for real-time applications like chat functionality, aligning with the project's needs.

Node.js is compatible with various platforms, including Windows, Linux, and MacOS, enhancing the application's accessibility for users.

To initiate the server development process, several essential steps were performed. Initially, a folder was created to serve as the host for the server. Following that, it was necessary to run the command "**npx express-generator –ejs**". This command generated the fundamental skeleton of a Node.js application, incorporating the EJS template engines. Consequently, he following directories and files were created: **node_modules, public, routes, views, bin**, as well as the **app.js**, **package-lock.json**, and **package.json** files. In this project, the "**bin**" folder was not needed and has been replaced by the "**config**" folder, which contains the settings for the online database.

The way the project is organized can be observed in Figure 37.

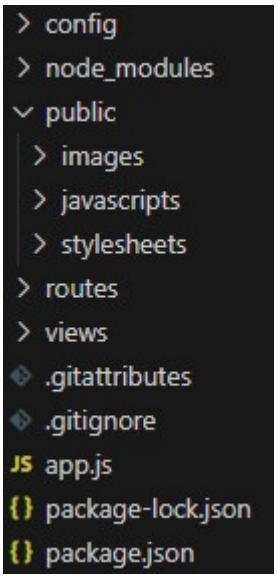

Figure 37 – Structure of the project

The ”**node_modules**” folder plays a crucial role by storing the libraries, packages, and third-party dependencies needed for the project. As dependencies are installed to enable the project functionality, they are organized within this folder. Similarly, the “**package.json**” file contains vital information about the project, including its version, dependencies, and other relevant data. Given that the folder can expand significantly depending on the dependencies of the project, it is recommended to include this folder in the “**.gitignore**” file. This ensures that when committing to GitHub, it does not come with dependencies. This practice helps prevent performance issues, conflicts, or even unnecessary history.

The “**public**” folder contains the project’s static files such as HTML, CSS, JavaScript, images, and other similar resources. This folder can be seen as the client interface, containing the elements that the user observers in the browser, without having any kind of processing about the server. This type of design allows you to put the server code and the most sensitive data separate, providing a certain security by reducing potential attacks that target static resources.

Meanwhile, the “**routes**” folder contains the mechanisms necessary to respond to specific URLs. That is, each route will determine the processing procedure, more specifically, each route encompasses HTTP methods, such as GET, POST, PUT and DELETE.

The “**views**” folder contains the presentation layer. Everything the user perceives is inside this folder. This folder usually has templates that contain the HTML/EJS markup, which serve as allocators for dynamic data and logic that will be processed by the template engine.

The “**app.js**” file serves as a startup file for the application, containing code to configure the project, such as the dependencies that will be imported, the startup of the application, the definition of routes, the start of the server and the handling of errors. It can also contain settings to handle login, security, and authentication.

The architecture to be used on the server consists of the Client-Server architecture. This type of architecture divides the system into two parts, one part is responsible for interacting directly with the user. In more detail, this part can run on several devices and is responsible for dealing with the user interface, receiving inputs from the same and sending to the server, in addition it also receives the

response from the server, and it must be displayed for the user to be able to observe. That process can be observed in Figure 38.

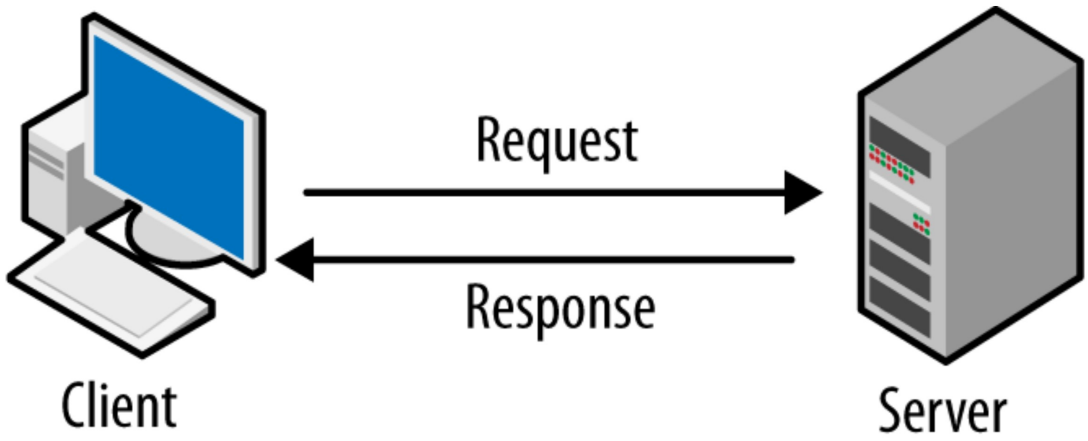

Figure 38 – Client-Server Architecture[43]

For the server part, it is responsible for managing the resources, processing user requests and responding to them. Usually, the server handles the database and services that the client needs. The server must receive requests from customers, interpret them and process them according to the appropriate way, in addition to that it is also in charge of handling business logic, application security and data storage.

This type of architecture can be beneficial, because it allows a scalability on the part of the server independently of the client, that is, it is possible to change the number of servers to deal with the growing number of clients, in addition to that the storage and management of data by the server can improve the integration of the data and its consistency. The Client-Server architecture represents an easy-to-manage architecture because you can change either side without affecting the other. The division of the project is carried out between the "views" and "public" folders on the client side, and the "routes" folder on the server side.

AUTHENTICATION

There are two types of users, one representing the client user and the other representing the server user. To make the reading more eloquent, I will refer to these users as the “client” and the “server” respectively.

To take advantage of the application, whether as a client or a server, the user must have a registered account in the database. Otherwise, access to the application will be denied. The required information for registering a new account in the database includes the user’s name, which may serve as an identifier in certain functionalities, their email address, and the password.

To securely register a new user, the **bcrypt** module is used. This module hashes passwords securely, preventing brute force attacks, thereby enhancing user protection. The **bcrypt** module

[43] https://madooei.github.io/cs421_sp20_homepage/client-server-app/

possess various features, such as salting, which involves adding a random value to the password value before hashing. Since each password has its unique salt, precomputed table attacks have a difficult time. Additionally, the cost factor can be adjusted, essentially determining the intensity of the hashing process based on user needs. It is important to note that a high-cost factor slows down the hashing process but significantly increases the difficulty and time required for brute force attacks. In Figure 39, you can see how a brute force attack works. It begins with a hacker who employs an automated tool to carry out numerous attempts on a victim's computer or system. The hacker's goal is to gain unauthorized access to information that they are not supposed to have access to.

Figure 39 – Brute force attack[44]

Brute force attacks involve systematically attempting various combinations of characters, progressing from simple to complex, in an effort to guess the encryption key, in this case, the password. On the other hand, a precomputed table attack, commonly known as a rainbow table attack, represents a more sophisticated method to obtain passwords. This attack involves creating and using precomputed tables that store the results of various hashes.

Figure 40 illustrates how a rainbow table attack function. Rainbow tables are created through a series of hashing and reduction operations.

[44] https://www.simplilearn.com/tutorials/cryptography-tutorial/brute-force-attack

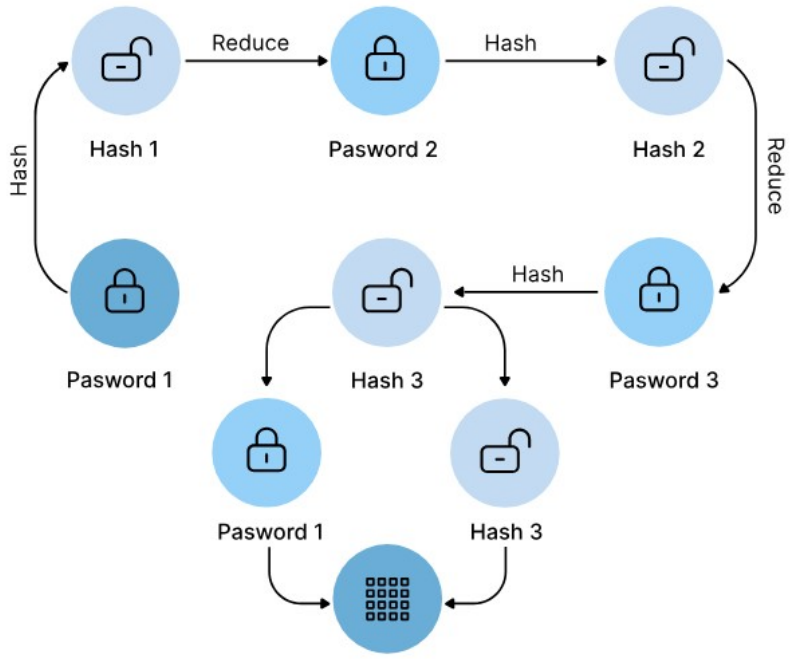

Figure 40 – Rainbow table attack[45]

The hash process of the **bcrypt** module includes generating a random salt, combining it with the password value, and the hashing this combined value. The **bcrypt** module applies the hash function to this value multiple times, with the number of iterations chosen by the user. The resulting hashed value is then stored in the database.

Returning to the registration of a new user, connecting to the database is essential to add the new user. Initially, it is necessary to distinguish whether the user is from the server of the database, due to the existence of two different tables. Following this verification, a SELECT query would be used to search the database but, on this case, I'm using the Firebase Realtime Database, so I look with the reference for users with the same email. If such a user exists, the user is informed, as duplicate email addresses cannot be allowed. If all validations pass, the user's entered data is inserted into the database, now with the password encrypted. The user is then notified of the successful operation.

The login process follows a similar procedure, requiring verification of the user type to perform a search. If no user is found, the user is informed and prompted to register. If a user with that email already exists, a password verification is performed by comparing the password passed by the user and the encrypted password stored in the database. This verification uses the '**compare'** method from the **bcrypt** module, which takes the unencrypted password and the encrypted password as arguments. If the user is valid, their information is stored in a variable, for the users from the server the user data is stored in the browser's session storage, for later use in other functionalities.

Some validations are performed for these two functionalities, such as: it is verified if the email is a valid email, if the name and the password are empty fields or not. For the server, if the login is successful, they are sent to the homepage, for the client they are sent to the virtual tour. From the

[45] https://nordvpn.com/pt/blog/what-is-rainbow-table-att ack/

homepage the user can access the chat or logout.

The login design can be seen in Figure 41.

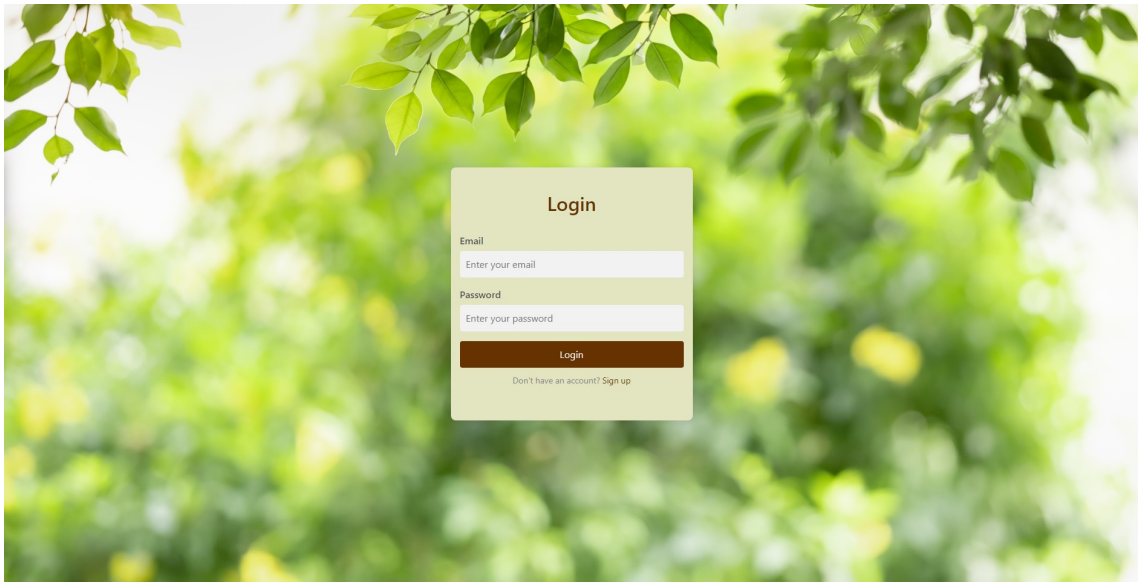

Figure 41– Login

CHATBOX

To access the conversation window, the user needs to be logged in. When the user clicks on the conversation option, a validation is performed to ensure that the user session has not been closed. This validation involves verifying whether the session variable is empty or not. If the variable is empty, the user will be redirected to the login page to log in. This step is essential to prevent unauthorized access and to avoid any unauthorized operations. By taking this measure, it is possible to maintain data integrity and consistency, and not only is it a strong security measure, but it also serves as effective error prevention.

The chat design is quite simple and intuitive. When the user opens the page to initiate a conversation, the first thing they see is a rectangle with two sections. One section displays the clients who are logged in, along with the latest message between the server with the active session and the client in question, along with the date of that message. The other section represents the user's chat. In this section, the message history will appear, along with the form for sending new messages. The information regarding users with active sessions and messages is immediately transmitted upon the page's opening.

In Figure 42, it is possible to observe how the chat functionality looks without any user logged in.

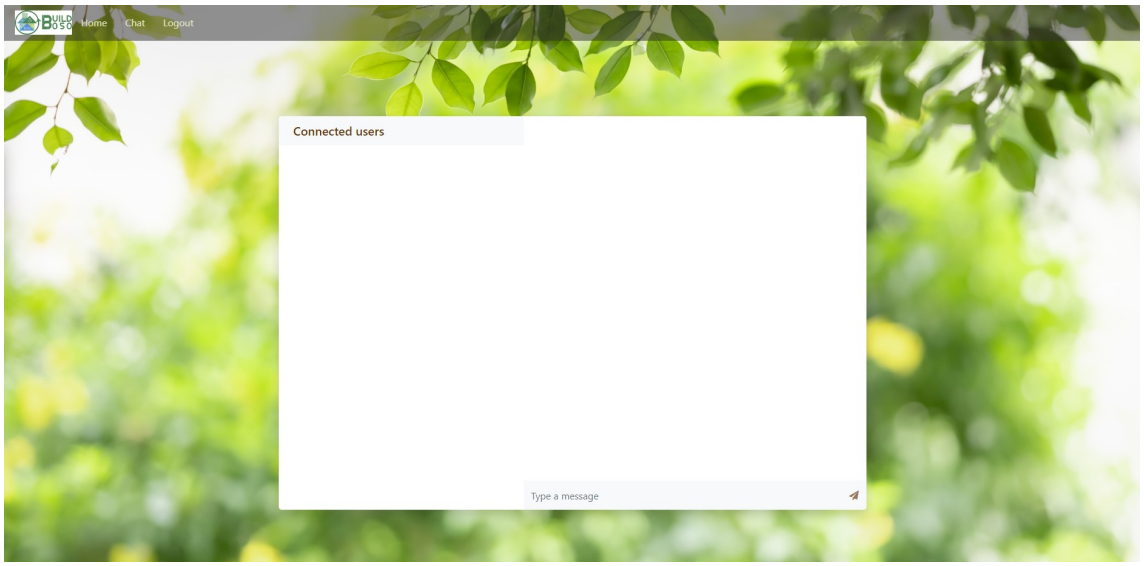

Figure 42 – Chat without people

For the proper functioning of this feature, two functions are invoked. One function is designed to retrieve the latest message between the server with an active session and the clients. This function takes the user's email and their name as parameters. With this information, a search is conducted in the database, specifically in the messages table. However, prior to proceeding to this step, it is necessary to determine the user type, distinguishing between a client and a server.

Following that, two searches are performed. The first search collects all messages where the user is the recipient, and the second search gathers all messages where the user is the sender. These actions involve querying the messages table to identify messages where the "sender" matches the user's email and messages where the "receiver" matches the user's email. The messages meeting these criteria are stored in separate variables.

Later, these messages are iterated to verify if the recipient is different form the sender. If that is the case, the messages should be saved, thus storing the most recent messages. The messages are arranged in descending order according to the date of sending. A "set" is created, which is a structure like lists and arrays, but it does not allow repetition of elements. This feature makes it perfect for message storage. This "set" is used to make tracking messages between two users possible. Subsequently, the variable containing the latest messages must be iterated, and each message will be assigned a key to distinguish itself among messages, or more specifically, to identify the conversation between two participants. If a message does not have a key, that key should be added to the created set, and the message should be stored, ensuring the storage of the last sent message. These stored messages are organized in ascending order based on their sending date and should be returned at the end of the function.

