

POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria del Cinema e dei Mezzi di Comunicazione

Tesi di Laurea Magistrale

**Sviluppo di una simulazione interattiva per la
gestione e navigazione di una collezione d'arte**



**Politecnico
di Torino**

Relatori

Prof.ssa Tatiana Mazali

Prof. Andrea Bottino

Dott. Francesco Strada

Candidato

Vittorio Elia

Anno Accademico 2021-2022

Abstract

Il presente elaborato descrive il progetto Meta Gallery, un sistema composto da un'applicazione simulativa realizzata tramite il game engine Unity, mediante le librerie grafiche per la distribuzione web, la quale dialoga con un server remoto per l'esplorazione di ambienti virtuali popolati dalle gallerie d'arte degli utenti.

L'obiettivo principale consiste nella creazione di un'infrastruttura online per permettere ai collezionisti d'arte di creare un biglietto da visita virtuale modulabile e costantemente aggiornabile, la cui gestione avviene tramite un'interfaccia di semplice utilizzo. La possibilità di controllo, oltre che di navigazione, di ambienti 3D e di visitare le gallerie d'arte altrui rende il sistema l'unica risposta sul mercato a questo tipo di requisito offrendo una gestione dinamica delle opere e un'esperienza più intima e contemplativa dell'esplorazione delle stesse. Un ulteriore aspetto di fondamentale importanza per i galleristi e gli enti museali è la dislocazione delle opere in differenti località, anche remote, con l'impossibilità di esporre e rendere visibile integralmente la propria collezione.

L'analisi dello sviluppo del progetto copre tutte le fasi dal testing delle potenzialità tecnologiche di vari tool alla creazione di un sistema dimostrativo comprensivo delle funzionalità principali ideate.

Grazie alla collaborazione con l'azienda Motion Pixel SRL, promotrice del progetto a seguito delle proprie indagini di mercato, si sono affrontate varie soluzioni, le quali hanno portato alle scelte di natura tecnica e di design migliori per il prodotto finale, analizzando per ognuna le criticità e le potenzialità.

La conclusione del lavoro rappresenta il primo passo verso un sistema completo ed altamente personalizzabile, la sua composizione nella versione dimostrativa proposta è stata progettata per essere modulabile e futuribile sia a livello di applicativo simulativo, sia a livello di struttura online per la gestione degli utenti e delle relative opere.

Indice

1	Introduzione.....	1
1.1	Motivazione.....	1
1.2	Obiettivo	2
1.3	Architettura del sistema.....	2
1.4	Motion Pixel SRL.....	3
1.5	Struttura dell'elaborato	4
2	Stato dell'arte.....	5
2.1	3DVista.....	6
2.2	ArtSteps.....	8
2.3	ART.SPACES.....	11
2.4	Space.....	16
2.5	Room.....	19
3	Requisiti.....	21
3.1	Processo decisionale riguardante il DB e il server web	23
3.2	Processo decisionale riguardante il game engine	25
3.3	Navigazione e visualizzazione quadri.....	29
4	Portale web per la gestione delle collezioni	30
4.1	Struttura e logica del database.....	31
4.2	Features del portale web	34
4.2.1	Configurazione della connessione al database	36
4.2.2	Homepage	37
4.2.3	Registrazione di un nuovo utente	39
4.2.4	Login, Profile, Logout	41
4.2.5	Indicizzazione immagini e cancellazione	43
4.2.6	Gestione degli inviti ad altri utenti.....	44
4.2.7	Upload dei quadri ed elaborazione dei dati.....	45
4.3	Dialogo con l'applicazione simulativa.....	46
4.3.1	Script di richiesta della lista di quadri di un utente	47
4.3.2	Script di login e di richiesta dell'elenco di amicizie di un utente.....	47
5	Applicazione di simulazione virtuale WebGL.....	48
5.1	Architettura del progetto.....	48
5.2	Struttura grafica della scena Main Menu	49
5.2.1	Gerarchia degli oggetti.....	50
5.2.2	Interfaccia utente ed elementi utilizzati.....	53
5.3	Features ed algoritmi del menu principale.....	55

5.3.1	Gestione dell'interfaccia e degli elementi 2D	55
5.3.2	Game Manager, Start e caricamento asincrono	58
5.3.3	Pannelli per il login e logout	59
5.3.4	Pannello delle amicizie	60
5.4	Menu di pausa	62
5.5	Struttura grafica della scena House.....	63
5.5.1	Gerarchia degli oggetti.....	63
5.5.2	Gestione della navigazione in prima persona.....	65
5.5.3	Feedback grafici e interfaccia 2D.....	66
5.6	Features ed algoritmi della scena House	68
5.6.1	Database Manager per il download della lista dei quadri di un utente	70
5.6.2	Pics Manager per il posizionamento in scena dell'immagine su una cornice ..	70
5.6.3	Frames Manager per l'assegnazione di una cornice al quadro.....	72
5.6.4	Puntamento, navigazione e Spots Manager	72
6	User Test.....	73
6.1	Gallerista	74
6.2	Contabile.....	76
6.3	Proprietario galleria.....	77
6.4	Possibile acquirente.....	78
7	Conclusioni e lavori futuri.....	79
	Bibliografia	81

Indice delle figure

Figura 2.1: Ambiente di navigazione su 3DVista	7
Figura 2.2: Interfaccia informativa di un quadro su 3DVista	8
Figura 2.3: Collezione visitabile su ArtSteps	9
Figura 2.4: Creazione collezione su ArtSteps	10
Figura 2.5: Navigazione in ART.SPACES	12
Figura 2.6: Visualizzazione di un quadro in ART.SPACES.....	13
Figura 2.7: ART.DEPOT: Caricamento di un quadro su ART.SPACES.....	13
Figura 2.8: Creazione di un'exhibition in ART.SPACES	14
Figura 2.9: Scelta dell'ambientazione virtuale in ART.SPACES.....	14
Figura 2.10: Creazione di un'attività per i contatti in ART.SPACES.....	15
Figura 2.11: Utilizzo dell'app ART.AUGMENTED [9]	15
Figura 2.12: Configurazione di un device esterno su Space.....	17
Figura 2.13: Lobby di ingresso su Space	17
Figura 2.14: Navigazione attraverso un club di sci su Space	18
Figura 2.15: Flusso di creazione di un nuovo spazio virtuale su Space.....	18
Figura 2.16: Aggiunta di un prodotto all'ambiente virtuale su Space	18
Figura 2.17: Utilizzo della visualizzazione di modelli 3D in Room	20
Figura 2.18: Utilizzo della AR in Room	20
Figura 2.19: Spazio virtuale ricreato su Room	20
Figura 3.1: Creazione HTTP Request su Unreal Engine.....	27
Figura 4.1: Server del Database dall'interfaccia di phpMyAdmin.....	31
Figura 4.2: Tabella accounts	32
Figura 4.3: Tabella friends.....	33
Figura 4.4: Tabella images.....	33
Figura 4.5: Homepage.....	38
Figura 4.6: Form di registrazione	39
Figura 4.7: Form di login.....	41
Figura 4.8: Feedback in testa alla pagina	41
Figura 4.9: Pagina Profile.....	42
Figura 4.10: Pagina imageList	43
Figura 4.11: Invito ad un amico	44
Figura 4.12: Esempio di upload fallito.....	45
Figura 4.13: Form di richiesta dati dei quadri in POST.....	47
Figura 5.1: Gerarchia oggetti Main Menu	51
Figura 5.2: Oggetti della scroll view delle amicizie.....	52
Figura 5.3: Pannelli del Main Menu	54
Figura 5.4: Main Menu Manager.....	57
Figura 5.5: Caricamento asincrono della scena alla pressione del tasto Start.....	59
Figura 5.6: Componenti Scroll View	62
Figura 5.7: Modello della villa.....	63
Figura 5.8: Gerarchia oggetti scena House	64
Figura 5.9: First Person Controller	64
Figura 5.10: Gerarchia oggetti dei punti di spawn dei quadri.....	65
Figura 5.11: Inizio navigazione scena House	66
Figura 5.12: Visualizzazione quadro, navigazione sull'oggetto standUpPosition.....	67
Figura 5.13: Architettura scena House	69
Figura 5.14: Flusso di allocazione quadri.....	71

Indice dei diagrammi

Diagramma 1: Architettura del sistema Meta Gallery	22
Diagramma 2: Funzionamento architettura web	30
Diagramma 3: Tabelle database	32
Diagramma 4: Pagine pubbliche del portale web.....	34
Diagramma 5: Componenti di back-end del portale web.....	35
Diagramma 6: Script php richiamabili dall'applicativo WebGL	46
Diagramma 7: Architettura scene del progetto Unity	49
Diagramma 8: Elementi grafici del menu principale	50
Diagramma 9: Diagramma delle classi della scena Main Menu	55
Diagramma 10: Diagramma delle classi scena House.....	68
Diagramma 11: Casi d'uso dei test.....	73

1 Introduzione

1.1 Motivazione

L'interesse nei confronti della ricostruzione di ambienti virtuali fuori dall'ambito dell'intrattenimento è cresciuto esponenzialmente negli ultimi anni, legandosi a idee, tecnologie e strutture già affermate quali contenuti multimediali tradizionali e virtual tour, ma anche a concetti più giovani nel panorama tecnologico come *metaverso*¹, NFT, AR, VR/360°.

Il fascino di queste piattaforme o tecniche colpisce figure aziendali di natura informatica o con una forte presenza sul Web ma risulta anche molto forte in singoli utenti e persone di differenti contesti d'origine. Se da un lato è vero che parte del pubblico generalista di Internet trovi nella realtà virtuale applicata alla riproposizione e navigazione di ambienti reali una scarsa utilità senza uno scopo preciso, un altro lato altrettanto forte vede una grande potenzialità del mezzo arrivando a richiederlo, costruirlo o appoggiarsi a piattaforme già esistenti in ambiti molto variegati, anche lontani dal mondo tecnologico.

L'azienda Motion Pixel SRL ha raccolto sin dai primi mesi del 2022 numerosi attestati di interesse nei confronti della realizzazione di progetti trasversali con componenti audiovisive e virtuali come simulatori, ambienti interattivi di supporto alla narrazione, stanze nel Metaverso per esercitare alcuni lavori con il pubblico come l'insegnamento. In particolar modo tramite la collaborazione con figure professionali nel mondo dei beni culturali e del collezionismo è nato il progetto Meta Gallery per la creazione di una piattaforma multiutente di gestione di una propria galleria di quadri, il tutto corroborato da feedback positivi di galleristi, artisti e semplici appassionati.

Le necessità maggiori analizzate da Motion Pixel per quanto riguarda la creazione di una galleria d'arte virtuale risultano essere la navigazione in un ambiente realistico e personale dove esporre e contemplare le proprie opere creando un database vivo per la gestione della collezione, la portabilità immediata tra dispositivi per un accesso semplice e da qualunque luogo, la possibilità di visitare le gallerie altrui e una gestione semplice dell'aggiornamento del database.

Il target principale risulta essere costituito proprio dai proprietari delle collezioni, i quali posseggono numerose opere di natura simile o molto diversa con una forte attività in entrata e uscita, oltre alla necessità di presentare un biglietto da visita della propria raccolta. I quadri sono spesso in località diverse o presentano difficoltà di esposizione ed è normale per i collezionisti dedicare una parte del proprio tempo (o di quello dei collaboratori) alla gestione dei database e all'archiviazione di dati.