In addition to retrieving the latest messages with the users, it is also essential to fetch the users who are currently online to display them. To achieve this, it is necessary to check whether the array that stores the clients is empty or not. If the array is populated, these users should be stored in variable, allowing access to them later. It is important to invoke the previously mentioned function to retrieve the recent messages involving these users.

Users who are present in a list need to be mapped so that it becomes feasible to search for the latest message where they are either the message recipient or the sender. If a message meeting this criterion exists, it should be stored and subsequently retrieved to be displayed to the client. This identical process is executed for each user, enabling the user and their most recent message to be observed in the conversation window.

To be able to observe the latest new message sent by the client, it is necessary to utilize the appropriate route to manage this situation. The process of retrieving the last message begins by storing both the server and the client, these variables containing data will be used at a later stage. A connection must be established with the database to create a reference to the messages table. Once this reference is established, it becomes crucial to organize the messages by their sent date. This organization is essential to determine the latest sent message.

After the messages are organized, a brief validation process is conducted. If the message sender is the server and the recipient is the client, or vice versa, the goal is to retain the most recent message between them. However, to discover if a message itself is the last one exchanged, another validation process is executed. If no messages have been stored previously or if the timestamp of the message under analysis is more recent than the timestamp of the stored message, then the new message must be stored. This process iterates through the entire array of messages, ensuring each message is assessed.

In Figure 43, a representation of how the design of the chat functionality looks without a user selected to start a conversation is observed.

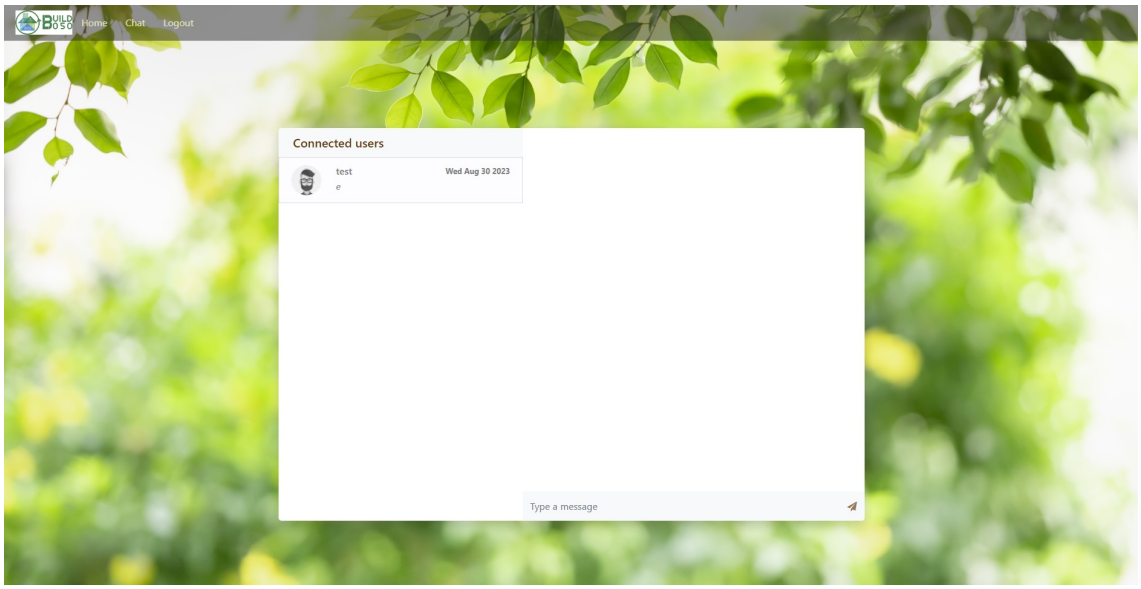

Figure 43 – Chat without selection

The user can load conversations with other users, which essentially maintains a message history with each user. This functionality enables the observation of message exchanges that have taken place previously. To load these messages, the user must initiate the conversation window with a specific user by clicking on that user's name. Once this action is performed, the historical message retrieval process is activated.

This process requires to save both the sender's and the receiver's email addresses. This step is crucial for distinguishing between users during the message display. Since we are interacting with the message table within the database, the first step involves establishing an initial connection with the database. Following this, a reference to the message table is created, and the messages are organized chronologically based on their sent timestamps.

The messages are stored in an array, specifically containing their content along with the user type. This array of messages is then iterated through, and two validations are conducted. The first validation checks if the sender corresponds to the server and if the recipient corresponds to the client. In this scenario, the user type should be labelled as 'ejs'. The second validation checks the opposite condition: whether the sender corresponds to the client and the recipient corresponds to the server. In this case, the user type should be labelled as 'unity.' After these validations, the obtained data is stored within the array of messages.

There is a route like the conversation history, but that route is specifically created for the client. In this route, there is no need to store the user type, only the client's messages need to be retained.

If the users wish to utilize the feature of sending messages to other users, they can input the message content and select the option to send the message. The process behind sending the message involves extracting various pieces of data related to the message. This includes the message's content itself, the user information of the sender, and the user information of the intended recipient.

To obtain the information of the user who will send the message, we can access the session variables that were previously created. As for the user who will receive the message, this data is stored after the user to be approached has been selected by the sender.

It is necessary to establish a connection with the database to search for the user who will receive the message. This search can be conducted in either of two tables: the client's table or the server's table. The choice of table depends on the sender. If the email of the avatar accompanying the message is empty, it means that the sender is from the server. Conversely, if the email is filled, it is indicated that the sender is from the client and intends to send a message to the selected avatar.

Should the user be in either of the tables, their information needs to be stored for later use. This enables indirect manipulation of the data, avoiding direct referencing.

To conduct the preceding process, it is essential to utilize a reference to the user table. For the subsequent step, a reference to the message stable is required. The new message needs to be incorporated into the database. This entails utilizing the previously established reference to establish a connection with the messages table. The content of the message must encompass all the antecedent reference data the message data alongside the user data.

Lastly, a push operation must be executed on the database to effectively insert a new object.

Previously, I discussed all the necessary routes to make the application functional. However, on a server, in addition to routes, there are also files that function as intermediaries between the user interface and the routes. These files give life into every action taken by the user. In other words, when a user clicks a button, it is these files that determine what actions the server should perform.

To enable the mentioned functionalities, specific functions have been developed. One of these

functions aims to load the message history between two users. The process of this function begins by initially receiving the client's data, which has been chosen by the server to display the message history between them. This received data is then stored in variables for later use. Since the server is selecting a client from the list of connected users, it is necessary to remove the selection of the previously chosen user. This selection status can be identified by the colour difference between the user in question and all other users.

After the previous selection has been deselected, the currently chosen user needs to be marked as active, resulting in a change of colour. Now, with the conversation opened, it should be possible to send messages to the user. In other words, the messaging form should become active and usable. Additionally, the messages exchanged between users need to be displayed. This is achieved through a route that distinguishes between user types, namely 'ejs' and 'unity'. By utilizing this route, an array of messages is returned, allowing us to determine the user type. Depending on the user type, the colour and alignment of the message need to be adjusted. For instance, messages from users of the 'ejs' type should appear in green on the right, while messages from users of the 'unity' type should appear in grey on the left. This differentiation ensures that it is easy to identify the sender of each message.

An important consideration is that the messages are contained within a scroll view. Consequently, it is crucial to keep the scroll positioned at the bottom to instantly view new messages as soon as they are sent.

In Figure 44, it is possible to notice how the design changes when a user is selected, how the user on the "connected users" tab gains colour and the conversation between the two users is showed on the right side.

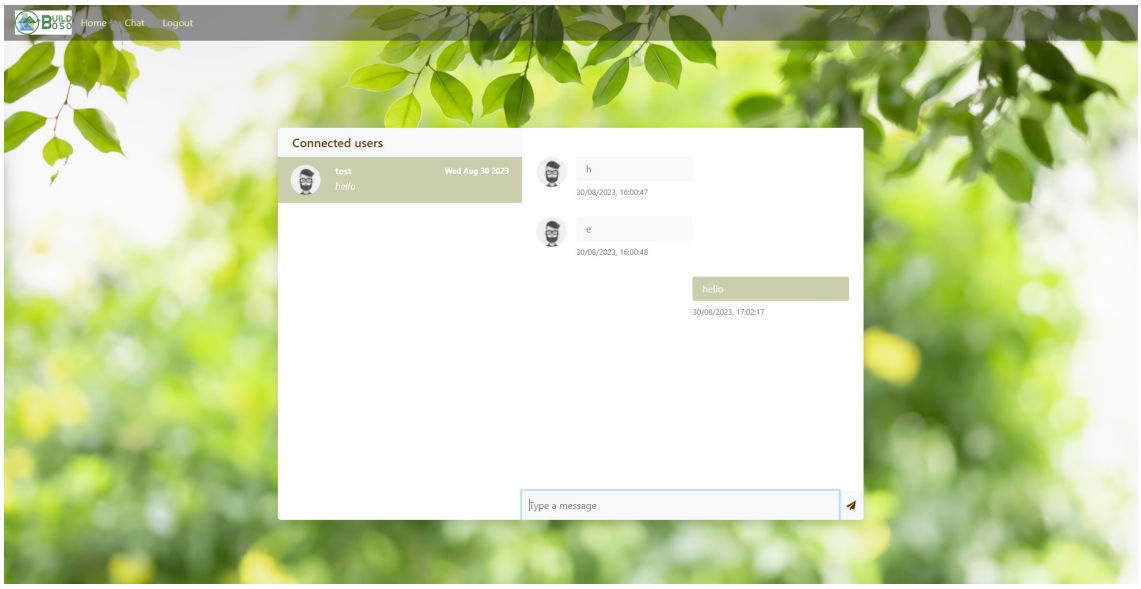

Figure 44 – Chat with selection

When the user sends a message using the form, the route responsible for adding new messages is invoked. The message is then transmitted via the content present within the form.

The functionality of displaying the latest message when a user sends a message or updates the

list of users upon their login can be achieved through the implementation of a listener. A listener is a function that awaits an event in order to respond to it. An analogy in real life would be like having a conversation with someone in a foreign language and having an interpreter who listens to one person and then translates the message for the other person. For example, on person speaks Japanese, the interpreter listens and then translates the message into English for the other person to understand. Similarly, the listener in this context receives a message and proceeds to display it. This involves invoking a function that checks whether an ongoing conversation exists. If so, the new message is displayed within the conversation window, following the same design as the previous messages.

Returning to the listener, when the message sender is the client, a function is triggered to retrieve the latest message. This allows the message to be displayed in the section dedicated to the connected users. The retrieval of the last message is facilitated by the route responsible for handling recent messages, as mentioned earlier. On the other hand, if the sender is not the client and the message type is an “update”, the size of the array that hold the number of clients is analysed. If there are no connected clients, the conversation window is closed. The same action is taken if the client with whom the server was communicating has logged out. Contrarily, if there are connected clients, the list of users is simply updated accordingly. This mechanism ensures a coherent and dynamic display of messages and user updates within the application.

4.2.3. Virtual Tours

When it comes to implementing the server, it becomes crucial to also develop the client that will seamlessly interact with the functionalities established on the backend. The central focus of this application revolves around the concept of virtual tours, providing users with the ability to explore 360-degree images. This concept shares some similarities with Google Maps but exhibits certain unique features. One key distinction lies in the simplified navigation method compared to Google Map’s Street View. Additionally, within these virtual tours, avatars are strategically placed in the scenes to offer guidance and assistance to clients as they embark on their immersive journeys.

DESIGN

The initial step to proceed is the creation of the application’s design. There are two types of designs involved: one representing client authentication and the other representing image navigation. Just like on the server, for users to enjoy their virtual tours, they need to create a new account if they haven’t already. The design for the login/registration match that of the server, with the primary colours being green and brown. These colours were chosen because they are related to the project’s theme, “Build2050”, which aims to contribute to a more sustainable future. Given this theme and the fact that the project logo incorporates the colour green, symbolizing nature, health, and vitality, the chosen colours for both the website and the authentication section align with this theme.

The authentication section, both on the client and server sides, features a background image of tree leaves, which serves to connect the user with the environment and nature. The background colour of the login/registration section is a pale green, chosen to evoke feelings of growth and

freshness, aligning with sustainability. Moreover, it provides comfort to users. The colours of the buttons and text are in a brown shade, representing tree bark. This choice creates a good contrast with the background and ensures readability. This colour tone suggests stability, reliability, and a connection to the Earth. Additionally, some texts are in grey, striking a balance between readability and subtlety. When used for body text, it lends a professional and clean appearance to the interface.

The navigation component of the interface consists of a spherical element with a 360-degree image serving as its background. Each scene within this environment offers a distinct method of navigation. In other words, depending on the specific scene, the navigation button appears in different locations. Furthermore, there are avatars present in certain scenes to enhance the user experience.

The navigation button is symbolized by a door, which represents a transition point between scenes. This choice of symbol is grounded in its familiarity and its inherent intuitiveness. In our everyday lives, we use doors to move from one room to another, and similarly, in this virtual visitation context, the door symbolizes the transition between different scenes.

Within the functionality of scene transitions in the project, there exists a script attached to the scene transition icon. This script has the role of conveying information to the SceneLoader and change the design of the mouse. The script receives the index of the currently open scene, discerns whether the transition button corresponds to the subsequent scene or the preceding one, and reference the GameManager, if applicable.

This script comprises three distinct functions, each of which is intricately linked to mouse interactions. First, there is a function primarily dedicated to altering the cursor's appearance. When the mouse pointer moves away from the transition button, this function is responsible for resetting the cursor type to "normal", changing the design.

Secondly, there is another function that is specifically designed to detect when the mouse pointer hovers directly over the transition button. In this scenario, it promptly transforms the cursor type into "clickable", signifying to the user that this element is interactive and can be clicked upon.

Finally, the third and perhaps most important functions takes charge of managing the scene transition process itself. Upon detecting a mouse click on the transition button, this function effectively loads the subsequent scene, more specifically it calls the function from the SceneLoader that contains the logic of the transition to load the next scene. It passes the GameManager, if one exists, and conveys a Boolean value that determines whether the transition should lead to the previous scene or proceed to the next one.

In Figure 45, it is shown how the sphere, representing the image to be seen, is created.

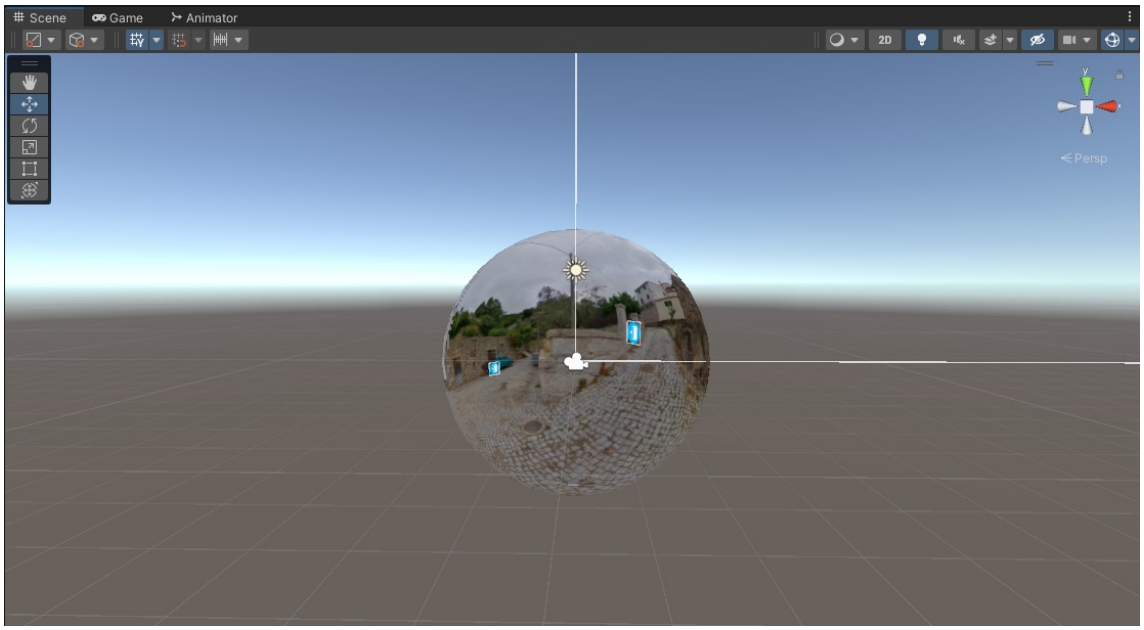

Figure 45 – Navigation design

In addition to the previously mentioned aspects, it is important noting that in every scene, expect during the authentication process, an icon is positioned in the upper right corner of the interface. This icon facilitates user logouts from the application.

What makes this icon particularly noteworthy is its widespread presence in various devices, such as television remote controls and computers. Its significance lies in symbolizing the fundamental action of powering on and off a device, a concept universally understood by users worldwide.

The icon itself features a distinctive red, carefully selected to grab the user's attention. Red is a colour often associated with stops and alerts, symbolizing the termination of the program.

When a user clicks on the logout button within the application, a specific interface related to the logout functionality is activated. First, the screen starts to darken, drawing the user attention to the logout process. Simultaneously, a rectangular dialogue box appears on the screen. This dialogue box serves the purpose of seeking confirmation from the user regarding their intent to terminate their session. This confirmation step if crucial because it prevents accidental logouts, which can be frustrating for users.

Within this dialogue box, the user is presented with two distinct buttons, each with its own colour and labels. These buttons are designed to provide a clear and intuitive choices to the user.

The first button is prominently coloured green and is labelled "Yes". This choice is intentionally associated with the colour green, which is commonly understood as a symbol of affirmation or confirmation. When the user clicks on this "Yes" button, it signifies their consent to log out. The system then proceeds to terminate the user's session. Specifically, it erases all session-related data and redirects the user back to the initial login window.

On the other hand, the second button is distinctively red and labelled "No". This button contrasts with the green button and serves as a visual cue for the opposite action – rejection or cancellation. If the user clicks on this "No" button, it signifies that they do not wish to log out after all. In response, the logout interface is closed, and the user is returned to the normal application, as if the logout action

was never initiated.

In Figure 46, a representation of what the scene where the user logs in looks like, it is possible to observe the logout button together with the button to change scene.

Figure 46 – Scene transaction button and logout button

In Figure 47, it is possible to observe how the logout popup looks like.

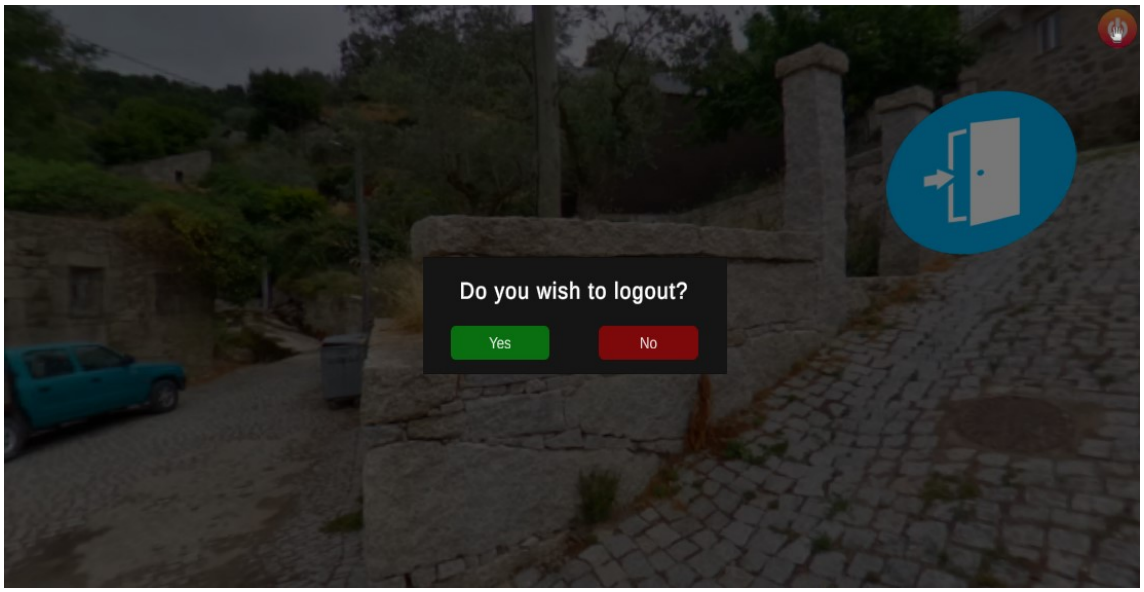

Figure 47 – Logout interface

AVATAR

Two types of tools were used in the creation of avatars. The initial program used for creating these avatars was “MakeHuman”. As stated by Briceno et al. (2018), MakeHuman is an open-source software with the primary purpose of generating realistic 3D humanoid models. Developed using the Python programming language, it boasts an intuitive graphical interface, making it user-friendly. This

interface allows users to manipulate various aspects of their avatars, such as gender, age, weight, height, and more.

MakeHuman provides support for various platforms, including Windows, macOS, and Linux. Additionally, users have the option to contribute to the available assets by uploading their own creations or downloading assets and integrating them into their projects. However, after conducting several tests, the final decision was not to use this platform. This choice was made because the avatars it offers do not provide a wide range of diversity and do not align with our project's objectives.

In Figure 48, a representation of what the MakeHuman interface looks like is shown. On the left the user has the option to change the details about the avatar; on the centre the avatar to be created is showed; and on the right it shows the category of the avatar.

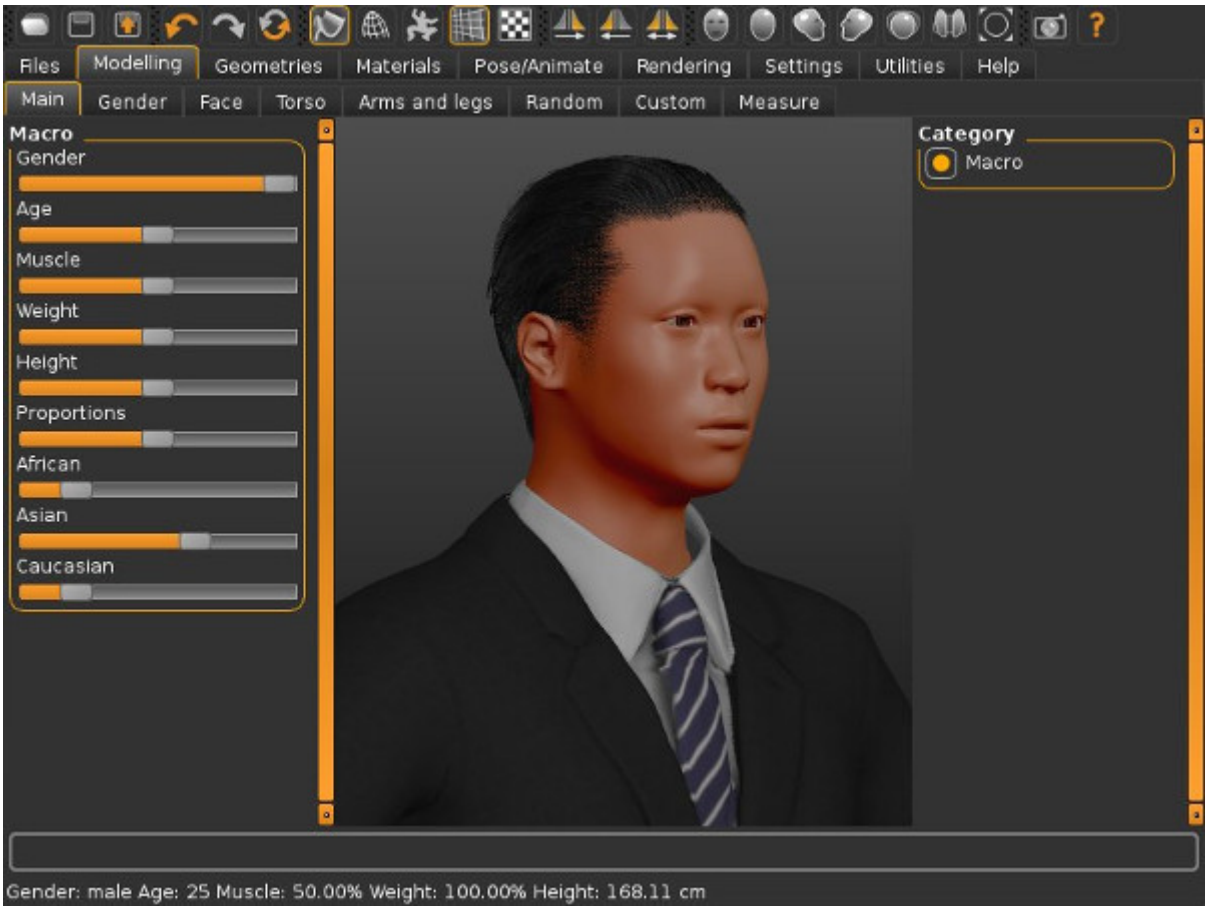

Figure 48 – MakeHuman interface[46]

"Ready Player Me" is a platform designed with the purpose of assisting in the creation of 3D models/avatars. It allows its users to customize their avatars according to their preferences, giving them control over aspects such as clothing, body, hair, accessories, and more.

Like MakeHuman, Ready Player Me offers compatibility with VR, AR, and gaming platforms. Furthermore, it integrates seamlessly with the animation platform Mixamo, which it will be discussed further later. The platform is incredibly user-friendly and simple to use. Due to these features, it was

[46] http://www.makehumancommunity.org/wiki/Documentation:What_is_MakeHuman%3F

chosen as the platform of choice for creating avatars.

Initially, two avatars were created, corresponding to two admin users. To be as inclusive as possible, both a male and a female avatar were developed. However, it is important to note that the current updated version of Ready Player Me does not include the FemaleAnimationTarget and MaleAnimationTarget, which are necessary for Mixamo.

To overcome this, it was required to download both the FemaleAnimationTarget and MaleAnimationTarget from the Ready Player Me website[47]. These animation targets play a crucial role when using Mixamo.

Figure 49 shows what the user sees when he logs in into the ReadyPlayerMe website. On the left he can access his avatars and in the centre he can see the last avatar that he interacted with.

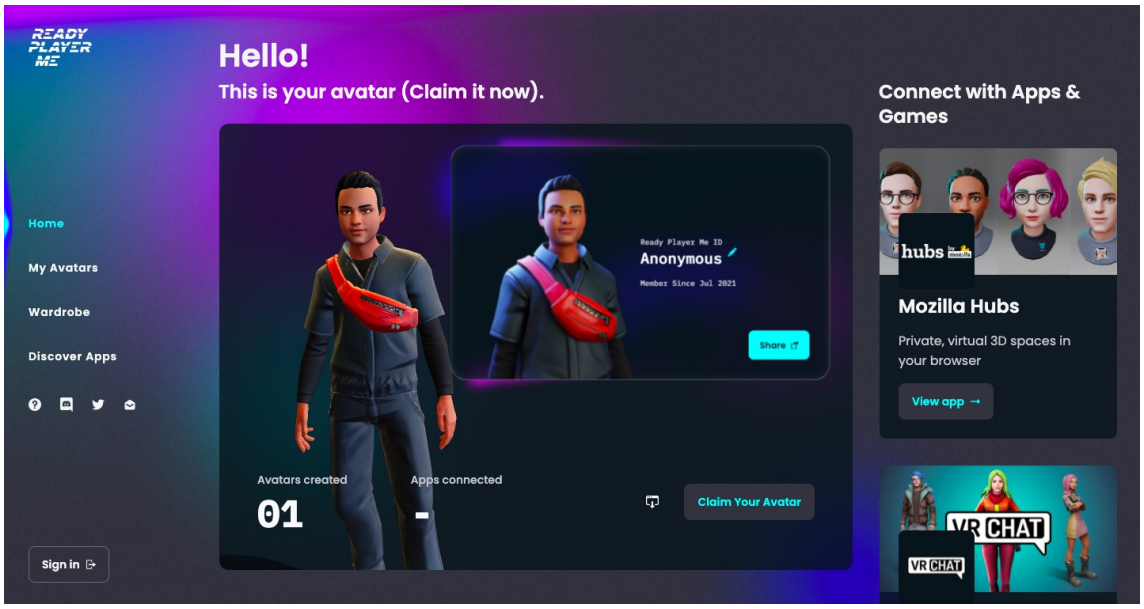

Figure 49 – Ready Player Me interface[48]

Mixamo is an online platform that offers a wide range of animation assets for 3D avatars. Users can upload their avatars and select their desired animations. In some cases, to ensure proper animation execution, it may be necessary to assign specific skeleton parts to the avatar. Additionally, Mixamo allows users to adjust timing, speed, and even combine animations to create unique sequences. Mixamo integrates seamlessly with Unity3D, making it a popular choice when implementing animations for avatars in Unity3D-based projects.

Has it is possible to observe in Figure 50, on the left the user can selected the preferred animation, that will be see on the right side, where the avatar is.

47 https://docs.readyplayer.me/ready-player-me/integration-guides/unity/animations/loading-mixamo-animations

48 https://vrscout.com/news/ready-player-me-vr-avatar-hub/

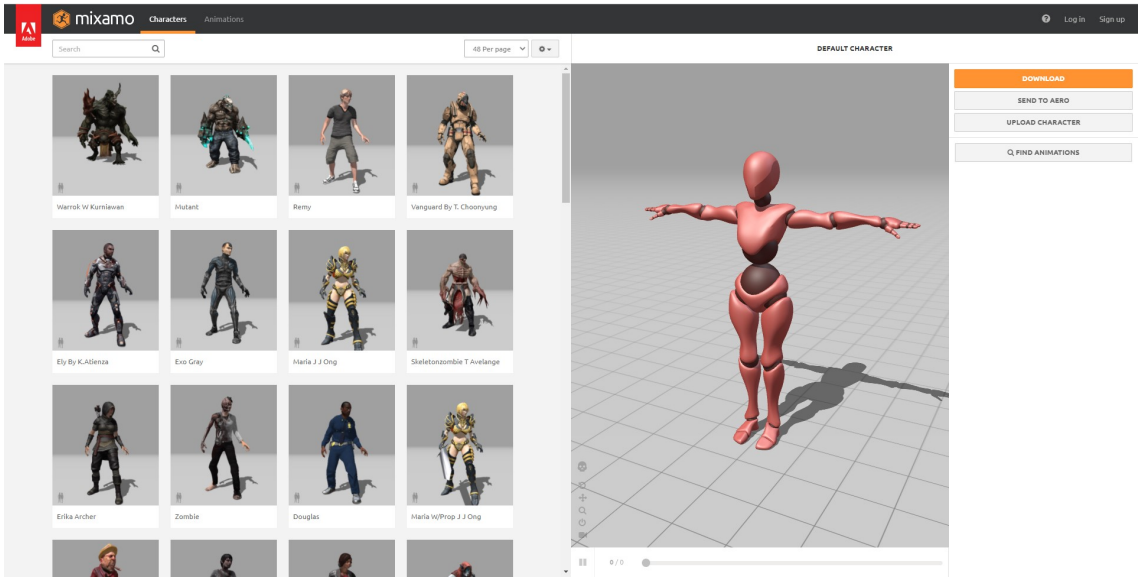

Figure 50– Mixamo[49]

To make use of the Ready Player Me platform, users must first install the Software Development Kit (SDK) to load their avatar into the scene. Once the SDK is successfully installed, the loader is initialized, and it becomes essential to input the URL of the avatar that the user created. This URL can be obtained from the platform itself. Subsequently, all that remains is to load the avatar.

These avatars will server for more than one purpose. They will be used for static dialogues with the avatars besides the dynamic interaction.

When the Ready Player Me SDK is installed, the popup of the Figure 51 will appear for the user to put the URL of the selected avatar and the option to load will be available.