Secondariamente le caratteristiche di Meta Gallery si adattano alla creazione di una community di appassionati, così come agli enti d'arte e strutture museali, specialmente per ragioni di mercato e per l'interesse mostrato dalle compagnie locali e nazionali nei confronti di prodotti VR. Sono inoltre proprio i musei le strutture con le maggiori difficoltà espositive, molto spesso con una grande quantità di opere immagazzinate e non accessibili al pubblico.

¹ Inteso come uno spazio virtuale 3D, multiutente e condiviso in cui le persone possano convivere [32]

1.2 Obiettivo

L'obiettivo estetico è la realizzazione di un ambiente intimo, fornendo la sensazione di trovarsi nella casa del collezionista piuttosto che ad una mostra in un museo.

Come descritto nel **CAPITOLO 2** la base di partenza per il lavoro di tesi è stata una demo dimostrativa creata da Motion Pixel SRL, la quale, non essendo interattiva, ha rappresentato solo una prima traccia estetica per quella che è stata poi la realizzazione da zero di un sistema formato da un applicativo *WebGL*² creato con *Unity* [1] ed una piattaforma web che permette il dialogo con un database SQL.

L'analisi dello stato dell'arte è fondamentale per comprendere i bisogni del mercato e nel caso di Meta Gallery mostra la mancanza di una soluzione che soddisfi i bisogni dei collezionisti. La componente più forte in tal senso è la gestione di un ambiente 3D sincronizzato al database di opere del proprietario, automatizzando i cambiamenti nel mondo virtuale in base a ciò che è archiviato in quel dato momento. Il target dei tool principali esistenti è alquanto generalista, motivando una completa e facile gestione all'interno del sito web con grandi limitazioni in termini di creazione degli ambienti. Il mercato presenta ambientazioni totalmente realistiche nel caso dei virtual tour tradizionali oppure totalmente irrealistiche nel caso di design del mondo 3D da parte dell'utente.

L'interesse da parte dei target mostrato per Meta Gallery segue quello mostrato per il metaverso, il quale richiede la navigazione attraverso luoghi realistici che necessitano di tridimensionalità e di gestione grafica sia delle strutture che degli oggetti presenti al loro interno. Questo collegamento indiretto mostra anche l'importanza di avere un sistema di gestione del database, modulare e strutturato, il quale possa prevedere la verifica del possesso delle opere oltre a garantire l'accesso e la visualizzazione delle gallerie a un numero ristretto di persone. La natura di Meta Gallery è in un certo senso commerciale di fronte alle richieste dei collezionisti e meno legata alla semplice riproduzione e rappresentazione, mentre sul mercato l'aspetto esplorativo e gestionale è sempre fortemente legato e soprattutto sequenziale: una volta impostata e ricreata la scena è necessario ripercorrere tutto il flusso di lavoro per modificare gli elementi presenti nel mondo virtuale.

1.3 Architettura del sistema

L'analisi dello sviluppo del progetto copre tutte le fasi dal testing delle potenzialità tecnologiche di vari tool, alla creazione di un sistema dimostrativo comprensivo delle funzionalità principali ideate. Prima della descrizione del prototipo dimostrativo di Meta Gallery, infatti, l'elaborato analizza il lungo periodo di preproduzione attraverso il quale il design è trascorso.

Partendo dai requisiti di progetto, il processo decisionale ha riguardato ogni componente del progetto, in particolar modo le scelte riguardanti il motore di database, il servizio di hosting web, il *game engine*³, le librerie grafiche e il design dell'esperienza utente. Tali decisioni sono procedute in parallelo in quanto l'una dipende dall'altra e per ogni componente sono stati eseguiti molti test per provare le opzioni estetiche e funzionali.

A seguito dei test la progettazione del sistema è stata definitivamente architettata sulla presenza di un portale web e di un applicativo di simulazione 3D WebGL.

² Librerie di grafica 3D per i browser

³ Ambiente di sviluppo per la gestione del render grafico, delle interazioni e degli input

Il flusso completo di operazioni disponibili per l'utente prevede l'accesso o registrazione alla piattaforma tramite il portale web, il quale fornisce funzionalità di upload dei quadri, invito ad altri utenti per la visita della galleria, controllo e modifica della collezione. Le pagine del sito web dialogano con il DB attraverso gli script del server, il quale contiene anche i codici per accogliere le richieste HTTP da parte dell'app simulativa. Il client WebGL è organizzato su un progetto Unity a due scene: Main Menu e House. La prima contiene la GUI 2D del menu principale per effettuare il login ed avviare la navigazione attraverso la propria collezione o quella di un amico. La seconda è costituita da un'ambientazione 3D e permette la navigazione in prima persona attraverso le stanze di una villa con un movimento a checkpoint (puntamento verso un'area di spostamento), la possibilità di visualizzare i quadri da vicino a dimensione reale (come definita in fase di caricamento) ed aprire un pannello 2D con le informazioni dell'opera.

1.4 Motion Pixel SRL

Motion Pixel è una realtà che si occupa di comunicazione video da oltre 15 anni, lavorando per aziende, startup, enti pubblici e cooperative.

Fondata da Stefano Sburlati (regista ed esperto di comunicazione audiovisiva) raccoglie diversi professionisti con un taglio multidisciplinare. Fra gli altri, Motion Pixel ha lavorato con: *Avio SpA, Alfa Romeo, Comune di Torino, Engim Internazionale, LVIA, Expo 2015, Mercedes Autocentaurio, Ruta40 Tour Operator, Jeep, LVIA, 5T, Tristone e Whirlpool* [2].

Nel corso del 2020 Motion Pixel è divenuta una società di tipo SRL, in accordo con il maggior volume di lavoro e per abbracciare migliori opportunità di sviluppo e crescita. L'attrezzatura hardware è in costante aggiornamento ed evoluzione, attualmente per VR e 360°, Motion Pixel dispone di diverse camere top 8K fra cui Insta360Pro 1, Insta 360Pro 2, Kandao Qoocam 8K.

Nonostante il focus sul videomaking, l'azienda ha iniziato l'ampliamento del proprio focus ai concetti di metaverso e simulazione virtuale, ciò partendo dai numerosi lavori di video immersivo effettuati. Infatti, spesso i target interessati alle esperienze VR a 360° esplorano anche la simulazione di ambienti 3D per la riproposizione di spazi fisici e reali.

Motion Pixel si è occupata di numerosi lavori di integrazione del video con l'interattività offerta dai motori di gioco, con un sempre maggior numero di richieste per la creazione di esperienze di natura totalmente virtuale, sfruttando l'esperienza ottenuta dall'azienda in termini di manipolazione dei media per la simulazione di luoghi tridimensionali relativi sia al mondo industriale/commerciale, sia a quello dell'arte e dell'intrattenimento.

1.5 Struttura dell'elaborato

Il lavoro di analisi del sistema Meta Gallery è suddiviso in 7 capitoli in cui sono trattati tutti gli aspetti relativi al progetto: dalla fase di preproduzione al processo di sviluppo con relative conclusioni.

Nel [Capitolo 2](#) viene presentata l'analisi dello stato dell'arte, descrivendo la situazione attuale nel mercato concorrente all'applicazione valutando punti di forza dei principali strumenti di creazione online di tour e ambienti virtuali, oltre che di spazi correlati al concetto di metaverso, rilevando quali caratteristiche di Meta Gallery non trovano risposta nei prodotti già esistenti, ovvero quali bisogni risultano non appagati per i target di riferimento

Nel [Capitolo 3](#) vengono esplicitati i requisiti tecnici e funzionali dell'app, descritta la struttura (divisa in un applicativo WebGL di simulazione 3D e in un portale web comprensivo di sito pubblico e database) e il processo decisionale portato avanti durante tutta la fase di preproduzione motivando le scelte compiute in termini di tecniche e tecnologie utilizzate, confrontando tutte le alternative prese in considerazione.

Nel [Capitolo 4](#) viene trattata la progettazione lato server con l'analisi dell'interfaccia e dei rapporti funzionali del sito web navigabile pubblicamente, oltre alla descrizione del back-end del portale attraverso la configurazione del database e dei codici che ne regolano la comunicazione ed il flusso di informazioni.

Nel [Capitolo 5](#) è presente l'esposizione dell'applicazione creata con Unity per l'esecuzione web partendo dalla struttura del progetto diviso principalmente in due scene, ovvero il menu principale e la scena simulativa principale. Vengono descritti i componenti visivi e grafici delle due, le relazioni tra le classi e i codici che regolano l'interazione tra gli oggetti virtuali, nonché l'invio delle richieste web al server per il dialogo con il portale web e quindi con il database.

Nel [Capitolo 6](#) vengono descritti gli User Test effettuati per provare i vari componenti dell'applicazione attraverso degli scenari d'uso configurati per simulare l'utilizzo del sistema da parte dei target di riferimento.

Nel [Capitolo 7](#) viene analizzata la situazione attuale del progetto prototipale attraverso le potenzialità, i limiti, l'attinenza alle richieste iniziali, gli step futuri e la prontezza del sistema ad essere aggiornato ulteriormente.

2 Stato dell'arte

Il mercato dei tool online simulativi nell'ambito di mostre, gallerie d'arte, musei e installazioni offre numerose alternative per il target generalista, tutte derivate di un nucleo fondamentale: il Virtual Tour.

I tour virtuali, infatti, sono l'espressione più comune della VR e l'anello di congiunzione più importante tra essa e gli enti museali, ma più in generale con le attività di stampo turistico. L'utilizzo maggiore prevede la ricostruzione realistica, tramite fotogrammetria o più semplicemente di foto scattate in loco, degli ambienti con la possibilità per l'utente di navigare attraverso i luoghi e visualizzarli preferibilmente a 360° gradi. Le possibilità di movimento sono solitamente limitate, con una navigazione su binari attraverso i punti di interesse evidenziati dagli elementi 2D di interfaccia.

Si tratta in questo caso di riproduzione di ambienti statici, immutabili nel tempo o quantomeno in un periodo più o meno lungo, il cui scopo è la promozione degli stessi luoghi o la visualizzazione di aree non accessibili al grande pubblico.

Uno dei sistemi più completi e popolari per la creazione di tour virtuali guidati è *3DVista*, un insieme di strumenti per la realizzazione di percorsi navigabili attraverso un ambiente costituito principalmente da foto o video 360°. La sua analisi è di forte importanza per due motivi: è uno dei tool di questo tipo più potenti in commercio, nonché quello che fornisce la resa grafica più apprezzabile, ma soprattutto tramite *3DVista* è stato creato l'ambiente di prova dimostrativo di *Meta Gallery*, un percorso attraverso render 360° del modello di una villa popolato di quadri. Il prototipo è stato utilizzato dall'azienda *Motion Pixel* durante la fase di ideazione del progetto e di indagine, sondando il terreno e chiedendo pareri a galleristi ed esperti d'arte nella zona di Torino.

I prodotti più di interesse da analizzare per lo sviluppo di *Meta Gallery* sono quelli destinati alla riproduzione di mostre e installazioni, tool che permettono di creare ambienti virtuali non reali dove predisporre degli oggetti. Sono solitamente piattaforme online le quali consentono sia la creazione ed il design della propria installazione, sia la pubblicazione della stessa, alimentando una community interna e permettendo l'integrazione dentro altri siti web.