[49] https://www.ketra-games.com/2021/04/getting-started-with-character-animation-using-mixamo-unity-game-tutorial.html

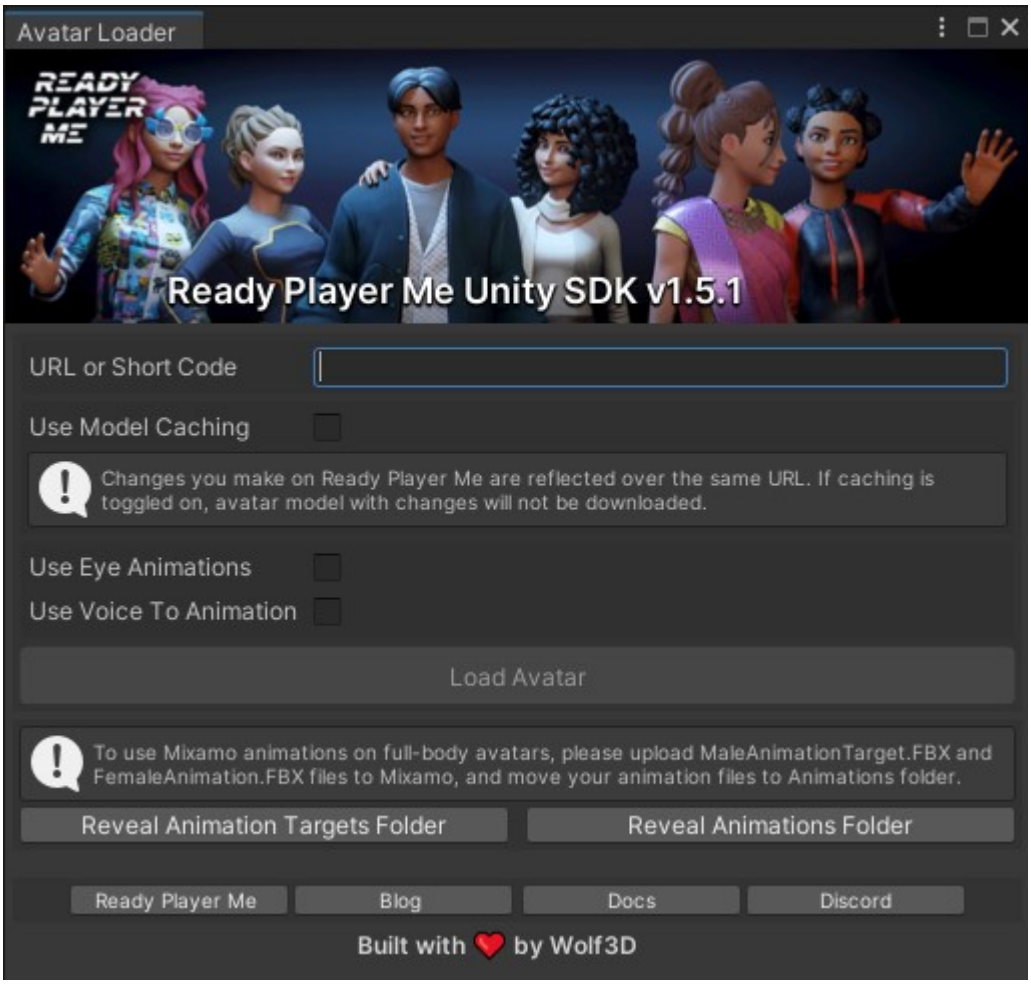

Figure 51 – Ready Player Me avatar loader[50]

The Unity animation process encompasses the creation and control of animation within an application, breathing life into objects, whether they are 2D or 3D. It is important to understand that there are various components associated with Unity animations, each serving a unique purpose. Here are some of the components:

- **Animator**: The animator is a critical component that enables the management and control of animations. It provides the means to create and orchestrate transitions between different animation states based on various criteria. This component essentially acts as the conductor of the animation orchestra, dictating when and how animations should play.
- **Animation cli**p: Animation clips are used to define specific animation sequences. They outline how an object's state changes over time. These clips serve as the building blocks for user's animations, specifying the exact motion or transformation an object undergoes during a particular animation.
- **Animation controlle**r: While the animator handles individual animation states, the animation controller takes on a broader role. It is here that the user defines the

[50] https://docs.virbe.ai/deploy-being/unity/ready-player-me-tutorial

overarching logic of his animations. This includes setting up transitions between animations based on triggers and conditions. In essence, the animation controller rules the entire animation flow and logic within the user's project.

- **Animation event**s: animation events are tools that allow the user to trigger specific actions or functions during an animation's timeline. These events can be strategically placed within the user's animation sequences to synchronize certain actions or behaviours with key moments in his animations. Theu are invaluable for creating interactive and dynamic animations.

In Figure 52, it is possible to observe how the unity animations are organized.

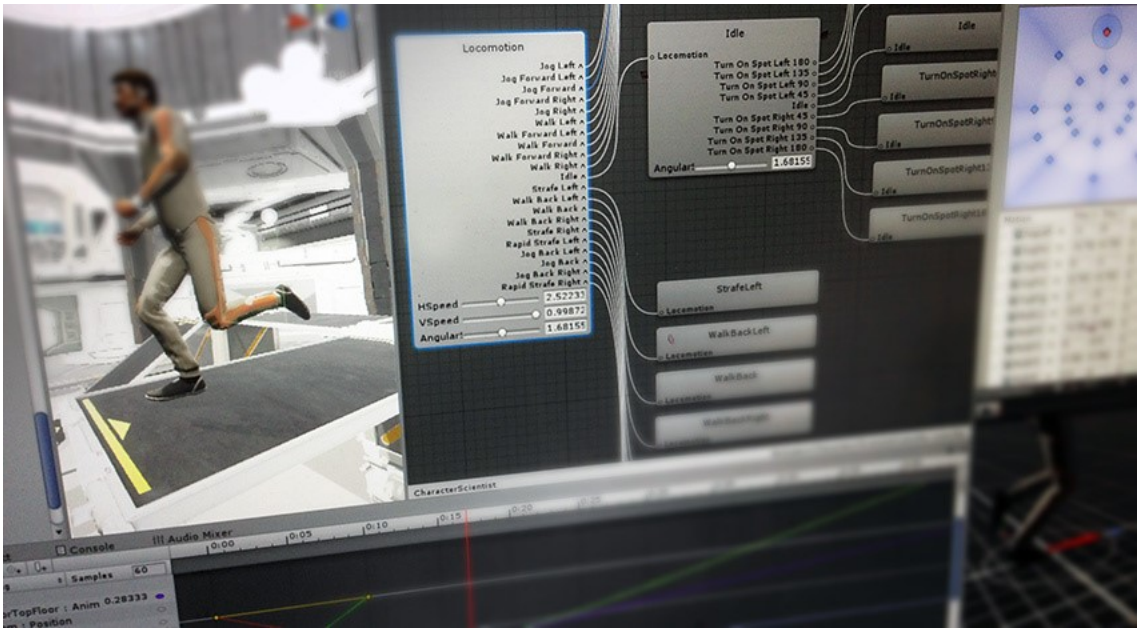

Figure 52 – Unity animations[51]

AUTHENTICATION

When the user first launches the application, they are presented with an authentication window. This window replicates the login experience found on the server authentication. Within this window, the user is required to register by providing their username, email address, and password. These input fields are neatly organized within a form. Additionally, the user can switch between the login and registration windows using two text links: "Don't have an account? Sign up" or "Already have an account? Sign in" Each of these links is associated with a script that intelligently checks which window is currently open, deactivates the open window, and activates the other closed window.

To make this dynamic interaction work seamlessly, it is crucial to define the game objects and input fields within Unity's Inspector for these text links. This establishes a direct connection between these objects, allowing for easy enabling or disabling as needed. Alternatively, developers can utilize Unity's "find" function to locate a game object by its name, simplifying the activation and deactivation

[51] https://docs.unity3d.com/Manual/AnimationSection.html

process.

Within the registration section, it is essential to define the elements associated with registration validation. This includes specifying text fields and setting up validation messages. The validation process begins by checking for the presence of the registration button in the scene. If it is found, a listener is thoughtfully added to it. Think of this listener as a trigger for an event. Inside this listener, a carefully created function is defined to execute when the registration button is clicked. In this function, thorough checks are performed to ensure the input fields are valid, confirming they are not empty and that the email is in the correct format. Regular expressions (regex) are employed for this purpose. If all the validation checks pass without any issues, a new user is created with the data provided by the user and the user is sent to the login window. In cases where there are problems with the input fields, the user is promptly notified of the specific errors.

To create a new user, the application relies on UnityWebRequest. This adaptable object facilitates communication with a web server, specifically the URL responsible for creating new user accounts. The user's data is accurately stored within a form and then transmitted via the web request. If the request encounters any issues or fails, the user is presented with clear and informative error messages.

The login process follows a similar logical flow. It initiates by checking for the existence of the login button within the scene. If the button is present, a listener is added to manage the login functionality. This listener serves as a crucial trigger for an event. Within this listener, user-entered email and password information is meticulously collected. These credentials are essential for executing the actual login.

Subsequently, a secure connection is established with the login route, and a UnityWebRequest is initiated, utilizing the gathered login data. If the login operation proves successful, the user is promptly notified of their successful authentication. Furthermore, their user data is securely stored for future interactions, like the session storage mechanism in web browsers. This storage of user information is implemented within the user session class, ensuring consistent access to user-specific data throughout their application journey.

Once the login process is successfully completed, the exciting virtual tour experience within the application can commence.

The “UserSession” class server as a mechanism for storing user data in the form of keys within the PlayerPrefs. PlayerPrefs is a class that stores user preferences between sessions. When a user logs in, keys associated with their user details are stored within the PlayerPrefs. These keys are later removed when the user logs out. These stored keys can be accessed and recognized through appropriate functions.

In Figure 53 and Figure 54, it is possible to observe the design of the user login and registration interface, which is similar to the application on the website.

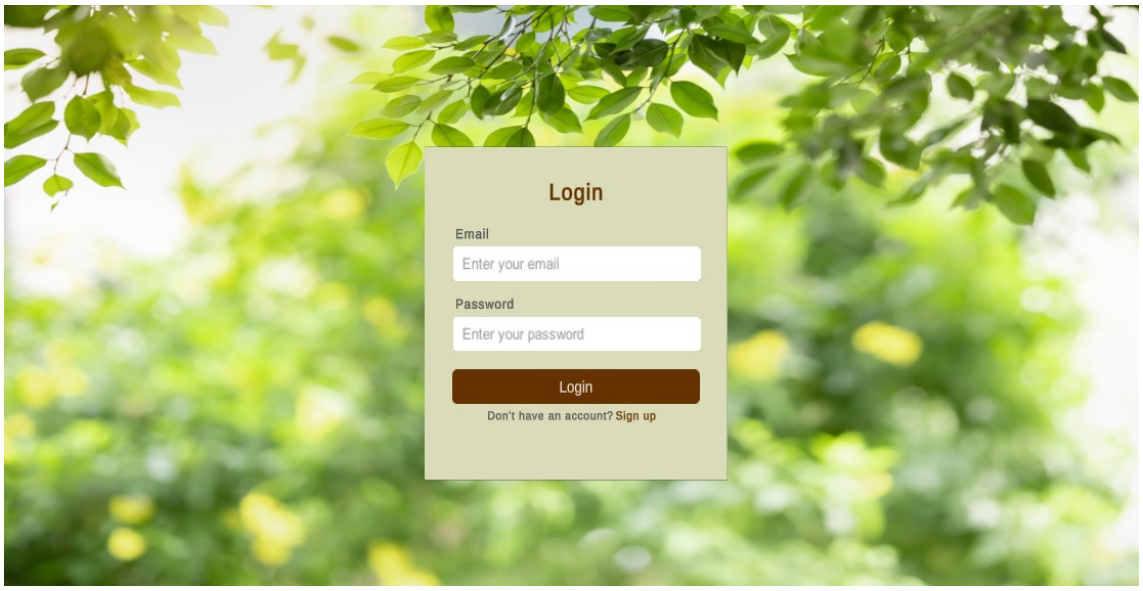

Figure 53 – Login (client)

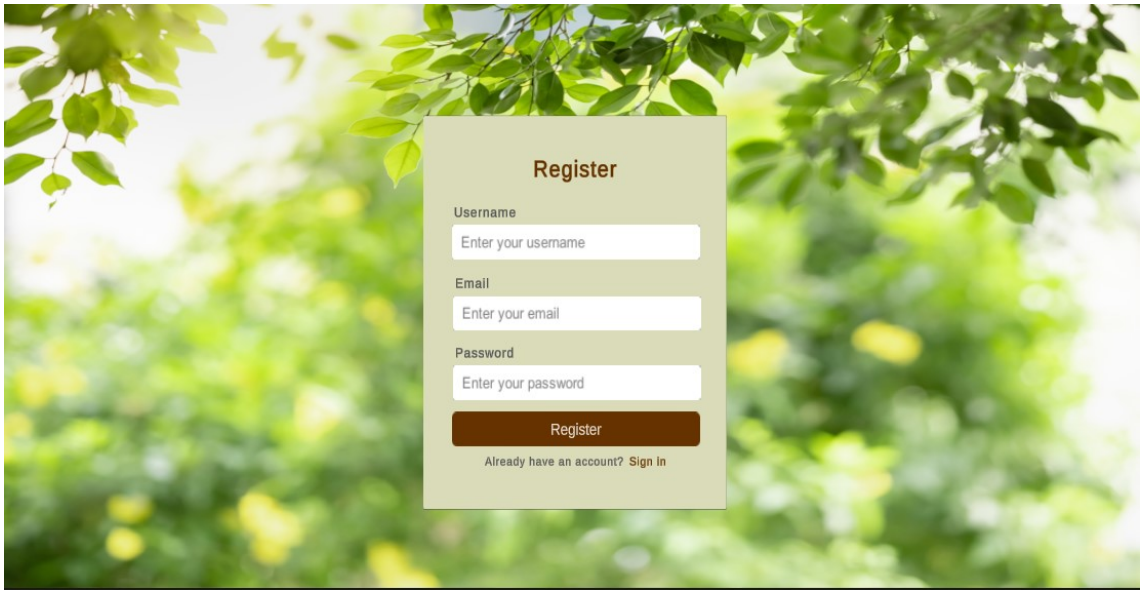

Figure 54 – Register (client)

SCENE CONSTRUCTION

To ensure a smooth and visually appealing transition between scenes, a scene loader was meticulously created to implement a crossfade effect seamlessly. The development of this scene loader involved the creation of a script, which takes several key parameters as inputs: an animator, a transition time, and the total number of scenes in the project. This script's core purpose is to manage the scene transitions smoothly.

First and foremost, the script checks whether a scene loader exists within the current scene. This is a crucial step in maintaining consistency and preventing unexpected behaviour during scene transitions. Once it verified the existence of a scene loader on scene, it proceeds to determine the number of scenes in the project and the current scene number. Loaded with this information, it makes

informed decisions regarding the next scene to load. If there are additional scenes ahead and the user doesn't want to go back to the previous scene, it proceeds to load the next one in sequence. Conversely, if there are no more scenes to load or the user wishes to go back, it delicately handles the transition to the previous scene, ensuring a seamless user experience.

It is important to consider that if the user wishes to transition between scenes, the connection that has been established via websockets must be disconnected. This ensures that when the user return to the scene where avatars are located, there are no duplicate connections for the same user.

When working with websockets and transitioning between scenes in an application, managing the connections efficiently becomes crucial. In this context, it is necessary to implement a mechanism that handles the termination of a websocket connection when a user navigates away from the current scene. To achieve this, the application should include a logic that detects when a user is leaving a scene or switching to another one. Once this event is detected, the system should initiate the disconnection of the websocket connection established for that user within the current scene.

One of the important components of this scene loader is the creation of an animator from scratch. The animator is responsible for arranging the crossfade effect. During the transition between scenes, the animator gradually adjusts the screen's colour, leading to a gradual darkening effect. This darkening effect is what created the fade-out impression as the current scene prepares to change. The timing and intensity of this fade are controlled by the specified transition time.

As the new scene loads and become the focus of attention, the animator's logic continues. It starts to gradually lighten the screen, creating a reverse effect compared to the initial fade-out. This gradual lightening of the screen essentially constitutes the fade-in effect, bringing the new scene into full view. This transition logic, carefully managed by the animator, ensures that the crossfade between scenes is not abrupt or shocking but rather a visually pleasing and seamless experience for the user.

In Figure 55, it is visible how the crossfade animation is set.

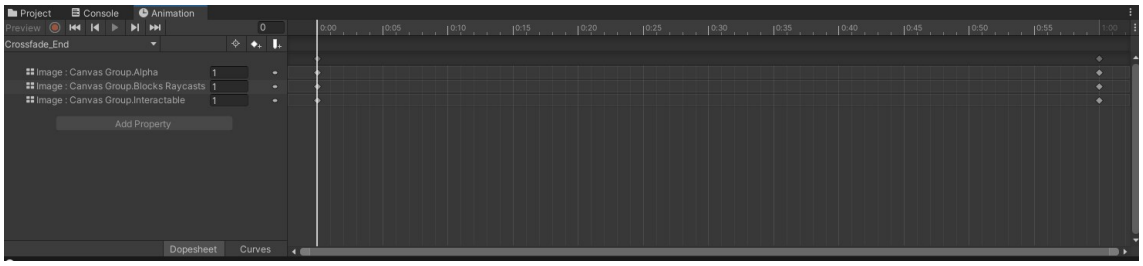

Figure 55 – Crossfade animation

This scene loader script, together with the custom animator, enhances the overall user experience by providing a refined and polished transition mechanism between scenes. It allows for a smooth crossfade effect, maintaining user engagement throughout the journey within the project.

In Figure 56, a representation of how the animator was structured in order to create the crossfade effect is shown.

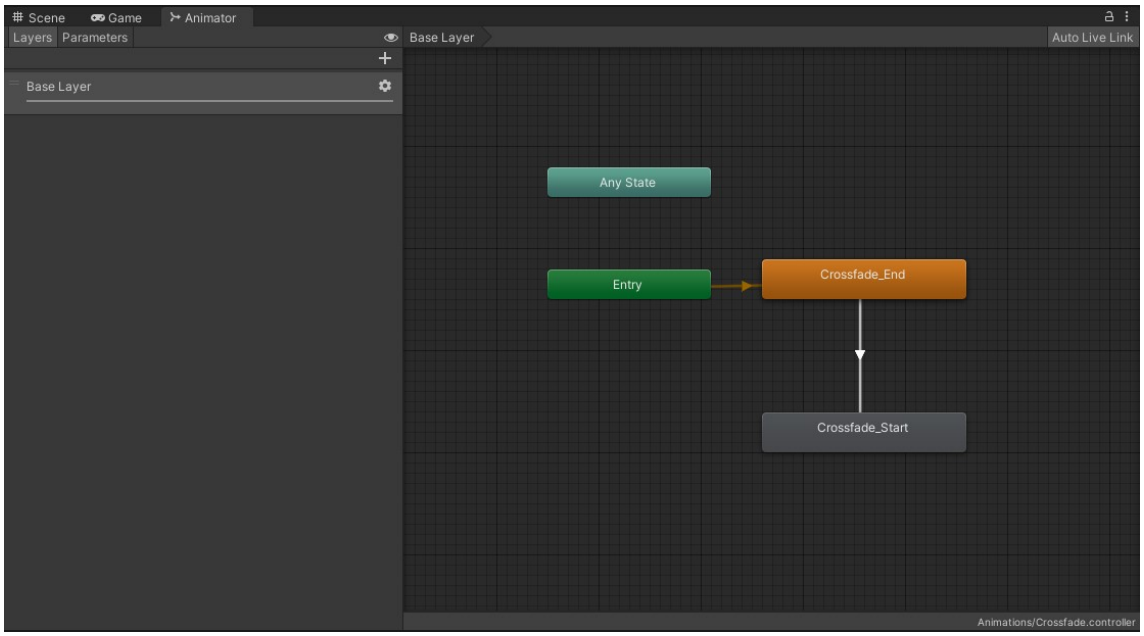

Figure 56 – Animator

To create 360-degree images, the process begins by constructing spherical environments. These spheres essentially encapsulate the images or image sequences that the user intend to navigate through. The spherical shape allows for a seamless immersive experience, as it surrounds the viewer, providing a sense of immersion.

Within these spherical environments, the transition icons are placed on scene in a strategic way. These icons serve as visual hints to the user, indicating points where they can transition from one scene to another. A key principle in positioning these icons is to place them at locations that align with the corresponding points in the next scene. For example, if in one image a car is depicted at a distance from the camera, and in the next image, the car is up close, this can be a suitable transition point. It provides users with a natural and intuitive sense of movement as they navigate through the scenes.

To make this navigation possible, it is necessary to implement two essential scripts:

Camera Rotation Script: This script is responsible for allowing users to observe the scene from any angle within the 360-degree environment. It is crucial for creating the immersive experience. This script typically relies on several data points:

- Mouse position: It tracks the last recorded position of the mouse cursor.
- Rotation speed: Determines how quickly the camera responds to mouse movements.
- Camera reference: Provides a point of reference for the camera.

The script consists of a function that continuously checks whether the right mouse button is pressed. If it is, the script calculates the change in mouse position since the last recorded position and adjusts the camera's angles accordingly. Additionally, it updates the initial position of the camera with the new position and applies the necessary rotation to the 360-degree object, aligning it with the user's intended view.

Scene Transition Script: This script is responsible for managing the transitions between scenes within the 360-degree environment. It ensures that when users interact with the scene transition icons,

they smoothly transition to the next scene.

The script recognizes three cursor modes:

- Normal: Representing the default cursor state.
- Drag: Activated when the user is dragging an object.
- Clickable: Triggered when the cursor is positioned over a clickable element.

The images associated with these cursor states are typically configured through the inspector.

The cursor mode is set to “ForceSoftware”, which enforces the use of software-defined cursors for consistency and smooth interaction. Additionally, the script checks if the left mouse button is pressed. When pressed, it changes the cursor state to “drag”. Upon releasing the left mouse button, the cursor reverts to its normal state. Furthermore, when the cursor hovers over a scene transition icon, the script switches the cursor state to “clickable”, indicating to the user that they can interact with the icon to transition to another scene.

By combining these scripts, an engaging and immersive 360-degree image navigation experience that allows users to explore and interact with different scenes seamlessly is created.

The fact that the project revolves around a platform featuring 360-degree navigation with two distinct roles, the user and the avatars, bears a striking resemblance to the concept of ShowMeAround.

INTERACTION

The application provides users with two distinct modes of interaction with avatars: dynamic interaction facilitated through server-side chat support and manual interaction that involves direct engagement with the avatars for basic information retrieval.

In a more visually intuitive manner, users can initiate manual interaction with an avatar by simply clicking on the avatar itself. This action triggers the appearance of a dialog box position at the lower part of the screen. The design of this dialog box draws inspiration from various gaming experiences that involve dialogues with avatars. As a result, the current dialog box takes the form of a rectangular container, within which users find essential information concerned to the guided tour location. This information includes fundamental details about the location, such as its name, geographical coordinates, and a brief description of what the user is currently viewing. The dialog box also distinctly features the image of the avatar to make it clear which character is “speaking”. Users can continue their conversation with the avatar by clicking on it, and the ongoing exchange is seamlessly displayed within the dialog box, ensuring that users can effortlessly navigate and make use of the application’s features. In instance where the dialog contains dynamic elements, such as images, hyperlinks, videos, and other interactive components, and arrow is displayed above the avatar’s head, indicating that these elements are clickable and will offer additional information or actions.

To go deeper into the technical aspects, each avatar is associated with a script that governs the manual interaction process. Within the inspector, the application’s developers can configure several critical parameters related to the dialogue interface. These configurations include the appearance and behaviour of the dialog box, links that users might wish to share, and the images of the avatar that is

interacting. Inside the script, static phrases intended to be conveyed through the avatar are stored within an array. The size of this array is flexible, dynamically adjusting to accommodate as many dialogue lines as necessary, thereby allowing for the inclusion of any desired content without constraints.

When a user clicks on an avatar, the application performs a series of checks. It verifies whether the avatar being used to transmit information is the male avatar or the female avatar, adjusting the avatar's visual representation on the dialogue box to match accordingly with the selected avatar. Additionally, the application checks for the activation status of the dialog box and examines whether there are unread lines of dialogue. If unread lines exist, the application verifies if the text has any dynamic elements, such as clickable links. If any dynamic element is detected, a notification is sent to the dialog box, instructing the user to click on the arrow positioned above the avatar. This arrow becomes active after this validation process, signifying that there is more interactive content to explore. If there are no additional lines to read, the dialog box is deactivated, and the text is reset, ensuring a smooth user experience.

The arrow responsible for opening dynamic elements possesses an associated script that plays a crucial role in its functionality. This script actively checks whether an avatar is present within the scene. When an avatar is detected, the dynamic element associated with it is stored in a variable and can be opened through a simple mouse click, enhancing the application's interactivity and user engagement.

In Figure 57, a depiction of the dialog box with the avatar without any interactive elements for the user to enjoy can be observed, and in Figure 58, it is possible to observe how the design changes with the interactive element.

Figure 57 – Dialog box without interactive elements

Figure 58 – Dialog box with interactive elements

CHATBOX

The dynamic interaction functionality with avatars, particularly the ability to engage in conversations with the server-side support, involves the development of four essential scripts. These scripts play distinct roles in ensuring the seamless operation of the system.

One of these scripts serves as a data repository for avatar-related information. This information encompasses crucial details such as the avatar's username, email address, and type, which distinguishes between Unity and EJS users/avatars. This stored information plays a crucial role in managing server-side validations.

Another script's primary purpose is to activate the conversation window with avatars. For this functionality to work smoothly, it requires specific inputs, including the class containing the avatar's data, the target avatar with which the conversation was initiated, and the GameManager. It is important to note that when this script is initially executed, the conversation window remains in a deactivated state. This decision is intentional, it ensures that the chatbox is not displayed immediately upon launching the project. Instead, it becomes visible only when explicitly requested by the user.

The conversation icon is strategically positioned above each avatar's head, and clicking on this icon initiates a validation process. This validation has the function of determining whether the conversation window is already active. In the event that the window is active, clicking on the icon results in the chatbot being closed. On a desktop platform, this effect can also be triggered by pressing the ESC key. Alongside closing the conversation window, this action also terminates the session with the WebSocket.

Conversely, if the validation process detects that the conversation window is not active, it proceeds to activate the window and restricts the option to navigate through the scene while the conversation is in process. To enhance the user experience within this functionality, there is an additional script responsible for disabling the mouse cursor. This script monitors the status of the conversation window, enabling or disabling the cursor accordingly. This ensures that users can

engage with the conversation more effectively without the distraction of an active cursor.

In Figure 59, the user can observe the dialog box icon above the avatar's head, clicking on that icon the user can interact with the support behind the avatar.

Figure 59 – Dialog icon on avatar

The "GameManager" script is a crucial component referenced in various preceding steps, it serves a critical role in managing the chat functionality between the client and the server within the application's architecture.

First and foremost, this script initiates its operation by establishing a set of variables that are crucial for seamless chat functionality. These variables include specifications such as the maximum allowable number of messages and references to essential game objects associated with the chat interface. These objects encompass elements like the chat panel, the scroll view, the text display, and more. Each of these variables is carefully defined to ensure a smooth interaction between the user and the chat system.

In addition to these variables, the script also defines a series of boolean variables. These

boolean variables serve various purposes, such as determining whether the chat is currently open, tracking whether it is the user's initial interaction with the chat, distinguishing whether a message originates from the player or contains informative content, and identifying the type of client device in use, distinguishing between desktop and mobile platforms. This careful consideration of boolean variables facilitates the script's ability to adapt and respond effectively to different user scenarios.

Another important aspect of the script's functionality involves the storage of avatar data. This data is encapsulated within a variable of the AvatarData type, preserved for future utilisation. The avatar data holds essential information related to the user's profile or identity, ensuring that it can be seamlessly integrated into the chat interface as needed.

For the chat system to maintain a comprehensive message history, it relies on the existence of a dedicated list. This list serves as a repository for storing all incoming and outgoing messages. Every message that goes through the chat system is stored in within this list, maintaining a chronological record of interactions.

One of the most important variables defined within the script is related to the WebSocket. The WebSocket is an important communication protocol that allows bidirectional data exchange between the client and server through a single, persistent connection. It plays a crucial role in facilitating real-time, interactive features and low-latency communication within the application. Additionally, the script considers the potential need to reconnect to a WebSocket if the connection is disrupted or closed, ensuring uninterrupted chat functionality.

In the "Start" function, the process initiates with a crucial step, the addition of a listener to the input field. This initial action targets mobile devices, where the user interaction differs significantly from the desktop. The purpose of this listener is to observe user behaviour in the context of mobile devices. Specifically, it verifies whether the user is currently on a mobile device. If this condition is met, it proceeds to check the text box within the user interface.

On that check is pays close attention to a key factor. It checks whether there is any text present within the text box when the user ends the edit of the input field. If that condition is satisfied, the boolean variable responsible for identifying the device as mobile changes to a "true" state.

Apart from this listener, the script attempts to reestablish a connection with a WebSocket. To achieve this, it invokes a dedicated function designed to manage reconnection events. This function follows a specific logic: it first checks whether the current reconnection attempt count is below the maximum allowable limit. If the limit has been reached, the user is informed of this, indicating that further reconnection attempts are futile. However, if the attempt count remains within the permissible range, the function proceeds. It increments the current attempt count and calls another function responsible with terminating the WebSocket connection, ensuring that no lingering connections persists.

Following this, a crucial verification step takes place. The scripts verifies whether this is the first attempt to connect to the WebSocket. If it is the user's first attempt, the WebSocket initialization function is called without any delay. Otherwise, if this is a subsequent connection attempt, the WebSocket initialization function is invoked with a predetermined delay.

The function responsible for terminating the WebSocket connection does a validation routine. It

verifies the presence of an active WebSocket and checks whether its state remains in the “open” status. If both conditions hold true, it takes some actions. Specifically, it deactivates both the scroll view and text display elements, disabling the user’s ability to interact with the chat interface. Furthermore, it proceeds to terminate the WebSocket connection, effectively closing the conversation window from the user interface.

The function responsible for initializing the WebSocket initiates a crucial process. It creates a new WebSocket instance, it with the appropriate URL corresponding to the server’s location. This WebSocket object has four key functions.

The **OnOpen** function that defines actions to be taken when a WebSocket connection is successfully established. It provides feedback in the form of a console message indicating the successful connection. Additionally, it resets the reconnection attempt count to zero and triggers the function responsible for notifying the user of the open connection.

The **OnMessage** function has a crucial role in handling incoming messages via the WebSocket. It begins by storing the server’s response withing a designated variable. It then conducts a series of checks, first it checks whether the sender of the message is the server itself and the message type corresponds to “onlineEJSUsers” or if the action belongs to a login request. If that is the case, is created a user list with the objective of storing the identities of users currently logged into the server. If this list’s size is different from zero, it triggers a function name “FindAvatarByEmail” for each user contained within the list. That function seeks and identifies the appropriated avatar that corresponds to the user on the server, facilitating its representation within the interface in a 3D form.

If the message type is not “onlineEJSUsers” – implying that a login action is not taking place – but instead involves the insertion of a message, the message is sent to the chat interface through a designated function responsible for handling message inputs. This function ensures that relevant information pertaining the message is sent, including the message content itself, a notation indicating that it is a user-generated message sent by the server, the email address of the user who initiated the message, and the email address of the user currently active in the client’s session. This information ensures that the message is accurately displayed and attributed within the chat interface, facilitating effective communication within the system.

If the action responds to the logout event rather than a login request, the function responsible for concealing the avatar associated with the user who just terminated the session is called and it proceeds to terminate the WebSocket connection.

The **OnError** function is responsible for taking care of errors when those occur in the WebSocket communication. It serves as a mean of providing the user with feedback regarding any issues or problems encountered during the communication process through a console message with the specific error.

The **OnClose** function is called when a WebSocket connection is closed. Its responsibilities include updating a boolean variable designed to determine whether the chat interface is currently open. It the application is still running, it invokes the previously mentioned function, which attempts to reestablish a connection with the WebSocket.

Now, let’s delve into a comprehensive explanation of the functions responsible for concealing

avatars and locating them based on email addresses. The function responsible for hiding avatars functions by maintaining a list of avatars. For each avatar within this list, it verifies if the email associated with the avatar matches the one target for hiding. If a match is detected, the function proceeds to identify the parent GameObject of the avatar and alter the state of this GameObject from active to inactive, in the end returning the avatar.

In contrast, the function responsible for locating the avatar operates in a similar way to the hiding function, with the distinction being that it changes the state from inactive to active.

Previously, the Start function was explained. Now the Update function will be explained. The function starts by verifying whether the text that is intended to provide instructions on closing the chat, both for mobile and desktop users, is empty. If the text is not empty, it is necessary to clear it. After that, two additional checks are conducted. One of these checks verifies whether the websocket is not null and whether the state is set to open. If these conditions are met, the function responsible for queuing sent messages is invoked.