Uno dei più completi tra questi strumenti è *ArtSteps*, una piattaforma online che consente la facile creazione di un ambiente 3D ed il posizionamento delle opere al suo interno ed al contempo presenta una propria community volenterosa di progettare mostre virtuali o ricreare gallerie famose esistenti.

Anche la piattaforma *ART.SPACES* si occupa di gestire delle gallerie virtuali con un approccio più formale di *ArtSteps* e a tratti anche più commerciale, in cui i protagonisti diventano venditore e acquirente, invece che la community. Questi strumenti si differenziano dagli scopi di *Meta Gallery* per diverse motivazioni, ma in particolare non permettono una gestione automatizzata e algoritmica della presenza delle opere nel mondo 3D; risultano, invece, essere degli editor di ambientazioni VR semplici e adatti a qualsiasi pubblico.

Un ulteriore ambito di ricerca utile per comprendere le abitudini e l'andamento dei target interessati al metaverso riguarda le app di creazione di spazi virtuali (spesso multiutente) non per forza legati al mondo dell'arte. *Motion Pixel* ha raccolto numerose richieste di approfondimento su questi temi da parte di aziende sempre più interessate alla creazione di spazi remoti per conferenze, lezioni, esposizioni e intrattenimento.

Tra gli strumenti più interessanti in tal senso, le piattaforme *Space* e *Room* permettono la creazione di spazi più o meno realistici: nel primo caso l'interfaccia e la struttura segue un vocabolario tipico di Internet permettendo agli utenti di ritrovarsi con facilità all'interno di eventi online; nel secondo la trasposizione 3D del mondo reale viene portata a un livello più alto in termini di realismo e professionalità.

Questi strumenti rappresentano un'importante traccia per il design di Meta Gallery in quanto comunicano con un target in parte comune, oltre che per il semplice fatto di essere ambienti virtuali simulativi. Studiare questi sistemi significa analizzare ciò che sta funzionando e cosa meno, oltre che comprendere quale linguaggio usare in contesti differenti come quello del business e dell'intrattenimento.

2.1 3DVista

La piattaforma online 3DVista [3] è composta da numerosi tool il cui principale risulta essere *Virtual Tour PRO*, ovvero il cuore della creazione degli ambienti virtuali percorribili dagli utenti. Il sito web fornisce da subito un forte focus sulla realtà virtuale mostrando come il suo utilizzo sia perfetto per la riproduzione di percorsi guidati e di punti panoramici, ma anche per l'E-Learning, ovvero la creazione di materiale didattico per la simulazione di contesti reali come quello medico.

Gli strumenti presenti permettono di operare su immagini (o video) a 360° disponendole in serie per costituire un percorso. La navigazione procede quindi attraverso sfere di grandi foto o render, potendo volgere lo sguardo in tutte le direzioni.

3DVista consente di disporre degli elementi guida in precisi punti dell'immagine attraverso i propri strumenti, ovvero interfacce 2D che rendono semplice la progettazione di un tour e la sua integrazione in siti web esterni.

Per realizzare il prototipo dimostrativo [4] è stato utilizzato proprio 3DVista partendo comunque da un modello 3D (lo stesso utilizzato poi nello sviluppo di Meta Gallery), ovvero una grande villa [5] al cui interno posizionare i quadri sulle pareti. Vista la natura fotografica dei tour previsti dall'applicazione sono stati realizzati numerosi render dell'ambiente 3D, utilizzando il software *Blender* [6] tramite il motore di rendering⁴ *Eevee* in modo tale da ottenere un risultato ottimo a livello grafico e appetibile visivamente. La gestione dell'illuminazione su Blender è infatti realistica e di altissima qualità, specie in confronto con qualsiasi libreria grafica per il web le cui prestazioni sono sempre ridotte, soprattutto in termini di illuminazione e ombre.

Il risultato è la navigazione attraverso le realistiche immagini della villa ottenute su Blender, le cui mura sono spoglie e i mobili o altri oggetti assenti, come mostrato in Figura 2.1.

La disposizione delle opere risulta invece configurata su 3DVista e con facilità sono state inserite sulle mura le opere, ovvero le immagini 2D "renderizzate", e gestito il loro orientamento per poter essere adagiate sulle pareti. L'interfaccia creata ha una natura bidimensionale evidente, con le immagini non appartenenti al mondo 3D e che non partecipano all'illuminazione dell'ambiente, si nota ad occhio come facciano parte di una *GUI*⁵ e non siano realmente poggiate sulle pareti.

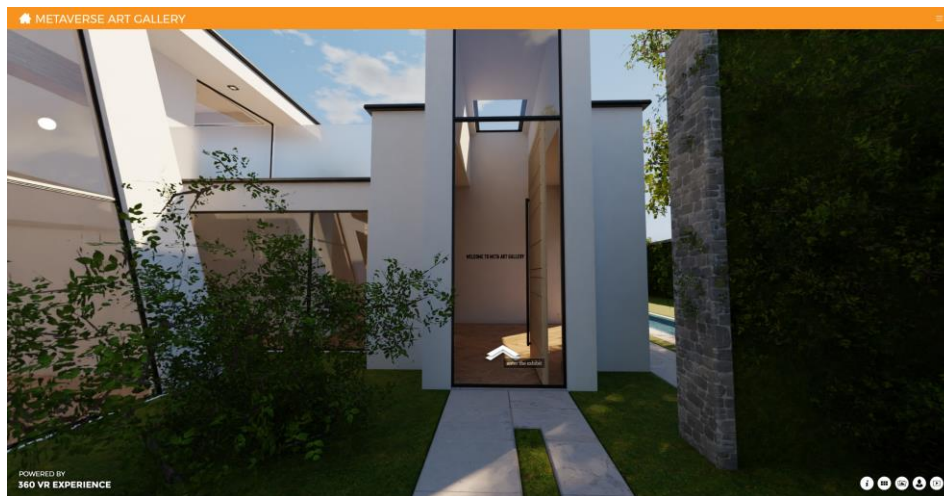
Attraverso la navigazione a checkpoint, ovvero il passaggio tra una sfera e l'altra con una dissolvenza tra le due, è possibile muoversi tra le stanze ed avvicinarsi ai quadri. Cliccando sulle opere, una volta posizionatisi nella zona di riferimento, è disponibile una visualizzazione più vicina: viene aperta un'interfaccia costituita da un pannello la quale

⁴ Software di gestione delle librerie grafiche e degli algoritmi di simulazione per creare dei render

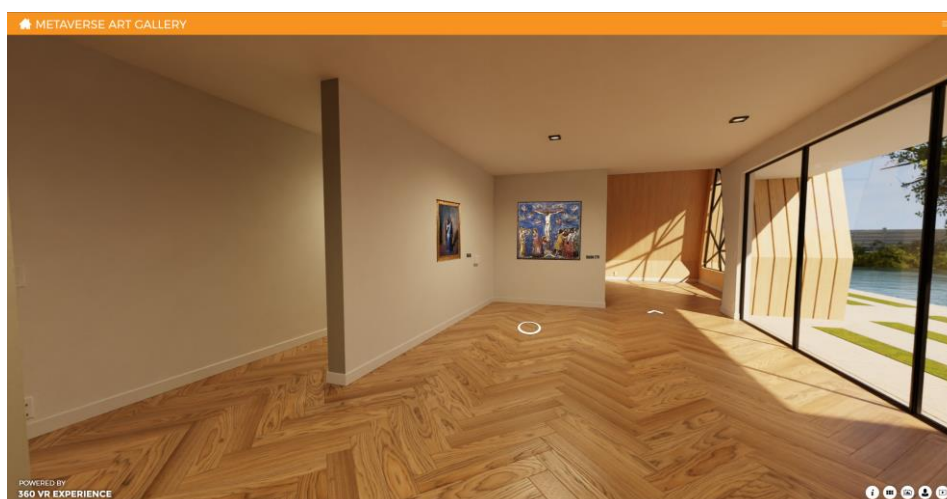
⁵ Graphical User Interface

contiene le informazioni del quadro, ovvero il nome, l'autore e ulteriori descrizioni (Figura 2.2).

3DVista risulta essere uno strumento efficace per la creazione di tour virtuali di natura tradizionale ma con una potenza grafica quasi imbattibile a parità di facilità d'utilizzo. Non consente però alcuna gestione degli input e delle interazioni nel mondo 3D attraverso un motore grafico e né tantomeno la sincronizzazione con un DB esterno. Il tool si adatta alla perfezione alla gestione di ambienti statici; la fluidità di navigazione e generale gradevolezza lo rende ottimo per esperienze dimostrative, contemplative e guidate. Tramite pochi click attraverso la demo estetica di Meta Gallery si ha subito un'idea visiva dell'app e della sua natura, ma senza una descrizione esterna o presentazione prima dell'utilizzo è impossibile percepire tutta la natura gestionale del sistema attraverso le componenti relative al database online.



(a)



(b)

Figura 2.1: Ambiente di navigazione su 3DVista



Figura 2.2: Interfaccia informativa di un quadro su 3DVista

2.2 ArtSteps

Lo strumento più completo in termini di facilità d'utilizzo ed esperienza utente per la creazione e condivisione di una mostra navigabile è ArtSteps [7].

La piattaforma è di natura user-friendly, ovvero il suo design è pensato per rendere la vita facile agli utenti che vogliono utilizzare gli strumenti disponibili. Dall'accesso alla creazione e pubblicazione, nonché alla visita alle altre *exhibition* (Figura 2.3), l'applicazione guida l'utente attraverso le proprie funzioni con una struttura grafica gradevole e semplice. La presenza della community è fondamentale, così come per altre piattaforme simili, concentrandosi soprattutto su un target di appassionati pronti a ricreare le loro gallerie o esposizioni preferite.

L'elemento che contraddistingue ArtSteps rispetto alla concorrenza (la quale si avvale di solito di preset di ambienti) è la possibilità di creare le proprie stanze dove porre le opere: tramite un editor 3D è infatti possibile disporre elementi semplici come mura e porte a proprio piacimento per disporre successivamente i quadri come immagini da adagiare su tali superfici, come mostrato in Figura 2.4.

Vengono inoltre posizionati dei punti di interesse fissi per definire dei luoghi di osservazione prefissati secondo una camera virtuale di cui è precisata la posizione e l'orientamento con la possibilità di aprire un pannello di interfaccia con la descrizione dell'opera o del punto di interesse.

La navigazione nelle gallerie d'arte di ArtSteps è libera trasportando il personaggio tra le aree calpestabili puntate con il mouse o muovendosi tramite input da tastiera. Da subito si nota la povertà grafica e la somiglianza tra le varie *exhibition*; infatti, le opportunità offerte dall'editor sono fortemente limitate, dando più che altro la possibilità di descrivere gli spazi e le distanze ma con un motore grafico il più limitato possibile in modo da garantire fluidità e velocità performativa in ogni contesto. Come per i competitor i quadri non sono integrati graficamente nell'ambiente ma risultano come texture 2D appoggiate sulle pareti. L'obiettivo della piattaforma è quello di rendere il più semplice e veloce possibile la creazione della galleria per alimentare una community di stampo mainstream, appassionata e volenterosa di mettere insieme le proprie opere preferite. Alcuni galleristi inoltre utilizzano le piattaforme come ArtSteps per integrare all'interno del proprio sito la *exhibition* creata ma sempre con lo scopo di venire a vedere con i propri

occhi le opere; la natura grafica di questi siti è infatti non simulativa, con l'intenzione di presentare al pubblico un'anticipazione della mostra in questione, fornendo un'idea riguardo lo stile e natura della stessa.