The next verification checks whether the chat is in an open state. If the chat is indeed open, a series of verifications takes place. For instance, it checks if the application is being launched on a mobile device through the function that verifies if the system is an android or iOS. If it is a mobile device, the text sent to empty in the beginning will alert the user that he can close the dialog box by clicking on the dialogue icon, if it is not a mobile device the text will alert that closing will be executed by pressing ESC.

One of the verifications is whether the chat is not focused and if the space key is clicked, it that's the case a message is sent to the chat to alert the user that the space was pressed, different from the user text, this one is in read, it symbolizes an information. If there scenario where the input field is not empty and the user clicks on the return key on the desktop or stops editing on the mobile device, the message is sent to the chat through the appropriated function, that will comprise the message's content, an identifier indicating that it was sent by a user, the user's name and email, and the designated recipient, who corresponds to the user whose avatar was selected. If the input field GameObject is not focused and it is not a mobile, the input field will become editable.

In Figure 60, it is demonstrated how the dialog box shows for the desktop client, you can see that the dialog in red changes from the desktop client to the android client. In Figure 61, it is demonstrated how the dialog box shows for the android user.

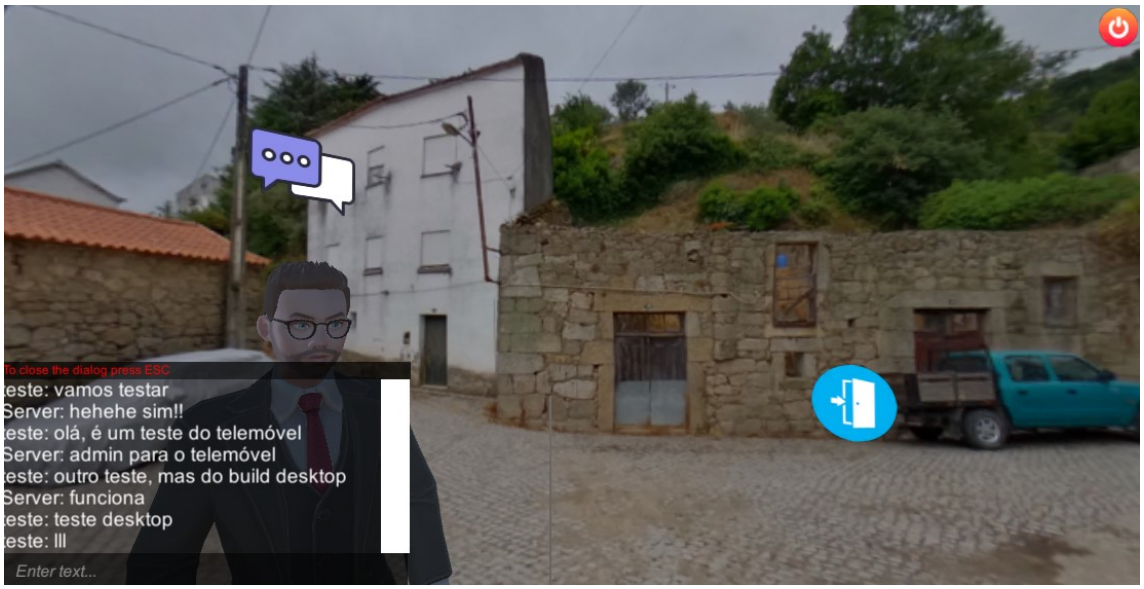

Figure 60 – Application on desktop

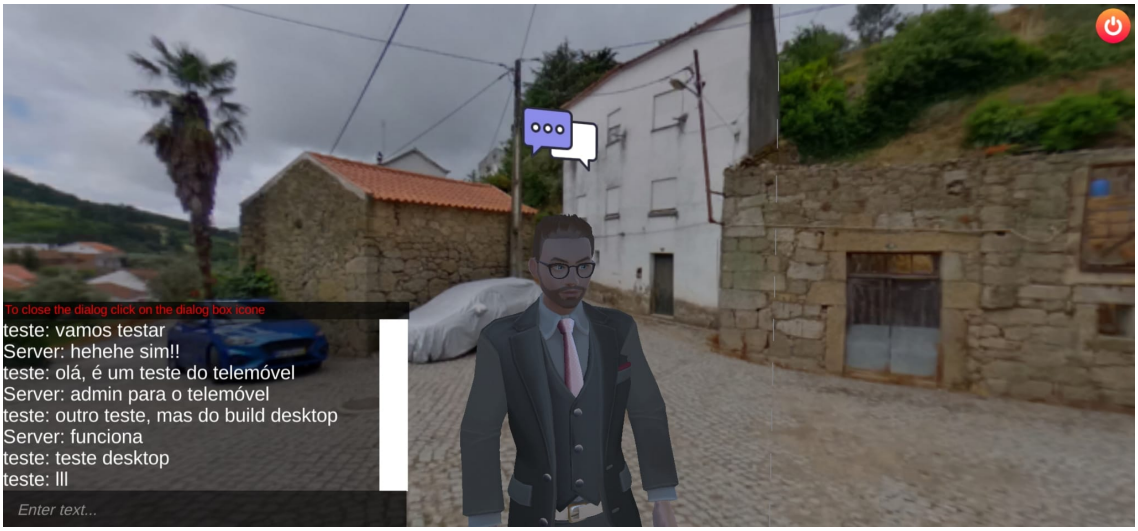

Figure 61– Application on mobile (android)

When a user selects an avatar within the application, it triggers the execution of a specific function. This function serves the critical purpose of storing the avatar's data in a variable known as selected avatar data. This stored data becomes important for later retrieval of the avatar's conversation history.

Before proceeding further, the system realizes a check to verify whether any avatar has indeed been selected. If no avatar has been selected, an error handling mechanism is invoked. However, if an avatar has been successfully selected, the system proceeds to initiate another function responsible for retrieving the conversation history associated with that avatar.

The function responsible for managing the avatar's conversation history initiates by clearing the conversation window. To accomplish this, it systematically destroys all game objects recognised as child elements of the chat panel. Simultaneously, it clears the message list with the use of the clear method.

Once the chat window has been cleared and is ready for new data, the system proceeds to store the URL for the message history route in a designed variable. This URL is backed with essential information, including the email addresses of both the selected avatar and the unity user.

With the necessary information, the system then proceeds to make a specific type of network request known as a “UnityWebRequest”. In this case, the request type is a “Get” request. If this request is successfully executed, the system receives a JSON response, which is stored within a designated variable. To work effectively with this JSON data, it is essential to store it within the “data” property using JSONUtility.

Subsequently, the JSON response must be parsed to extract the relevant information and structure it appropriately. This information is then stored within a list created for the chat history.

For each message contained within the chat history list, the system performs a conditional check. If the message’s sender is identified as the server, a specific action is taken. This action involves sending the message data to the server, including the sender’s name, the recipient’s email, the sender’s email, and the content of the message, all under the identifier “Server”.

Conversely, if the sender of the message is determined to be the logged-in user, the system executes a similar action, sending the message data with the user-related information and the user’s name. If the network request is unsuccessful, the program will notify the user of the encountered error.

When a message is generated within the unity application, it has a dual purpose, not only is directed to the chat interface for immediate display, but it also requires transmission to the server for further processing. This multifaceted task is executed through a function designed to handle this specific functionality.

This function starts by accepting a message as parameter. Its initial action is to check the status of the WebSocket connection. For the process to work, the WebSocket connection must be in an open state otherwise the user will be informed. This check is important since the program will attempt to send data over the WebSocket and can result in an error.

Once the WebSocket status is confirmed as open, the message data must be structured in a specific way to facilitate its transmission and interpretation on the server side. To acquire this, the message data is organized into a structured format known as MessageData. This class not only contains the actual content of the message but also includes metadata related to both the sender and the recipient.

The recipient of the message is determined by the selection of an avatar to star conversation. Each avatar within the unity environment is associated with specific user data. Consequently, the message can be correctly directed to the intended recipient by using the data associated with the chosen avatar.

After the message data has been suitably formatted, the next step is to convert it into JSON format. This conversation is required because the sever has been designed to handle messages in JSON format. This ensures seamless communication and processing between the client and the server.

Once the message has been converted into JSON, it is transmitted via the WebSocket connection to a specific URL endpoint. This URL corresponds to the server route responsible for

processing and storing messages submitted through a form. The success of this transmission is indicated in the Unity console. If sent successfully, a confirmation message appears, otherwise and error message is displayed.

The chat history requires a certain handling. As messages accumulate over time, a limit on the number of messages stored is enforced. When this limit is exceeded, the oldest message in the list must be removed, and the corresponding visual element, represent by a GameObject, is destroyed. This practice helps maintain a manageable and non-excessive message history.

Furthermore, a crucial verification process is performed to determine the eligibility of messages for display. Specifically, it checks whether the recipient corresponds to the user with an active session, and whether the sender aligns with the selected avatar or vice versa. This verification is important to ensure that only messages between two specific users are displayed in the chat interface, thus preventing unauthorized access to private conversations.

If the conditions for message display are met, a new message is dynamically generated within the chat interface. This message includes the content of the message, along with the sender and recipient. Moreover, the message is added to the list of existing messages, and two functions are invoked to help that functionality. One function manages the colour scheme of the message, ensuring visual coherence within the chat interface. This function examines the type of message being processed, if the message originates from a user, then it is displayed in white in the chat box otherwise if it is an informational message it is displayed in red, that will provide clarity and readability of the chat interface. The other function enables automatic scrolling to the most recent message, eliminating the need for users to manually scroll down to view new messages. This function executes a specific coroutine designed to reset the scroll view's position to zero.

Besides these functions, there is another function. This function is responsible for notifying the server whenever a user establishes a connection. It apprehends and stores user-specific data within a designated variable, and subsequently transmits this data to the WebSocket.

It is important to note that the script extends beyond functions. It incorporates various serializable classes to organize and manage data effectively. Together, these functions and classes form an integral part of the script, ensuring a smooth functionality of the messaging system while providing a structured approach to managing data and user interactions.

These classes include:

- A class responsible for storing user data and maintaining a list of all users.
- Other classes dedicated to managing message data, which includes an enumerator to categorize message types.
- A class designed to store received server messages, containing essential information such as the action to be taken and the type of message.
- A class aimed to store historical message information.
- A class dedicated to maintaining a comprehensive record of message history in the form of a list.

Architecture diagrams

Client Authentication

In Figure 62, the client begins the authentication process by entering their login credentials. The application then tries to establish a connection with the server using these credentials. Since the server requires verification, it checks for the existence of a client in the database with these credentials by executing a query. Regardless of the result, the server is informed of the search outcome. If the search is successful, the client is granted access to log in to the application.

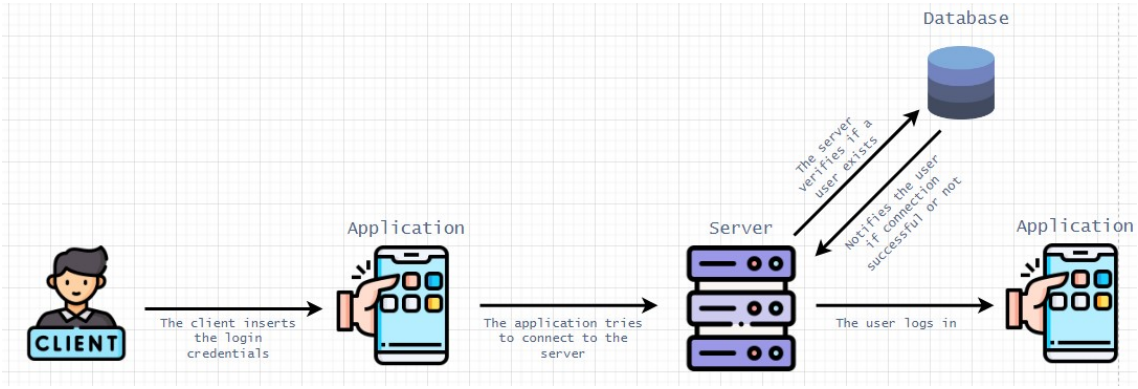

Figure 62 – Client Authentication

Admin Authentication

In Figure 63, the authentication process is similar to Figure 62, but instead of the admin logging in from the client application, they initiate the authentication process in the web application.

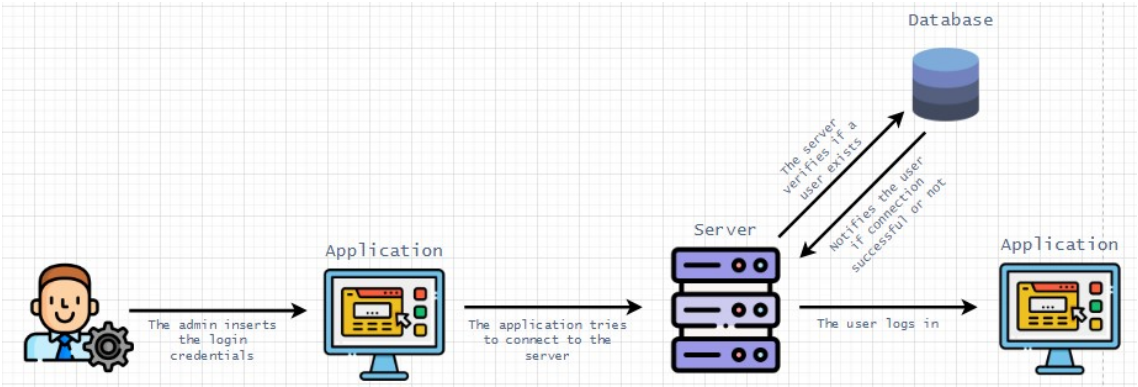

Figure 63 – Admin Authentication

Client Message History

In Figure 64, when the client interacts with the avatar to initiate a conversation, the chatbox opens. This functionality is enabled by the system’s ability to maintain a message history. The application attempts to request this history from the server, which, in turn, seeks to retrieve the conversation between the client and the avatar's user from the database using a query. If the search is successful, the historical conversation is then transmitted to the Unity application and displayed in the client's chatbox for their observation.

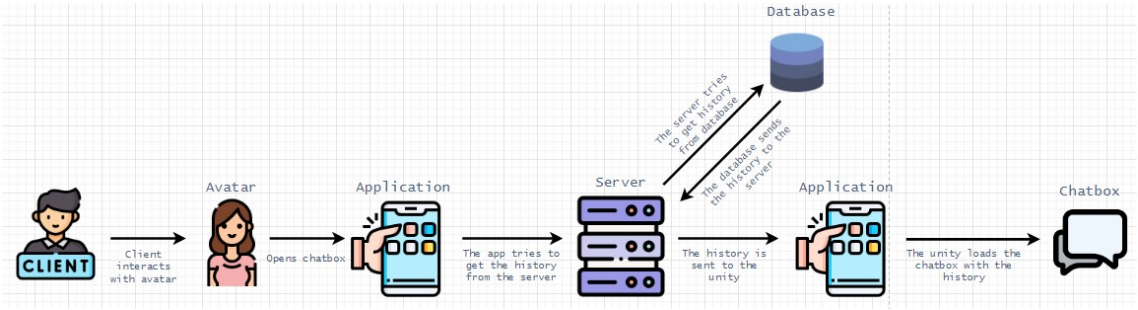

Figure 64 – Client Message History

Admin Message History

In Figures 65 and 66, the process that occurs when an admin attempts to open a conversation with message history is illustrated. The admin begins by opening the chatbox tab within the web application. At this point, the application needs to display a list of connected clients, so it requests this list from the server. The server then presents this list on the website interface. Next, the admin must select a specific client from the list. Once a client is selected, the web application initiates a request to retrieve the message history from the server. To accomplish this, the server must, in turn, make a request to the database to fetch the conversations between the two users involved in the selected chat. Once the database returns the conversation to the server, the message history is sent to the server's frontend, allowing the admin to view it in the conversation interface.

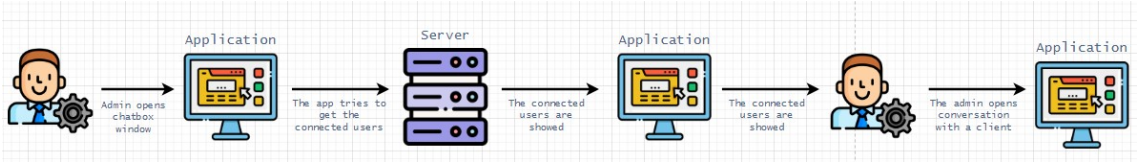

Figure 65 – Admin Message History (1/2)

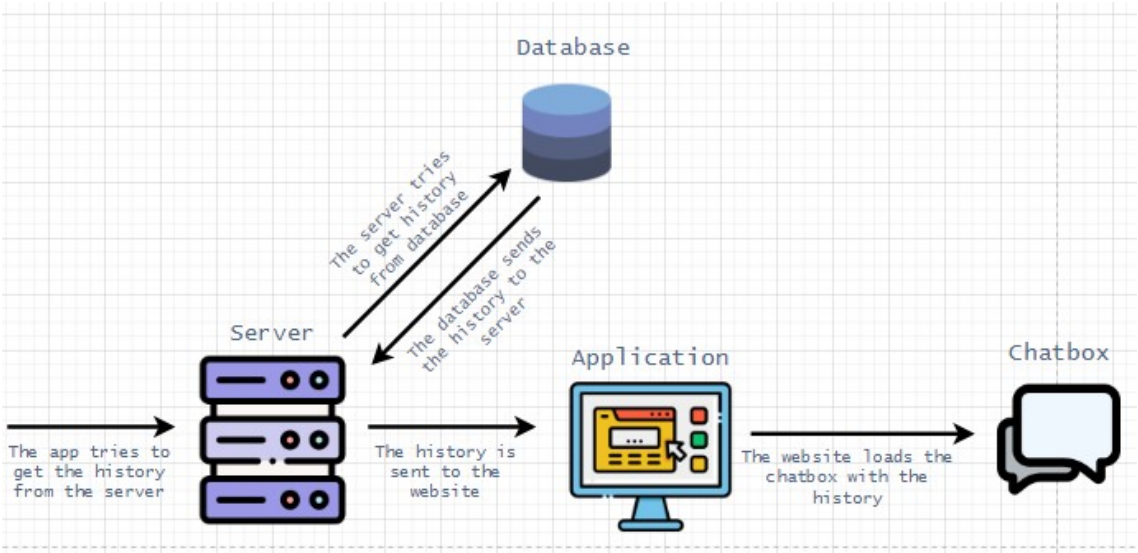

Figure 66– Admin Message History (2/2)

Client Message Sending

In Figure 67, the client initiates a message exchange with the avatar. The application attempts to

transmit the message to the admin whose credentials match those of the avatar. Subsequently, the server tries to incorporate the newly composed message into the database. If this process is successful, the server proceeds to dispatch the message to both the administrator and the client. As a result, both parties can view the message within their respective conversation windows.

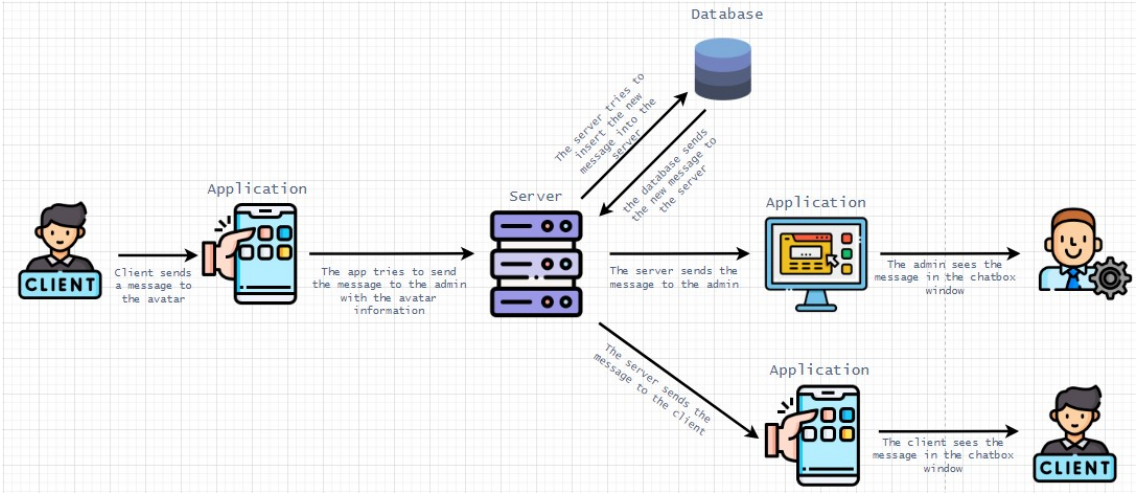

Figure 67– Client Message Sending

Admin Message Sending

In Figure 68, the process is similar to Figure 67. The only difference is that the user initiating the conversation is the admin, and the application attempts to send the message to the selected user.

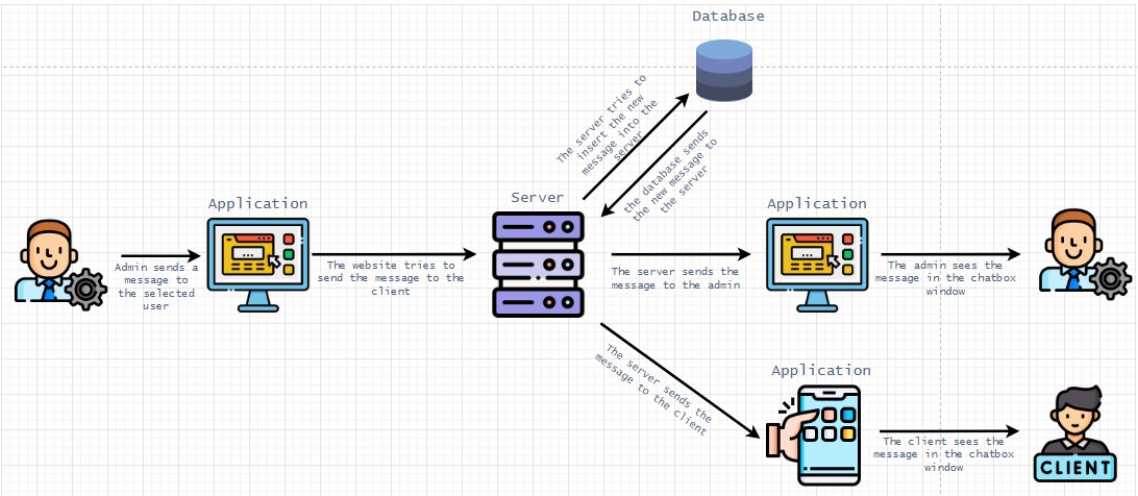

Figure 68– Admin Message Sending

4.2.4. Physical tours

As previously mentioned, engaging with audiences both online and in-person is something vital for today's fast-paced digital world. In addition to virtual tours, which in turn allow individuals to explore a space or experience remotely, it is also essential to make physical tours captivating. That can be achieved with the use of avatars, virtual tours can be engaging through real-time interaction with avatars, but physical tours can also be engaging throughout the use of the AR technology, where

avatars are overlayed onto the physical world, creating a blended experience. The visitors can use their smartphones to scan logos or images related to a particular location and when they do so, the avatars in the logos come to life, providing additional information and entertainment.

The integration of avatars from virtual tours into the physical tours creates a seamless and captivating experience for visitors, closing the gap between virtual and physical environments. This not only enhances the overall visitor experience but also strengthens the marketing efforts of the location.

AR COURSE

To establish a solid foundation before diving into the development of an AR application from the scratch, it was important to undertake a comprehensive training course[52], provided by Unity. The goal of this course was to equip participants with the skills necessary to create an AR application utilizing markers and planes. This course is divided into two distinct projects, each with its unique focus and application.

The initial project centres around the development of a marker-based AR application, which essentially involves a technique that relies on the recognition of real-world images acting as markers to trigger the display of digital content. This approach has been chosen as the primary technique for the application under development in this project. In a marker-based AR application, the software must be capable of identifying specific images or patterns in the physical environment, often referred to as markers, which serve as triggers for the overlay of augment content.

The second project explores the creation of an AR application utilizing plane detection technology. This technology makes use of the innate capabilities of smartphones to detect flat surfaces within our environment. This functionality enables augmented reality to superimpose 3D objects onto these flat surfaces, as perceived through our smartphone's camera. This offers users a more dynamic and immersive AR experience, as the virtual elements interact with the real-world environment in a more sophisticated manner.

AR APPLICATION – CREATION

Before delving into the development process, it is crucial to acquire and set up Vuforia SDK. This SDK can be obtained from the official Vuforia website[53]. To access the SDK, the developer needs create an account on the platform.

Once the SDK is successfully downloaded, the next step involves integrating Vuforia into the Unity environment. This integration is essential as it provides the necessary framework and tools for building AR experiences using Vuforia's capabilities. It is worth noting that Unity offers a predefined template for AR applications, which serves as a solid starting point for such projects. However, in the context of the current project, this templated was not utilized due to customization preferences.

With Vuforia integrated into Unity, the developer can begin adding Vuforia game objects to the

[52] https://learn.unity.com/course/create-with-ar-markers-and-planes?uv=2021.3

[53] https://developer.vuforia.com/downloads/SDK

project. In this project, two key game objects were utilized: the AR camera and the image target. The AR camera, which is fundamental to the AR experience, requires configuration through Vuforia. This configuration can be accessed via the Inspector panel of the game object by selecting the “Vuforia Behaviour” section. When configuring the AR camera, the developer will be asked to provide a license key for the application’s usage of Vuforia, it is possible to observe that option in Figure 69.

To obtain the required license key, one must navigate to the Vuforia website and through the menu there will be an option “Develop”. Clicking on this option opens a window displaying the keys associated with the developer’s account. The specific key needed for the project in hand is the basic key, which can be generated by selecting the “Get Basic” button and assigning it a name. Upon completion, a unique key will be generated for the developer to copy and paste into the appropriate field within the Unity environment.

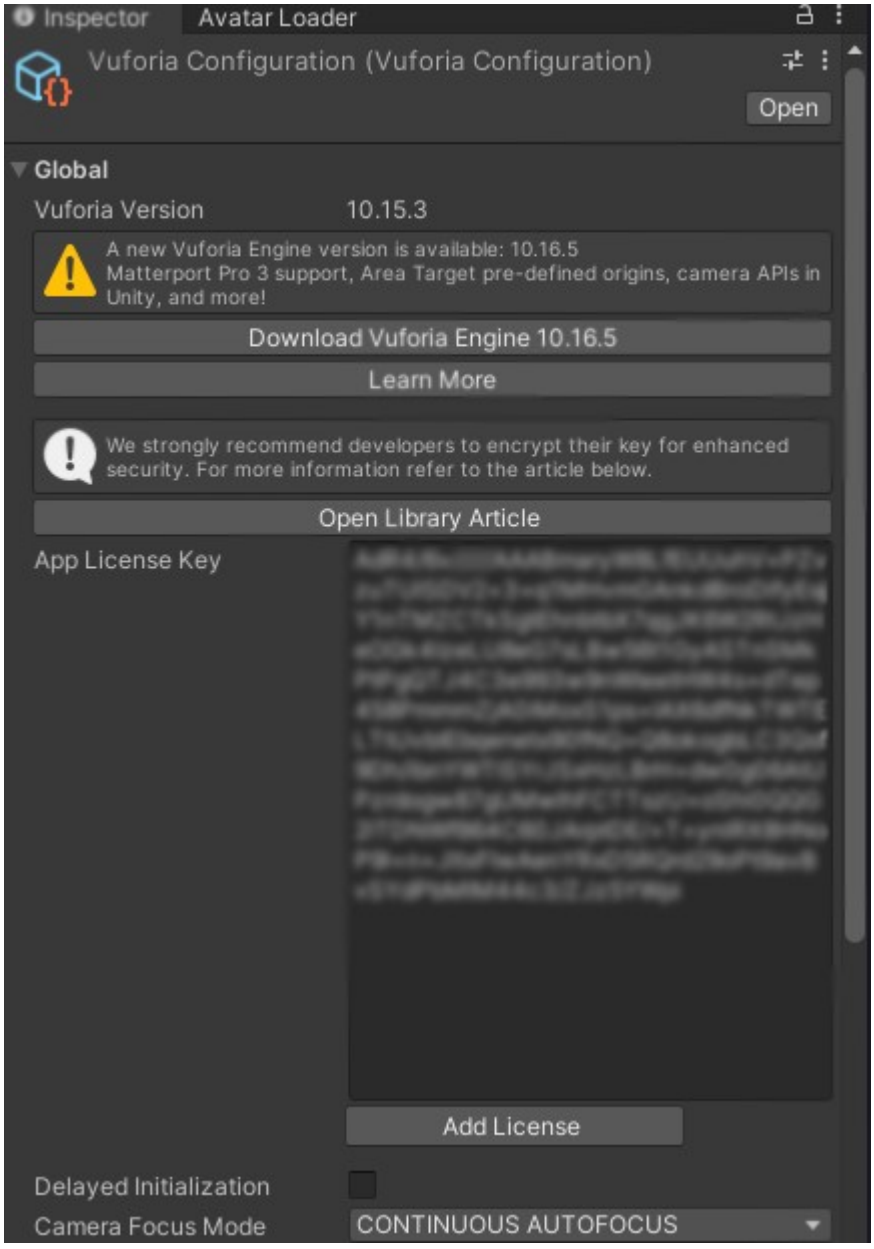

Figure 69 – Vuforia key

In addition to acquiring a license key, the developer must create a database within Vuforia's "Target Manager" feature, accessible via the "Develop" section. This database serves as a repository for marker information and related data. When creating the database, it is important to ensure that it is configured as a "device" database and a name is assigned to it.

Once the database is established, by clicking on its name a window that allows the developer to add markers is opened. Selecting an appropriate image to serve as an AR marker is a crucial step in the process. To be successful, an AR marker image should adhere to specific criteria: it should be in either PNG or JPG format, exhibit strong colour contrast, and be of a considerable size to ensure reliable recognition by the AR system.

After selecting the ideal image for the marker, it is necessary to download the database to the Unity editor. Following the download, the database must be correctly imported into the Unity project.

Subsequently, the user can start to incorporate the image target game object into the project. This game object is important in associating the AR experience with the markers that were defined earlier. Within the image target game object, there will be a script in the Inspector known as "Image Target Behaviour". It is necessary to configure this script by selecting the "from database" option and specifying the previously created database in the corresponding field. Additionally, the developer must designate the specific target that he wishes to associate with the image target.

Within the project's hierarchy, the avatar created for virtual tours must be located inside the image target game object. The avatar should align precisely with the chosen image marker, creating the illusion that the virtual object is coming from the real image.

Finally, the application can be built for the Android platform, allowing the user to experience and interact with the augmented reality environment directly from his own device.

In Figure 70, how the AR environment is structured on the Unity application is observed and in Figure 71, it is possible to observe what the user sees using the application on the AR target.