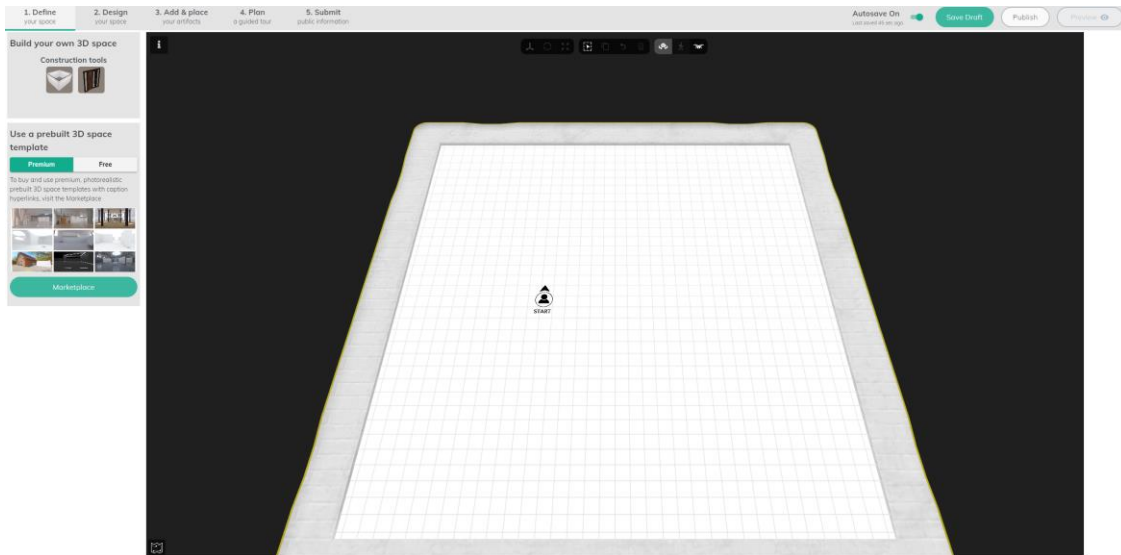
Non vi è alcuna gestione *live* dell'elenco dei quadri, costringendo gli eventuali proprietari delle opere ad azioni di indicizzazione tradizionale, utilizzando questi strumenti appunto come spunti promozionali. Il target principale di questi sistemi risulta sempre essere il pubblico e gli appassionati.

La sua interfaccia rapida e gradevole, unita alla gratuità ed alla facilità di registrazione e creazione delle exhibition lo rendono un prodotto ottimo anche per la diffusione tramite social e l'integrazione nei siti web: le gallerie possono essere consigliate dai proprietari attraverso i canali di comunicazione e crescere in termini di popolarità sulla piattaforma.

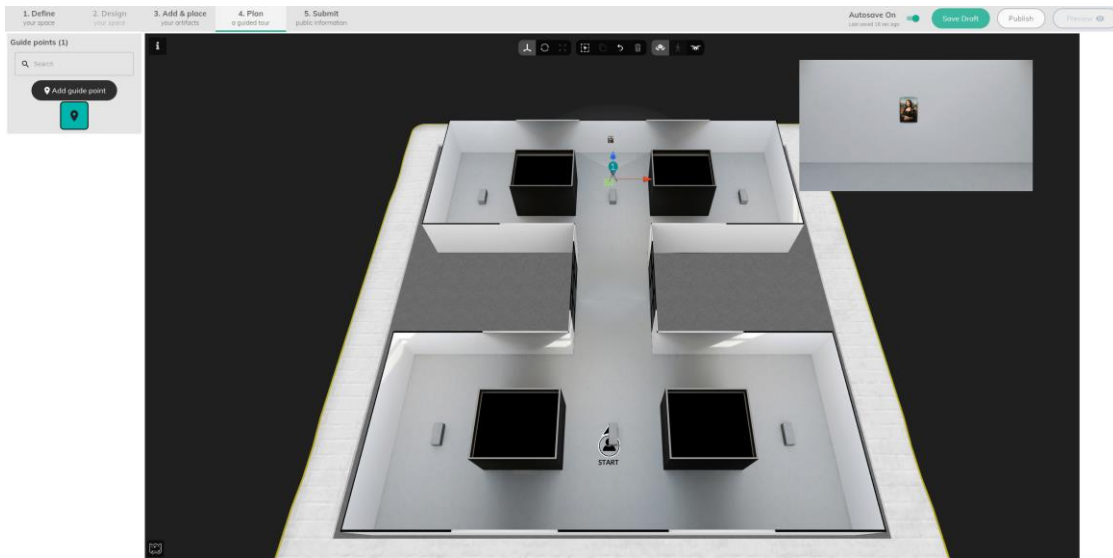


Poster OIM UI 22
424 17 days ago

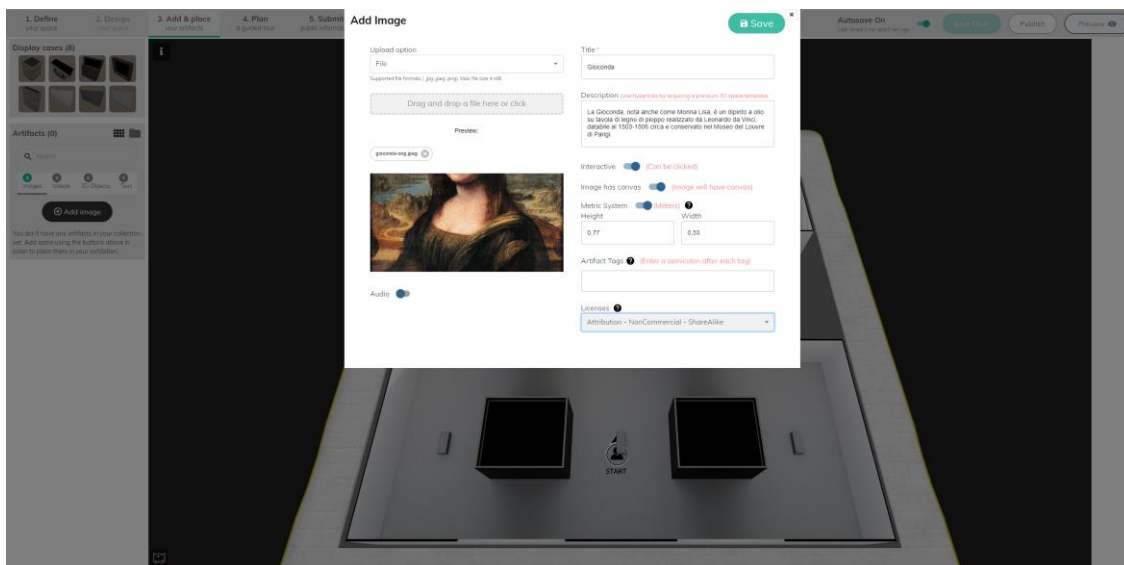
Figura 2.3: Collezione visitabile su ArtSteps



(a)



(c)



(c)

Figura 2.4: Creazione collezione su ArtSteps

2.3 ART.SPACES

Il tool ART.SPACES della piattaforma *KUNSTMATRIX* [8] permette la creazione di mostre di opere d'arte, condivisibile e visitabili da chiunque. Per la gestione e configurazione delle esibizioni mette a disposizione diverse pagine e strumenti.

È presente un'area di inventario, chiamata *ART.DEPOT* (Figura 2.7), di gestione di una collezione di opere da caricare per poterle poi disporre in una mostra virtuale; per ogni quadro fornisce la possibilità di inserire l'immagine, le informazioni relative ed alcune indicazioni per la visualizzazione 3D (ad esempio lo spessore ed il colore della cornice). L'interfaccia di modifica del set di opere è molto statica ed utilizza dei classici form e componenti HTML.

Nella pagina *ART.SPACES* è possibile creare una *virtual exhibition* (Figura 2.8), definendo una mostra virtuale, con tanto di definizione del periodo di apertura della stessa, oltre alla descrizione e all'inserimento di un logo. Configurata l'esposizione si passa alla scelta di uno tra i preset di ambienti virtuali (Figura 2.9), costituiti da mura spoglie da popolare con le opere. È possibile soltanto cambiare il colore delle pareti e caricare un audio di sottofondo all'esplorazione.

Selezionando un'architettura che può essere "combinata", invece di una stanza di tipo *standalone*, è possibile indicare il numero di stanze disponibili ma non modificare l'architettura.

Dalla pagina di gestione della galleria è possibile ottenere il link per l'integrazione in siti web, è infatti uno degli utilizzi più comuni della piattaforma quello di creare un tour virtuale di una mostra da includere nel sito della stessa.

Nell'area *ART.OFFICE* (Figura 2.10) l'utente può gestire i propri contatti (amicizie) e creare un'attività, ovvero organizzare la comunicazione con gli utenti collegati sotto una veste formale, inviando email o proponendo un incontro.

La pagina *ART.AUGMENTED* (Figura 2.11) riporta il link per il download dell'app di realtà aumentata disponibile su Android e iOS, il cui funzionamento prevede di inquadrare un QR code da stampare e disporre su una parete per la visualizzazione delle opere in inventario.

Le esibizioni, in ogni caso, presentano uno stile grafico analogo tra loro (fatto evidente in fase di setup della galleria) puntando a creare un'estetica riconoscibile della piattaforma. La navigazione integra l'utilizzo di mouse e tastiera: oltre che con i tasti freccia, tramite il mouse è possibile muovere la camera solo in orizzontale ma anche spostarsi attraverso le superfici rendendo la navigazione e lo spostamento della camera complesso e confusionario. Spesso l'avatar virtuale inizia a muoversi da solo e il puntamento di un quadro spesso entra in conflitto con la navigazione stessa.

A livello di simulazione virtuale, non essendo stato utilizzato un motore grafico avanzato, le opere risultano appoggiate sulle mura come texture 2D, con la visualizzazione delle informazioni del quadro in un pannello bidimensionale e la possibilità (se impostata) di avviare un'attività per la compravendita dell'opera (se in vendita).

Le funzioni di ART.SPACES (in particolar modo quella di ufficio per la gestione dei contatti) rendono la piattaforma unica, distaccandosi dalla sola natura contemplativa ed esplorativa della concorrenza senza però rinunciarvi, proponendosi (in termini di marketing, interfaccia e pubblicità online) come app destinata ad un pubblico mainstream.

La piattaforma è in effetti una delle poche realtà che cerca di integrare la compravendita nella propria esperienza, anche se spesso non sfruttata dalle esibizioni presenti online in quanto poco realistica per come impostata. Infatti, le attività di contatto con il proprietario non si differenziano troppo dall'invio di un'email da parte di un acquirente o

di un venditore; inoltre, la natura grafica molto spoglia, oltre che priva di un'illuminazione curata, rende l'ambiente assolutamente non realistico. I quadri per come visualizzati danno l'impressione di essere immagini bidimensionali osservabili come si fa leggendo un elenco delle opere.