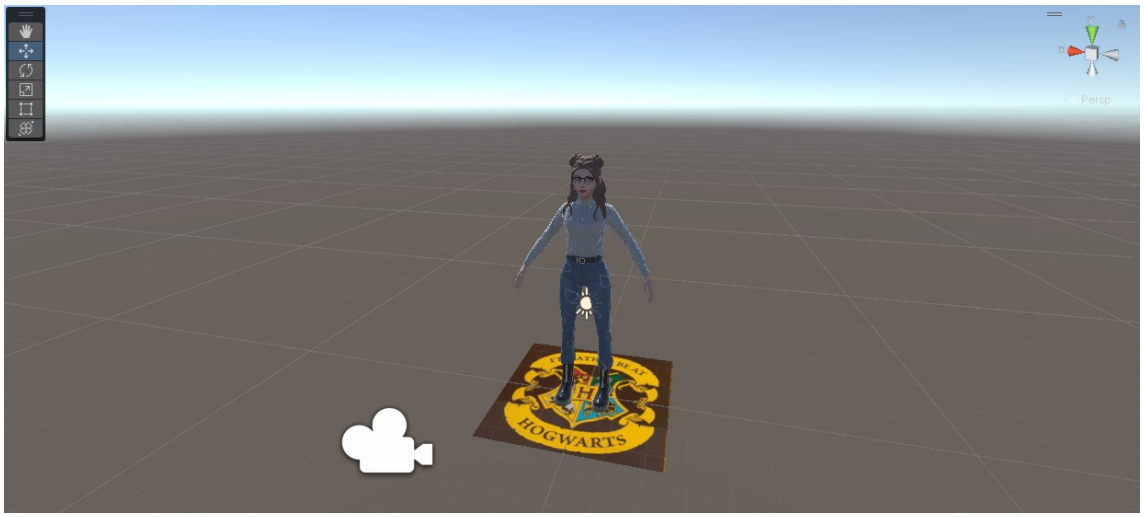

Figure 70 – AR environment (unity)

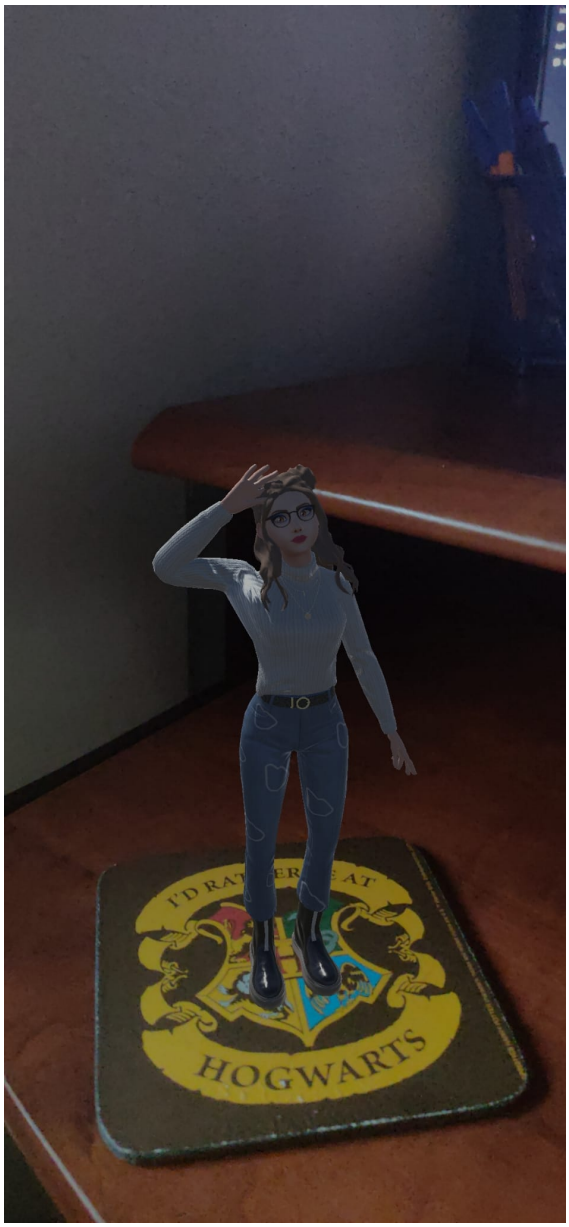

Figure 71 – AR environment (real time)

Conclusions

The primary focus of this dissertation centred on the development of an application designed to enhance guided tours. This enhancement was achieved through two key components: the utilisation of 360º images to showcase buildings to potential visitors and the integration of avatars to provide real-time assistance to visitors during these tours.

The utilisation of 360º images allowed for a dynamic and immersive virtual tour experience, enabling potential visitors to explore buildings from the comfort of their own devices. These images provided an in-depth, panoramic view of the buildings, offering a comprehensive understanding of the location.

Additionally, the application incorporated avatars to enhance the visitor's experience further. These avatars served as virtual guides, offering assistance and information as visitors navigated through the location. This real-time interaction between visitors and avatars added a special touch to the tour, making it more captivating. For the in-person guided tours, AR technology was utilized to elevate the quality, ensuring that visitors were engaged and immersed in their experience. As mentioned in Chapter 1, specifically in Section 1.2, avatars were initially introduced for demonstration purposes and were implemented in markers to allow visualization of the outcome. However, in subsequent phases, this technology can be used to create diverse elements that give life to the application.

The prototypes developed on this project was built upon the foundation of knowledge acquired throughout the computer engineering degree. All functionalities and features of these prototypes were meticulously explained in the preceding chapters, covering both concepts and detailed requirements. This comprehensive approach ensured that the prototypes met the expected standards and functionality.

To ensure the prototypes' successful implementation, continuous guidance and oversight were provided by the advisor, and during the process there was an additional collection of some potential users of the platform. This collaborative effort ensured that the project remained aligned with its objectives and met the requirements set forth.

In terms of design, the choice was made based on existing projects with a similar concept of 360º navigation in order to call up the familiar feeling. Furthermore, the creation of avatars was executed with simplicity and ease of use in mind, enabling seamless integration into the development platform. This approach ensured a more efficient workflow.

It's important to recognize the influence of user-friendly technology, as emphasized by TAM, which states that users are more likely to adopt a technology that is easy to use and useful. Therefore, a significant factor during the project's development was ensuring the ease of user interaction. Complexity diverts users' attention away from the platform, people prefer efficient solutions over spending time navigating complex features. While developing user interactions, the objective was to create a functional system that is user-friendly. To achieve this, the server-side functionalities involving the acquisition of new information, such as a user logging in or receiving a new message, were

designed with simplicity in mind. Instead of displaying buttons that refresh the connected users window, functionalities were implemented to automatically fetch the latest data whenever an event occurred, ensuring that users were informed.

In the visitor chat feature, where two platforms (desktop and mobile) coexist, the method of closing the conversation window varied. To reduce confusion, a message was incorporated within the conversation window, informing the users on how to close it. Furthermore, whenever new features were implemented, such as logout and scene transition, familiar icons were utilized. Using recognizable symbols, users instinctively understand how to navigate these features without the need for additional instructions, eliminating the need to teach visitors how to do some things.

While the dissertation successfully achieved its initial objectives, it is worth noting that some challenges were encountered, particularly in real-time interaction. These challenges arose from the simultaneous communication between visitors and administrators, with potentially multiple visitors and administrators engaging interactively with the user at the same time. This complexity required careful consideration and implementation to ensure smooth and effective real-time interactions within the application.

Future works

After the careful development of the application and a thorough evaluation, some features were identified that stood out in terms of importance to be implemented in future work, both for the 360-degree application and the AR application.

In the context of the 360-degree application, several features were pinpointed as potential future additions. Foremost among these ideas was the adaptation of the application for VR. This functionality would grant users an even more immersive experience, enabling them to explore virtual environments in 360 degrees with the sensation of being physically present in the location. Furthermore, the implementation of photogrammetry was considered a promising option for creating more detailed scenarios, offering users a more authentic experience. Photogrammetry is the process of taking multiple photographs from different viewpoints in order to generate detailed 3D models with accurate measurements.

Another functionality for the 360-degree application was the creation of more dynamic interactions within the virtual setting. This would include features such as the ability to turn lights on and off, move objects, and perform other interactive actions within the location.

As for the AR application, some ideas were outlined. One idea was the inclusion of virtual manuals associated with specific items in a house. This way users would have access to detailed information, tutorials, or utilization guides related to that specific item.

Furthermore, the inclusion of geolocation on the AR application would serve as a visual guidance to users, helping them navigate the physical environment more easily.

A crucial next step in the process is to conduct a more detailed analysis of the practical impact on users. Specifically, the aim is to measure user satisfaction during virtual tours with and without avatars, as well as physical tours with and without augmented reality. By focusing on these specific

aspects, the analysis aims to enhance the current understanding of user preferences but also provide well-informed decision-making.

Bibliography

Abi Tyas Tungga (2023). Bitbucket vs GitHub (Updated for 2023). https://www.upguard.com/blog/bitbucket-vs-github

Adam O'Grady (2018). GitLab Quick Start Guide: Migrate to GitLab for all your repository management solutions. https://books.google.pt/books?hl=en&lr=&id=jC59DwAAQBAJ&oi=fnd&pg=PP1&dq=gitlab&ots=9RGnjTAwVG&sig=bo2dW9Wg30tY9_PlagMJLJX6Lrg&redir_esc=y#v=onepage&q=gitlab&f=false

Alaeddin Nassani, Huidong Bai, Li Zhang, Mark Billinghurst (2021). ShowMeAround: Giving Virtual Tours Using Live 360 Video, May 08–13, 2021, Yokohama, Japan. DOI: https://doi.org/10.1145/3411763.3451555

Alvin Yong Chie Yew, Haji Mahran Dr Haji Morsidi and Jonathan H. Chan (2020). Augmented Reality Project Poster: Using Mobile Augmented Reality Application to Enhance Project Poster. In Proceedings of the 11th International Conference on Advances in Information Technology (IAIT2020). Association for Computing Machinery, New York, NY, USA, Article 31, 1–10. DOI: https://doi.org/10.1145/3406601.3406636

Angelov, V., Petkov, E., Shipkovenski, G., & Kalushkov, T. (2020). Modern Virtual Reality Headsets. 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). DOI: https://doi.org/10.1109/hora49412.2020.9152604

Antonín Šmíd (2017). Comparison of Unity and Unreal Engine. https://core.ac.uk/download/pdf/84832291.pdf

Briceno, L., & Paul, G. (2018). MakeHuman: A Review of the Modelling Framework. Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018), 224–232. DOI: https://doi.org/10.1007/978-3-319-96077-7_23

Cheng Xiao, Zhang Lifeng (2014). Implementation of Mobile Augmented Reality Based on Vuforia and Rawajali, 2014. DOI: https://doi.org/10.1109/ICSESS.2014.6933713

Chris Bradfield (2018). Godot Engine Game Development Projects. https://books.google.pt/books?hl=en&lr=&id=KMNiDwAAQBAJ&oi=fnd&pg=PP1&dq=godot+engine&ots=z10xQWIqV4&sig=1zF7AUAzYIG2x5C0NF8e1L5jJKk&redir_esc=y#v=onepage&q=godot%20engine&f=false

Chris Kemp and Brad Gyger (2013). Professional Heroku Programming. https://www.google.pt/books/edition/Professional_Heroku_Programming/QN1TaGKG0wUC?hl=en&gbpv=0

Cosentino, V., Luis, J., & Cabot, J. (2016). Findings from GitHub. Proceedings of the 13th International Workshop on Mining Software Repositories- MSR '16. DOI: https://doi.org/10.1145/2901739.2901776

Epic Games. (2017). Fortnite [macOS, Windows, PlayStation 4, Xbox One, iOS, Nintendo Switch,

Android, Xbox Series X/S, PlayStation 5]. https://www.fortnite.com/?lang=en-US

Facepunch Studios. (2018). Rust [macOS, Windows, PlayStation 4, Xbox One]. https://rust.facepunch.com/

Fred D. Davis (1985). A Technology Acceptance Model for Empirically Testing New End-User Information Systems. 1985. (PDF) A Technology Acceptance Model for Empirically Testing New End-User Information Systems (researchgate.net)

Guido van Rossum (1995). Python Tutorial. https://docs.python.org/3/tutorial/index.html

Guido van Rossum (2011). An Introduction to Python. https://books.google.pt/books/about/An_Introduction_to_Python.html?id=8-qkuAAACAAJ&source=kp_book_description&redir_esc=y

Hillmann, C. (2019). Comparing the Gear VR, Oculus Go, and Oculus Quest. Unreal for Mobile and Standalone VR, 141–167. DOI: https://doi.org/10.1007/978-1-4842-4360-2_5

Jarczyk, O., Gruszka, B., Jaroszewicz, S., Bukowski, L., & Wierzbicki, A. (2014). GitHub Projects. Quality Analysis of Open-Source Software. Social Informatics, 80–94. DOI: https://doi.org/10.1007/978-3-319-13734-6_6

John K. Haas (2014). A History of the Unity Game Engine. https://digital.wpi.edu/concern/student_works/tx31qh96p?locale=en

Lauren Landry (2020). WHAT IS HUMAN-CENTERED DESIGN?, 15 DEC 2020. https://online.hbs.edu/blog/post/what-is-human-centered-design

LI Na, HU Weihua (2012). Virtual reality applications in simulated course for tour guides. 2012. DOI: https://doi.org/10.1109/ICCSE.2012.6295385

Linden Lab. (2003). Second Life (v. 6.6.14.581101) [Windows, macOS, Linux). https://secondlife.com/

Luke Welling, Laura Thomson (2001). PHP and MySQL Web Development. https://books.google.pt/books?hl=en&lr=&id=G4dTRyvpfhoC&oi=fnd&pg=PA1&dq=php&ots=THMemDelmh&sig=Db6zE53Kk5kXX4C4ccKhFYVqSfM&redir_esc=y#v=onepage&q=php&f=false

Michael K. Lindell and Ronald W. Perry (1992). Behavioral Foundations of Community Emergency Planning, 1992. (PDF) Behavioral Foundations of Community Emergency Planning (researchgate.net)

Moroney, L. (2017). The Firebase Realtime Database. The Definitive Guide to Firebase, 51–71. DOI: https://doi.org/10.1007/978-1-4842-2943-9_3

Neil Middleton and Richard Schneeman (2013). Heroku: Up and Running: Effortless Application Deployment and Scaling. https://www.google.pt/books/edition/Heroku_Up_and_Running/IS4KAgAAQBAJ?hl=en&gbpv=0

Niantic. (2016). Pokémon Go [iOS, Android]. The Pokémon Company. https://pokemongolive.com/en

Ninja Theory. (2017). Hellblade: Senua's Sacrifice [PlayStation 3, Windows, Xbox One, Nintendo Switch, Xbox Series X/S]. https://www.hellblade.com/

Osman El-Said and Heba Aziz (2022). Virtual Tours a Means to an End: An Analysis of Virtual Tours' Role in Tourism Recovery Post COVID-19. Journal of Travel Research 2022. DOI: https://doi.org/10.1177/0047287521997567

Oufqir, Z., El Abderrahmani, A., & Satori, K. (2020). ARKit and ARCore in serve to augmented reality. 2020 International Conference on Intelligent Systems and Computer Vision (ISCV). DOI: https://doi.org/10.1109/iscv49265.2020.9204243

Paul DuBois (2000). MySQL. https://www.google.pt/books/edition/MySQL/JgFTUsIC0bUC?hl=en&gbpv=0

Paul Hemp (2006). Avatar-Based Marketing, June 2006. https://www.stanfordvr.com/mm/2006/hbr-avatar-based-marketing.pdf

Rare. (2018). Sea of Thieves [Windows, Xbox One, Xbox Series X/S]. https://www.seaofthieves.com/

Rasmus Lerdorf, Kevin Tatroe (2002). Programming PHP. https://books.google.pt/books?hl=en&lr=&id=7OjvOmol3CcC&oi=fnd&pg=PR9&dq=php&ots=1sWj6-a8w7&sig=VFDzA-kd4BHgQ0zm0FCR4P1B56k&redir_esc=y#v=onepage&q=php&f=false

Simon Gottschalk (2011). The Presentation of Avatars in Second Life: Self and Interaction in Social Virtual Spaces, 22 December 2011. DOI: https://doi.org/10.1525/si.2010.33.4.501

Simon Yuill and Harry Halpin (2006). Python. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb

Sonam Khedkar and Swapnil Thube (2017). Real Time Databases for Applications. International Research Journal of Engineering and Technology (IRJET). Volume: 04 Issue: 06. June -2017. https://www.irjet.net/archives/V4/i6/IRJET-V4I6401.pdf

So-mi Yoon (2014). Augmented Reality Comics and the Narrative, 2014. http://www.somiyoon.com/Yoon_Somi_MSDesign_Document_11-20-2014.pdf

S. Tilkov and S. Vinoski (2010). Node.js: Using JavaScript to Build High-Performance Network Programs, in IEEE Internet Computing, vol. 14, no. 6, pp. 80-83, Nov.-Dec. 2010, DOI: https://doi.org/10.1109/MIC.2010.145

Stephen Hillenburg. (2023). Nightmare in SquidVille [Windows, Linux]. https://spongebros.itch.io/nightmare-in-squidville

Steve Suehring (2002). MySQL Bible. https://www.google.pt/books/edition/MySQL_Bible/ZBKWA06r7mEC?hl=en&gbpv=0&bsq=MySQL%20Bible

S. L. Bangare, S. Gupta, M. Dalal, A. Inamdar (2016). Using Node.Js to Build High Speed and Scalable Backend Database Server. https://www.researchgate.net/profile/Sunil-Bangare-2/publication/301788361_Using_NodeJs_to_Build_High_Speed_and_Scalable_Backend_Database_server/links/57285d6c08aee491cb416ad6/Using-NodeJs-to-Build-High-Speed-and-Scalable-Backend-

Database-Server.pdf

Team Cherry. (2017). Hollow Knight [Windows, Linux, macOS, Nintendo Switch, PlayStation 4, Xbox One]. https://www.hollowknight.com/

Unknown Worlds Entertainment. (2018). Subnautica [macOS, Windows, PlayStation 4, Xbox One, NS, PlayStation 5, Xbox Series X/S]. https://unknownworlds.com/subnautica/

Wizbane. (2022). The Legend Of Lumina. https://wizbane.itch.io/the-legend-of-lumina