In questi casi una buona simulazione virtuale è essenziale per comunicare empaticamente con gli utenti, considerando l'importante questione delle consuetudini del mercato dell'arte: infatti l'acquisto vero e proprio di opere passa ancora da canali tradizionali, spesso con la necessità che l'acquirente veda l'opera con i suoi occhi una volta espresso il proprio interesse, o quantomeno contatti i galleristi (talvolta gli stessi autori).

ART.SPACES include tutte le features attinenti al mondo delle simulazioni virtuali di gallerie d'arte cercando sia di costruire una community di appassionati (come per ArtSteps l'utilizzo principale della piattaforma è quello di ricostruzione di mostre famose e condivisione delle stesse), sia di dialogare con un target interessato ad un utilizzo più commerciale. Il risultato è povero in termini di simulazione virtuale ma buono per la creazione di un indice di opere più dinamico, non separando mai la gestione della collezione da quella dell'ambiente virtuale: occorre infatti includere le opere dall'inventario all'esibizione in cui apparire.

La potenza di Meta Gallery in confronto a questo tipo di applicazioni è l'utilizzo di un game engine potente sia in termini di creazione degli algoritmi di allocazione dei quadri (automatici e dipendenti da quanto presente sul database remoto) che di simulazione grafica utilizzando le librerie WebGL, distaccandosi dalla cultura del metaverso tipica di Internet secondo cui creare quanti più spazi possibili, spesso spogli e simili alla riproposizione 3D di qualcosa di bidimensionale (come un indice di quadri).

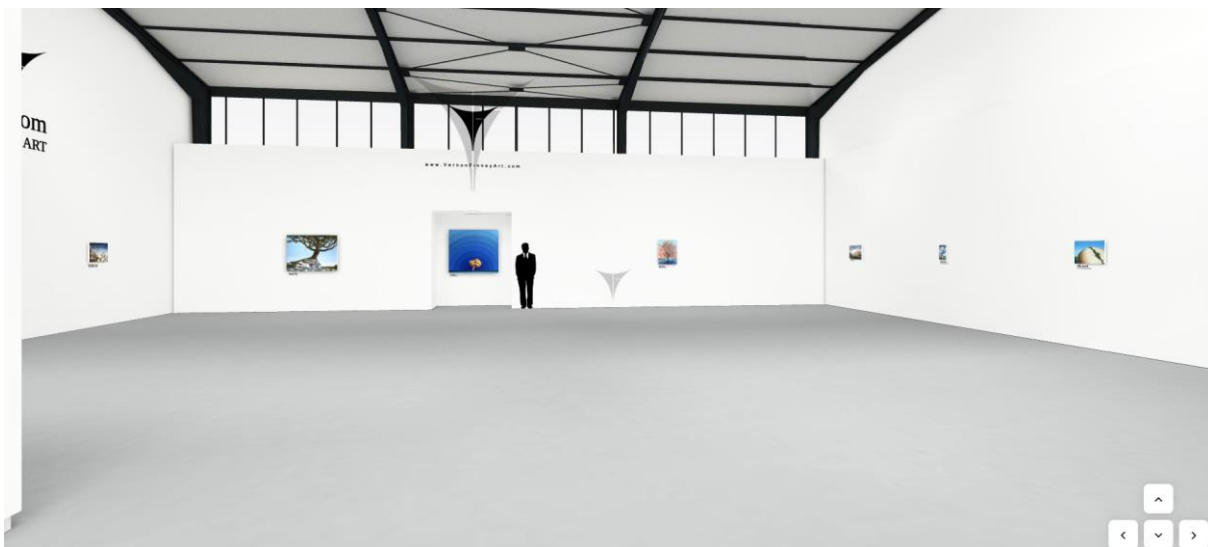


Figura 2.5: Navigazione in ART.SPACES



Figura 2.6: Visualizzazione di un quadro in ART.SPACES

ARTWORK*

JPEG (391.2KB) gioconda-orig

BASIC INFO | ADDITIONAL INFO | 3D-DISPLAY

TITLE*
Gioconda

YEAR
1503

ARTIST
Leonardo Da Vinci

HEIGHT	WIDTH	DEPTH	UNIT
77	53	13	cm

TECHNIQUE
[Empty field]

GENRE
Painting
e.g.: photography, painting

Figura 2.7: ART.DEPOT: Caricamento di un quadro su ART.SPACES

EXHIBITION TITLE * Cancel Save

DURATION
 from:
 to:

DESCRIPTION

IMAGE UPLOAD

*Files must be less than 5 MB.
 Allowed file types: png jpg jpeg.
 Images must be at least 480x480 pixels. Images larger than 5000x5000 pixels will be resized.*

Figura 2.8: Creazione di un'exhibition in ART.SPACES

STANDALONE ROOMS:

 154 45 POSITIONS (+2 ON THE FLOOR) 200M² BASE AREA 6M WALL HEIGHT	 151 28 POSITIONS (+2 ON THE FLOOR) 100M² BASE AREA 6M WALL HEIGHT	 29 40 POSITIONS (+3 ON THE FLOOR) 320M² BASE AREA 3,4/7M WALL HEIGHT	 26 24 POSITIONS (+3 ON THE FLOOR) 120M² BASE AREA 4M WALL HEIGHT
--	--	---	---

ROOMS THAT CAN BE COMBINED:

 19 37 POSITIONS 240M² BASE AREA 3/6M WALL HEIGHT	 17 53 POSITIONS (+3 ON THE FLOOR) 400M² BASE AREA 3,2/6M WALL HEIGHT	 16 36 POSITIONS (+3 ON THE FLOOR) 160M² BASE AREA 3,2 WALL HEIGHT	 13 28 POSITIONS 160M² BASE AREA 3,2 WALL HEIGHT
---	---	--	--

WALL COLOR *

- WHITE
- LIGHT GREY
- GREY
- GREEN
- DARK YELLOW
- RED
- ORANGE
- DARK BLUE
- ANTHRACITE

AMBIENT AUDIO

UPLOAD

Please be aware, that you absolutely NEED permission to use the audio file. Put any license and attribution info into the exhibition description, to avoid problems with copyright holders. If you use Creative Commons licenses, please take care to check if the license covers your use case (commercial or non-commercial) and that you use the correct attribution. Please see here for more info: https://en.wikipedia.org/wiki/Creative_Commons_license

Figura 2.9: Scelta dell'ambientazione virtuale in ART.SPACES

BASICINFOS * ATTACHMENTS Cancel Save

TYPE OF ACTIVITY * - Select a value -

DATE - Select a value -

ARTWORK e-mail
invoice
letter
meeting
note
phone call

please start typing the artwork title

CONTACT Add another item

NOTES

Figura 2.10: Creazione di un'attività per i contatti in ART.SPACES

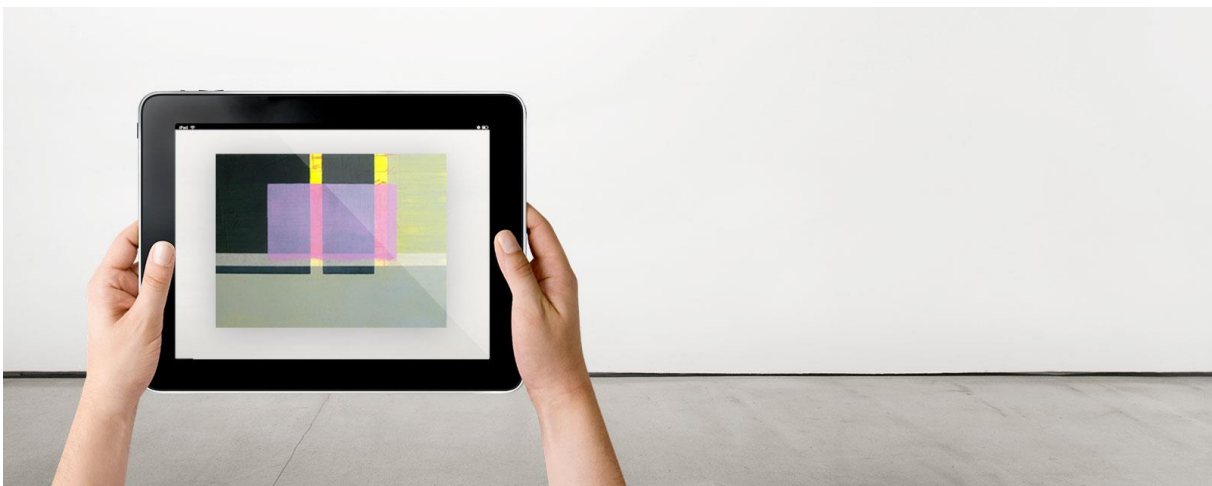


Figura 2.11: Utilizzo dell'app ART.AUGMENTED [9]

2.4 Space

La piattaforma Space [10] consente la navigazione e la creazione di ambienti virtuali 3D, collegandosi fortemente al concetto di metaverso e di realtà virtuale, consente infatti di utilizzare un visore o di accedere tramite smartphone alla simulazione, come mostrato in Figura 2.12. Nella più classica delle applicazioni di questo tipo offre un tipo di esperienza multiutente, in cui, nelle vesti di un avatar di gioco, poter visitare delle aree virtuali e comunicare con gli altri utenti (utilizzo del microfono).

Il punto di accesso per gli spazi virtuali è una *lobby* (Figura 2.13), ovvero una sala d'attesa virtuale, tramite la quale entrare nelle singole zone di interesse, oltrepassando le immagini raffiguranti il luogo di destinazione o cliccando su di esse per essere teletrasportati in quest'ultimo.

Lo stile grafico è abbastanza povero in termini di risoluzione e simulazione ma gradevole alla vista, sono spesso presenti molti colori e la possibilità di integrare media come immagini e video, come mostrato in Figura 2.14. La navigazione è libera e la gestione della camera è efficace.

La creazione di un nuovo spazio avviene specificando per prima cosa il tipo di attività dell'ambiente (Figura 2.15), ad esempio un negozio o un evento. Viene impostato poi se pubblicare o mantenere privato lo spazio, seguito dallo scopo (business o intrattenimento). Vi è quindi la possibilità di scegliere tra una serie di ambienti virtuali disponibili o di richiederne uno *custom*: al costo di 999.99\$ l'app promette di consegnare un nuovo ambiente 3D in 21 giorni. I preset, invece, contengono tutti uno stile grafico grazioso e colorato ma con uno scarso realismo in termini di materiali, illuminazione e modelli.

A questo punto nello spazio creato è possibile creare degli eventi a cui invitare gli utenti, specificando un nome, la data di inizio e fine, un periodo di sconti, i biglietti disponibili e il prezzo. Qui viene evidenziata la natura della piattaforma, il cui intento è quello di rendere disponibili degli spazi virtuali per creare degli eventi virtuali con l'intento di vendere un qualcosa o predisporre un incontro (conferenza, dialogo, ritrovo). A livello di personalizzazione dello spazio, infatti, l'unica opzione disponibile è l'aggiunta di prodotti e media, come mostrato in Figura 2.16, senza l'accesso ad un editor 3D di posizionamento e gestione dell'architettura e degli elementi. I prodotti in vendita e visualizzabili possono essere di tipo mediale ma anche modelli 3D da esporre per la vendita.

Gli ambienti accessibili al momento dalla piattaforma sono privi di pubblico, nonostante il forte impegno a livello di interfaccia nel rendere l'app un luogo di incontro immersivo. L'app si posiziona a metà tra il voler creare degli spazi di incontro e delle stanze di contemplazione di oggetti da comprare o soltanto visualizzare. A livello pratico gli spazi al momento sono vuoti e la piattaforma si occupa più che altro di ospitare eventi privati, oltre che essere un esercizio di stile e di dimostrazione delle potenzialità del metaverso; anche attraverso i canali social di Space è possibile notare una parabola discendente della propria popolarità, occupandosi ad ora per lo più di consulenza e divulgazione attraverso la propria app dei concetti riguardanti metaverso e VR. [11]

Dal punto di vista del design di Meta Gallery è interessante l'analisi dello stile grafico e soprattutto delle modalità di comunicazione tra venditori e clienti, in questo caso tutti utilizzatori della funzionalità principale dell'app, ma in cui i protagonisti sono gli utenti e non le opere e con un dialogo sincrono tra le varie parti, richiedendo la presenza contemporanea di tutti per una corretta esperienza. Si rivolge a un target di mercato di stampo commerciale o professionistico focalizzandosi sul trasportare gli attori nello stesso luogo portandoli faccia a faccia.

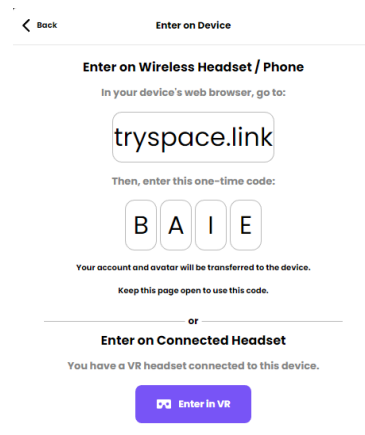
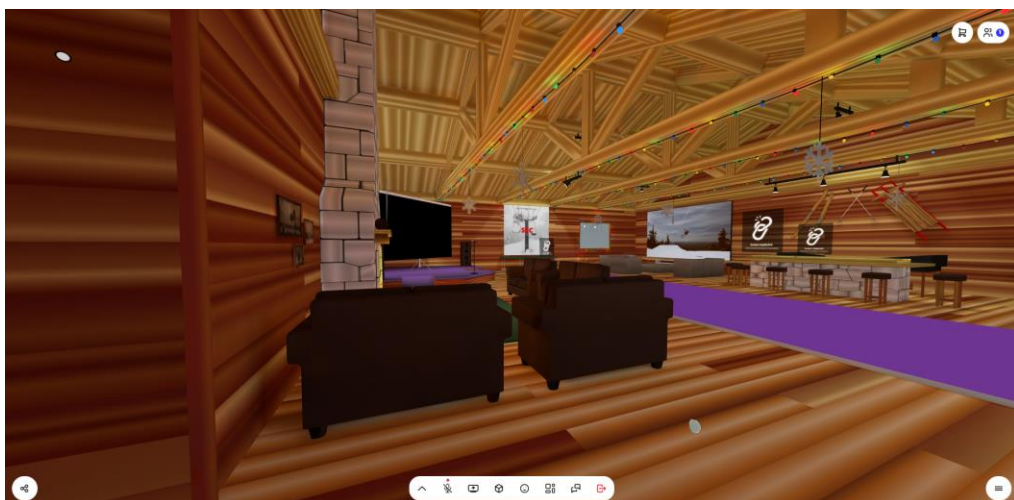


Figura 2.12: Configurazione di un device esterno su Space



Figura 2.13: Lobby di ingresso su Space





(b)

Figura 2.14: Navigazione attraverso un club di sci su Space

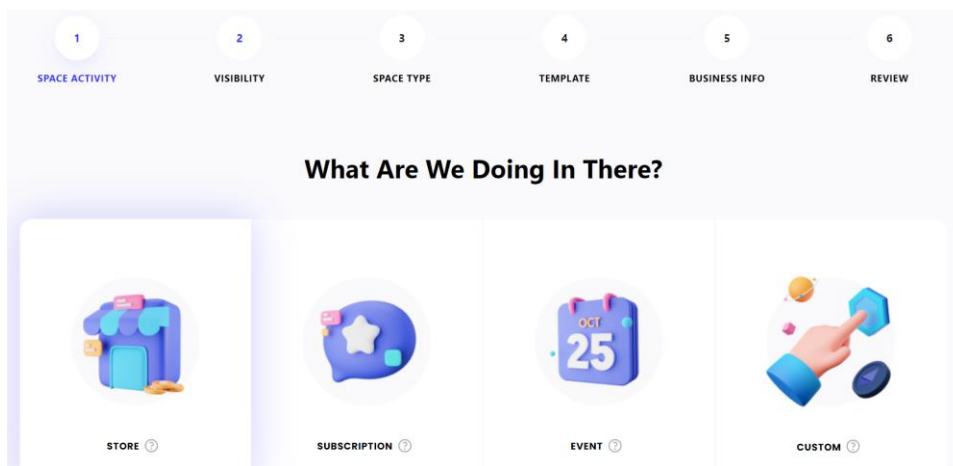


Figura 2.15: Flusso di creazione di un nuovo spazio virtuale su Space

← Add Product



<p>Title</p> <input type="text" value="Product title"/> <p>Description (optional)</p> <input type="text"/> <p>Image (.PNG, .JPG, .GIF) ⓘ</p> <div style="border: 1px dashed gray; padding: 10px; text-align: center;">  <p>Choose file</p> </div> <p>3D Model File ⓘ</p> <p><input checked="" type="radio"/> I have the 3D file ready</p>	<p>Product Type</p> <div style="border: 1px solid gray; padding: 5px; display: flex; justify-content: space-between;"> Product Type ⌵ </div> <p>Product Status</p> <div style="border: 1px solid gray; padding: 5px; display: flex; justify-content: space-between;"> Draft ⌵ </div> <p><input type="radio"/> I need help to create a 3D file</p>
<p>3D Model (.GLB, .glb) ⓘ</p> <div style="border: 1px dashed gray; padding: 10px; text-align: center;">  <p>Choose file</p> </div>	

Figura 2.16: Aggiunta di un prodotto all'ambiente virtuale su Space

2.5 Room

Room è una piattaforma per la creazione di esperienze virtuali [12], il servizio si rivolge ad aziende, negozi online ed enti culturali per lavorare professionalmente su contenuti 3D. Il principale topic affrontato da Room è la creazione di spazi per la vendita ed eventi virtuali o ibridi sfruttando le tecnologie digitali XR adatte ad un determinato obiettivo ospitando tutta l'esperienza nella piattaforma e visualizzando contenuti 2D, 3D (Figura 2.17), AR (Figura 2.18) e VR, come specificato sul proprio sito web. Il servizio è a pagamento ma offre una versione di prova per il testing con alcune funzionalità ridotte, non è possibile infatti caricare i propri modelli 3D e ha delle forti limitazioni in termini di risorse accessibili.

Essendo il focus molto aziendale, Room punta molto sulla presenza di un'efficace assistenza clienti e un'ampia documentazione delle potenzialità del prodotto, ponendosi anche come esperta consulenza riguardo la creazione di spazi virtuali e sfruttamento del metaverso.

La creazione dei modelli 3D da disporre negli ambienti virtuali può avvenire tramite diverse modalità [13]:

- scelta da un portfolio di oggetti personalizzabile in termini di materiali (colori) e dimensioni;
- scansione degli oggetti tramite dei software per smartphone per ricreare i modelli;
- caricamento una foto per sfruttare la *fotogrammetria*⁶, richiede infatti una serie di immagini scattate da differenti posizioni
- caricamento di un video 4k inquadrando ogni angolo di un oggetto (sconsigliata rispetto alle foto)
- caricamento di una singola foto o del link ad un prodotto presente nel proprio negozio virtuale
- caricamento di dati 3D come modelli o file CAD di planimetrie, progetti architettonici

L'editor 3D permette il caricamento all'interno di stanze virtuali di oggetti 3D e media (ad esempio foto o video all'interno di schermi), puntando esplicitamente a ricreare dei luoghi nel metaverso. Lo stile grafico degli ambienti è abbastanza spoglio e non molto ottimizzato; tramite l'utilizzo di modelli 3D è però possibile posizionare ovunque i propri oggetti e ricreare distanze e spazi realistici, come mostrato in Figura 2.19.

L'ulteriore possibilità è l'utilizzo dei modelli tramite realtà aumentata, posizionandoli nello spazio reale mediante lo smartphone con o senza l'utilizzo di marker per il tracciamento delle aree fisiche. Uno degli utilizzi sponsorizzati è la creazione di un libro per bambini con la realtà aumentata per aggiungere movimento e profondità.

Nonostante la presenza del piano free di prova del servizio, la piattaforma punta alla collaborazione a pagamento con aziende; infatti, le potenzialità maggiori prevedono l'intervento dei modellatori di Room per la creazione personalizzata e non automatizzata dei modelli 3D e delle ambientazioni. Tramite il servizio le aziende possono creare ambienti realistici e dedicati ad eventi o alla vendita sfruttando il metaverso attraverso l'utilizzo della grafica migliore possibile e di un design degli spazi adatto alle proprie esigenze.

⁶ Elaborare un modello 3D a partire dalle sue foto

Nonostante la diversità con il progetto Meta Gallery, puntando a creare esperienze fortemente personalizzate e limitate nel tempo (principalmente eventi), è di forte interesse l'utilizzo della simulazione virtuale per la rappresentazione degli oggetti in scale realistiche, oltre ad un'interfaccia del sito semplice ma allo stesso tempo professionale.



Figura 2.17: Utilizzo della visualizzazione di modelli 3D in Room



Figura 2.18: Utilizzo della AR in Room



Figura 2.19: Spazio virtuale ricreato su Room

3 Requisiti

Le richieste e l'analisi del mercato hanno portato alla definizione delle caratteristiche del sistema, ovvero un'applicazione simulativa che permette di navigare ed esplorare una collezione d'arte (la propria oppure una esterna grazie ad un invito), fornendo al contempo una struttura per la gestione della stessa.

L'esperienza utente si dirama quindi in due scenari esteticamente e funzionalmente differenti: un client per la simulazione 3D e un portale web per il controllo *user-friendly* del database delle opere. Seguono i requisiti emersi per le varie funzionalità.

Nel portale web i collezionisti devono aver accesso ad un'area riservata la quale consente loro di dialogare con un database contenente varie informazioni relative alle opere possedute, nello specifico campi testuali (nome, autore, dimensioni...) e un'immagine del quadro, ovvero una scansione o foto dello stesso da visualizzare nell'applicativo come *texture* all'interno di una cornice 3D. Tramite il portale gli utenti devono potersi registrare, effettuare il login, inserire o rimuovere dei quadri e invitare altri utenti a visitare la propria collezione.

L'applicazione simulativa rappresenta il cuore del sistema, nonché il motivo principale della sua creazione. In un ambiente virtuale 3D devono essere posizionati i quadri sulla base delle informazioni presenti sul database in quel momento, ricreando una galleria d'arte realistica. Devono essere quindi mantenute inalterate e proporzionali le dimensioni dei quadri e le loro immagini per una visualizzazione corretta ed attinente alla realtà. La modularità di sviluppo è essenziale per la creazione di un sistema basato su ambienti virtuali mutevoli e definibili: l'assegnazione e disposizione dei quadri deve essere indipendente dall'ambientazione 3D nella quale ci si trova, seguendo sempre delle regole e vincoli che garantiscano il mantenimento del realismo visivo ma anche il rispetto della volontà dei collezionisti.

Il client deve fornire quindi, a fronte di un login, l'esplorazione in prima persona (simulazione della navigazione reale attraverso una galleria d'arte) di un luogo virtuale contenente dei quadri. Attraverso il movimento e lo spostamento dello sguardo deve essere possibile visualizzare le opere ed accedere ad un'interfaccia contenente le altre informazioni archiviate proprio come i pannelli descrittivi accanto ai quadri in contesti reali.

Dovendo ricreare dei mondi virtuali simulativi con relazioni e comportamenti da definire, lo strumento da utilizzare per lo sviluppo è sicuramente quello del game engine, un software grazie al quale poter definire i rapporti tra gli oggetti 3D posizionati e le operazioni da eseguire dati determinati input.

Il terzo elemento è proprio il database contenente i dati delle opere, i file immagine, la lista degli account e le relazioni tra loro, ovvero la definizione di chi è abilitato a visitare un determinato utente. Il ruolo di questo componente è di permettere l'interrogazione e la modifica del suo contenuto agli script.

A livello di esperienza utente è necessario garantire la facilità di utilizzo dell'applicazione, destinata anche ad attori esterni non interessati alle dinamiche del portale web ma solo a visitare una determinata collezione. I requisiti del sistema si sposano bene con una natura web, evitando di costringere gli utenti all'installazione di un software e permettendo così una più comoda la modifica del contenuto del *DB* (database) da qualsiasi località.

Quest'ultimo deve necessariamente essere depositato su un server accessibile da remoto per permettere al client di scaricare ad ogni esecuzione la lista dei quadri da posizionare in scena.

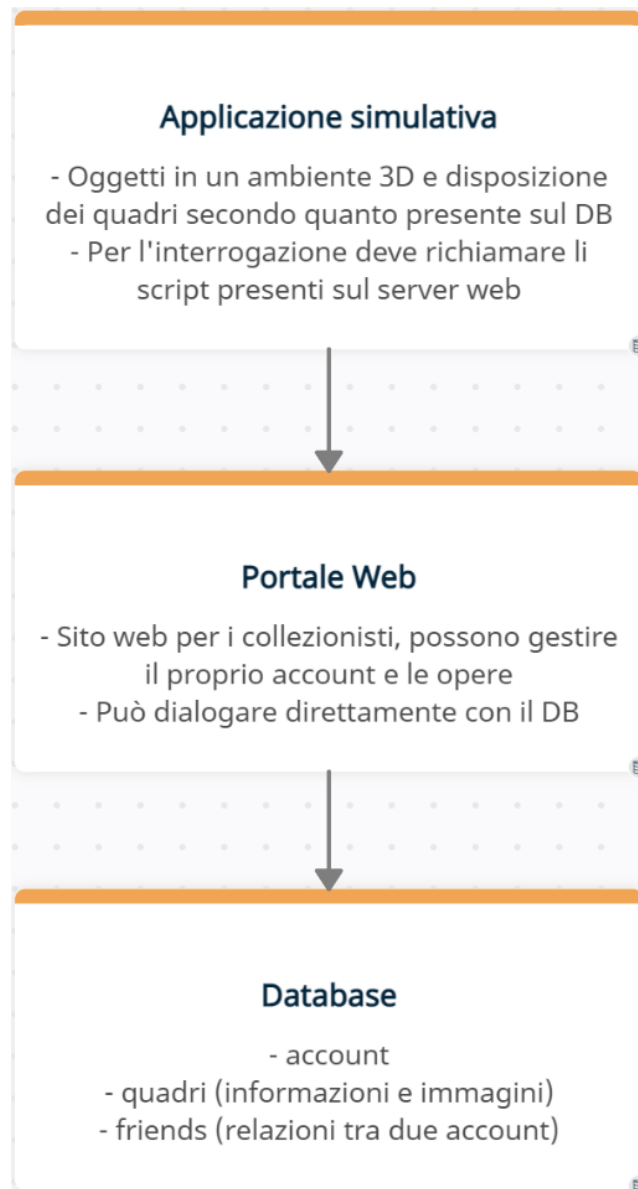


Diagramma 1: Architettura del sistema Meta Gallery

Dovendo creare il progetto da zero è stata necessaria una lunga fase di preproduzione composta da numerosi test per prendere diverse scelte, riguardanti principalmente la gestione del database, quale game engine utilizzare e le tecniche di navigazione e visualizzazione quadri.

Le considerazioni e i test riguardanti database e game engine, qui riportate uno di seguito all'altro, si sono verificate in realtà in parallelo studiando la migliore coppia di soluzione.

Se per la scelta del DB si sono percorse diverse strade tra i database relazionali, l'universo di *Amazon AWS* e *Firebase* di Google, la questione riguardante il game engine si è protratta tra *Unity* e *Unreal Engine*, confrontando le loro versioni di editor e la possibilità di costruire applicativi compilati per le varie piattaforme (Android, Windows..) o un'unica web basata sulla libreria grafica WebGL.

3.1 Processo decisionale riguardante il DB e il server web

Dalle prime fasi di progettazioni è risultata chiara la necessità di un sistema di gestione dei dati online, per permettere l'utilizzo multiutente del sistema ma soprattutto l'archiviazione dei quadri con le loro immagini e informazioni correlate.

L'indicazione iniziale data dall'azienda era di muoversi attraverso una struttura dati di prova su Amazon AWS creata insieme al prototipo dimostrativo su 3DVista: un semplice record di un paio di utenti (padri) possessori di file immagini (figli). Una lunga fase di test basata sulle strutture di Amazon AWS ha guidato le iniziali scelte specialmente riguardanti quale game engine utilizzare.

La libertà decisionale permessa dal design da zero del progetto ha permesso di tornare indietro sulle impressioni iniziali. L'archiviazione organizzata su Amazon S3, servizio che consente la collezione di grandi quantità di dati strutturati in gerarchie anche complesse facilitandone l'accesso in sicurezza [14], è risultato inutilmente complicato per una struttura come quella necessaria al progetto in questione: fin da subito è risultato chiaro che il DB sarebbe stato costituito da poche tabelle senza necessità di un aggiornamento serrato dei contenuti (utilizzati quasi sempre in lettura).

Anche le prestazioni necessarie al sistema non sarebbero dovute essere eccessive: l'interrogazione principale è la richiesta di una lista di quadri da posizionare nell'ambiente 3d, ovvero un numero limitato e piccolo di oggetti se utilizzata una programmazione attenta e modulare.

Per l'utilizzo previsto, un database di qualunque tipologia sarebbe stato sufficiente senza particolari differenze prestazionali tra uno e l'altro, è così che la scelta più ovvia è risultata essere un DB relazionale [15] interrogabile con il linguaggio *SQL*. I database relazionali hanno una forte documentazione a cui appoggiarsi, sono di semplice comprensione abbattendo i tempi di sviluppo e possono garantire la consistenza dei dati tra query simultanee e aggiornamento dati, definendo le relazioni tra gli attori.

Tra i servizi AWS per la configurazione di database SQL il più adatto a compiti di questo tipo è Amazon RDS, un'opzione che continua a risultare valida per una gestione futura più performante e industriale su ampia scala con grandi quantitativi di dati. Risulta comunque complesso muoversi all'interno del servizio soprattutto cercando di non usare le componenti a pagamento essendo ancora in una fase di test con diverse alternative gratuite.

Ciò che è risultato dall'utilizzo di AWS è la scoperta di una rete in cloud complessa e diramata, non comoda per dei test o prototipi ma molto adatta a invogliare le aziende o gli utenti a costruire la propria presenza interamente in quel sistema con i vari servizi in comunicazione tra loro a garantire performance e sicurezza di alto livello.

Da qui risulta come la configurazione degli utenti, dei punti di accesso, delle regole di sicurezza e soprattutto la convivenza con un servizio di web hosting avrebbe richiesto ore di studio, organizzazione e costi non necessari al progetto. Un aspetto importante di AWS è infatti il suo essere orientato all'industria, alla gestione massiva di dati con una tariffazione complessa a consumo legata ai flussi di dati, localizzazione dei server e sfruttamento di feature.

Il web hosting si lega alle questioni relative alla gestione del database in modo molto stretto, in quanto parte del sistema Meta Gallery risulta essere la gestione di una pagina utente per la creazione degli account, invio di inviti e soprattutto aggiornamento della collezione dati. Anche se ancora in un'ottica prototipale e scheletrica a livello visivo, è importantissimo proporre a futuri committenti la pagina web dimostrando la semplicità

potenziale di utilizzo sfruttando un back-end autonomo e solido. La configurazione manuale tramite *RDMS*⁷ visivo o da terminale non avrebbe lo stesso effetto e presenterebbe il sistema in una fase molto più embrionale richiedendo uno sforzo di sviluppo pari a quello dell'applicativo 3D.

Un'attenta fase di test si è quindi basata sul cercare il metodo migliore per costruire un database relazionale, basato su SQL. Inizialmente una soluzione in locale ha permesso di configurare le tabelle e i primi script utilizzando la piattaforma *XAMPP*: distribuzione di Apache che offre un DB *MySQL* configurabile dal client a riga di comando dello stesso RDBMS o tramite l'interfaccia di gestione *phpMyAdmin*. Infatti, *XAMPP* si basa su php per la scrittura di script e permette di creare un sistema web ospitato in locale (localhost).

I test su Amazon S3, RDS e MySQL sono proceduti parallelamente a quelli per la scelta del game engine, infatti la decisione sull'utilizzo di Unity o Unreal Engine avrebbe portato vincoli stringenti sulla tecnologia da usare per il DB e viceversa. Anche la scelta delle librerie grafiche e della versione dell'editor porta a strade molto diverse tra loro per quanto riguarda le richieste HTTP e in generale la comunicazione sulla Rete dell'applicativo.

Le differenze tra game engine sono approfondite nel capitolo di seguito, ma la possibilità ancora aperta in fase di preproduzione di scelta tra Unity e Unreal Engine (soprattutto fino alla scelta di sfruttare le librerie WebGL), ha lasciato aperte le varie porte riguardo a tutti i possibili servizi elencati in precedenza, come Firebase.

Firebase è una piattaforma per la creazione di applicazioni web ben documentata [16] e facilmente integrabile in Unity ma anche su Unreal Engine attraverso plugin esterni. Il servizio offre due soluzioni di database: *Cloud Firestore* e *Database in tempo reale*. I due sono differenziati in base alle esigenze strutturali, di scalabilità e di latenza, oltre che per una diversa tariffazione. Ciò che però caratterizza Firebase è la sua natura mobile; infatti, il suo utilizzo è molto consigliato per lo sviluppo Android ma è risultata presto evidente in fase di test la sua incompatibilità con le librerie WebGL, portando alla sua esclusione dal progetto.

XAMPP è un'ottima soluzione di test permettendo di ospitare un server web in locale ma dalle prime prove ufficiali di script e di interfacciamento con il game engine scelto si è optato per un provider di servizi di web hosting gratuito: *000webhost.com*. La piattaforma permette il caricamento di script e pagine html per la creazione di un sito web visitabile a un indirizzo url pubblico offrendo la possibilità di ospitare anche un database con accesso a *phpMyAdmin* tramite il pannello di controllo.

A seguito dei test elencati in precedenza è stata progettata la presenza web di *Meta Gallery*: pagina web presente all'indirizzo <https://metagallery.000webhostapp.com> e database di motore *InnoDB* in un server *MariaDB* (compatibile con *MySQL* per le *query*⁸ utilizzate negli script).

⁷ Relational database management system

⁸ Interrogazioni del database

3.2 Processo decisionale riguardante il game engine

Il primo scoglio decisionale nel processo di design dell'applicativo ha riguardato la piattaforma di distribuzione e le librerie grafiche.

Meta Gallery prevede un applicativo WebGL, questo per far sì di essere aperto con facilità da molteplici dispositivi di natura diversa ed essere ospitato da un sito Web, evitando processi di installazione. Questa soluzione risulta più efficace rispetto alla pubblicazione di *build*⁹ differenziate per Android, iOS, Windows o macOS tralasciando nella fase prototipale uno dei punti più interessanti tra le ispirazioni e potenzialità per il futuro del sistema: la realtà virtuale. Una versione VR del visore, per dispositivi Oculus, rappresenta dall'inizio del design uno snodo fondamentale per la vita della piattaforma, essendo di sempre maggiore interesse per gli enti museali e anche fortemente legato al mondo dei tour virtuali, esperienze immersive e realistiche nonché a quello del metaverso.

I prodotti WebGL necessitano di numerose accortezze di ottimizzazione grafica e computazionale non potendo essere a conoscenza del dispositivo utilizzato dall'utente, oltre alle limitazioni causate dai Browser e l'importanza di avere caricamenti rapidi e in particolar modo la scelta di questa tecnologia grafica ha posto i più importanti vincoli nel processo decisionale tra game engine e sistema DB/web hosting. Analizzando alcune esperienze virtuali presenti sul web e create con questa tecnologia, risulta complessa a livello computazionale la gestione dell'illuminazione con un numero limitato di riflessi e delle ombre che appaiono seghettate e poco realistiche. In generale si verifica quindi dell'*aliasing* [17] ovvero la rappresentazione delle immagini con un numero di pixel insufficiente per via di un sottocampionamento; un altro aspetto critico è relativo alle dimensioni dei file da scaricare per i browser e dalle limitazioni prestazionali di questi ultimi che causano caricamenti potenzialmente molto lenti e *framerate* non elevati. In ogni caso le necessità grafiche del progetto sono soddisfatte da un utilizzo ottimizzato e attento delle librerie grafiche WebGL.

Le due opzioni di sviluppo analizzate sono state l'utilizzo del game engine Unreal Engine (UE) e Unity.

UE è un game engine di proprietà di *Epic Games* [18], azienda in forte crescita con un impatto enorme nell'industria dei videogames, nonché potenza economica e leader di mercato degli ultimi anni. La sua community è molto attiva permettendone una crescita esponenziale mese dopo mese. Presenta numerose differenze tra versioni, specialmente tra la UE4 e UE5 (escludendo gli ambienti di sviluppo più vecchi) ma anche all'interno delle varie distribuzioni della 4. Il suo motore grafico è il più gradevole, indicato per la creazione di ambienti virtuali 3D accattivanti tramite l'efficace espressione di illuminazione, ombre, animazione e rendering che sfruttano i più recenti algoritmi, tecniche e librerie di dialogo con le schede grafiche.

UE è sicuramente orientato allo sviluppo di videogiochi, con automatismi e servizi che permettono di costruire le interazioni del mondo virtuale in modo rivoluzionario rispetto al passato. Le forti potenzialità di rendering permettono anche un'ottima gestione di *cinematics* e più in generale dell'animazione 3D. Molti aspetti di programmazione vengono semplificati tramite la struttura a nodi detta *Blueprints* nascondendo la complessità delle istruzioni utilizzate e delle funzioni. Infatti, molti plugin di terze parti mettono a disposizione nuovi nodi e la possibilità di ampliare le strutture *Blueprints*. L'utilizzo di componenti esterni è difatti quasi una necessità per l'utilizzo di Unreal Engine ma lo

⁹ Eseguibile compilato per distribuire l'app al di fuori dell'ambiente di sviluppo

diventa de facto durante lo sviluppo di un sistema di natura non strettamente videoludica o di animazione.

L'analisi del mercato di prodotti di terze parti atti a dialogare con un server web ha mostrato la presenza di diversi plugin per le richieste HTTP e l'interrogazione di un database mostrando un ulteriore ostacolo per lo sviluppo di Meta Gallery su UE: i tool sono per la maggior parte a pagamento, specialmente i più completi, rendendo più complessa la fase di testing e confronto tra le varie tecnologie, momento in cui già risultava più plausibile si sarebbe continuato a sviluppare su Unity e quindi poco utile acquistare dei prodotti.

Ad evidenziare la direzione intrapresa ormai da Epic Games di rendere Unreal un ambiente di sviluppo fortemente incentrato sul gaming e non sullo sviluppo di sistemi front/back-end c'è la forte difficoltà a testare e quindi implementare i plugin in base alle versioni, infatti tra ogni versione di UE spesso cambiano i nodi, i vincoli e le variabili richieste dai tool senza che ciò sia documentato nelle rispettive descrizioni presenti su *GitHub* o *Unreal Engine Marketplace*. Ciò risulta ancora più problematico vista la necessità di utilizzo di più *SDK* differenti (Software Development Kit, almeno uno per il dialogo con il database e uno per le HTTP request), con di norma versioni consigliate di Unreal differenti, almeno per ciò che riguarda i prodotti gratuiti utilizzati nei test.

Nei mesi successivi di sviluppo in realtà la scelta di progettare un sistema php/SQL ha portato a richiedere all'applicativo delle librerie per dialogare con il portale web solo tramite HTTP request lasciando agli script del server il compito di interrogare il DB, cosa che non sarebbe stata possibile con Firebase o Amazon AWS.

Gran parte dei software migliori e più documentati permettono l'integrazione di Firebase sia con UE4 che UE5. Come tutti gli *SDK* di Firebase si rivolgono allo sviluppo mobile, con la possibilità di integrarli per desktop tramite più passaggi da progettare in base al plugin scelto.

Riguardo il dialogo con i DB SQL sono pochissimi i kit gratuiti per Unreal e soprattutto rivolti alle prime versioni di UE4, una delle prime caratteristiche a suggerire la creazione di un sistema di back-end tutto interno al web host restringendo il requisito minimo per lo sviluppo dell'applicativo su Unreal alla necessità di dialogo con un server Web tramite delle richieste HTTP.

Per i servizi di Amazon AWS (ad esempio RDS) non è presente un kit ufficiale di Amazon ma diversi plugin di terze parti, in questo caso però il prezzo è molto elevato e sono stati evitati test in assenza di certezza di utilizzo di tale servizio di archiviazione dei dati.

I nodi Blueprints tra prodotti interni all'IDE e terze parti (in questo caso a prezzi molto contenuti e con un discreto numero di possibilità) necessari all'invio di richieste HTTP e dati *Get* e *Post* generalmente permettono di dialogare con il server tramite documenti in formato *JSON* in cui scrivere le richieste e i dati processati, come nell'esempio in Figura 3.1. Il plugin più documentato e utilizzato è *VaRest* [19] pubblicato su *GitHub* dallo sviluppatore di nome ufna.

Come descritto dallo stesso sviluppatore *VaRest* è un plugin per UE4 per la trasmissione di dati su HTTP tramite *REST* (Representational State Transfer), ovvero un insieme di regole e vincoli strutturali per la trasmissione delle Request che consegna l'informazione al server tramite formati come il *JSON* ed ha tra le caratteristiche più importanti l'essere *stateless*: ogni richiesta è distinta e indipendente, invece di utilizzare un approccio a connessioni tra client e server dentro la quale, una volta impostata, poter inviare indeterminati flussi di dati.

VaRest permette la costruzione della comunicazione REST tramite il solo utilizzo di nodi non richiedendo la programmazione C++ (su cui è basato Unreal Engine).

Essendo diventato uno standard per questo tipo di lavori su UE è presente una discreta documentazione e discussione da parte degli utenti sui forum dedicati e ufficiali di Unreal. Anche in questo caso i nodi cambiano nomenclatura, funzioni e variabili richieste in base

alla versione utilizzata con funzionalità che vengono deprecate nel corso degli anni e non sempre opportunamente sostituite.

Risulta comunque questo il metodo migliore per operare delle richieste Post e Get con il server per il processamento di dati, i quali, nel caso di Meta Gallery, definiscono il sistema legato al login dell'account, gli altri utenti visitabili e il download dei quadri con annesse informazioni.

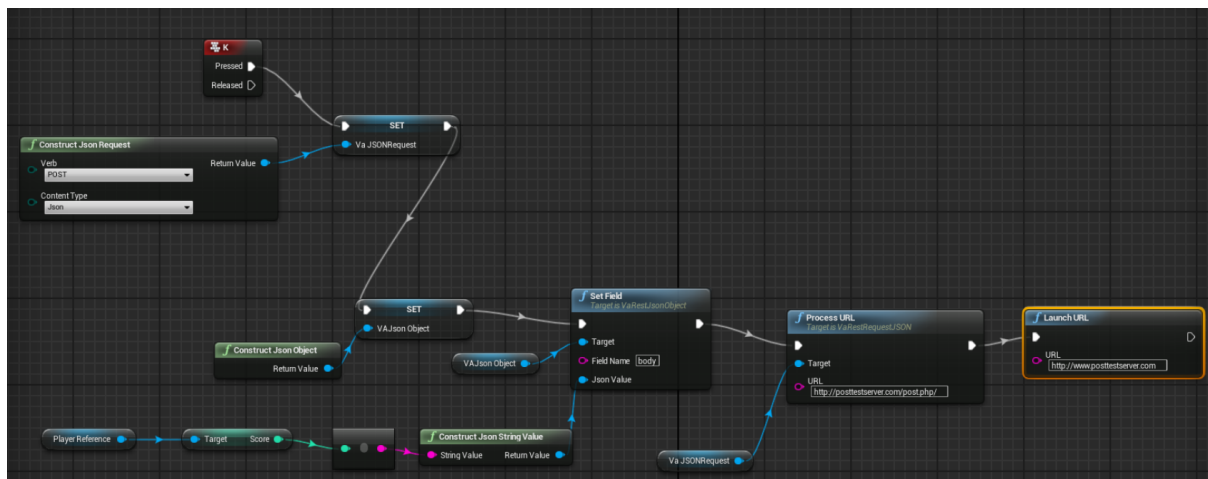


Figura 3.1: Creazione HTTP Request su Unreal Engine

L'aspetto più vincolante come per le altre componenti è risultato l'utilizzo delle librerie grafiche WebGL. Chiaramente Unreal Engine è orientato maggiormente allo sviluppo desktop e mobile ma non è stato totalmente chiaro inizialmente la non possibilità di operare con WebGL con le versioni più recenti. Infatti Epic Games tramite un comunicato ha dichiarato la chiusura dello sviluppo per tale libreria: "As of Unreal Engine 4.24, support for the HTML5 platform has been migrated out of the engine to a public Platform Extension that can be maintained and improved by community members" [20].

Epic Games ha difatti abbandonato il supporto a HTML5 dalla versione 4.24 di Unreal permettendo però l'integrazione delle funzioni ormai deprecate tramite GitHub con una procedura documentata sulla stessa pagina GitHub di UE lasciando lo sviluppo in mano alla propria community.

Ciò che è successo in realtà a partire dalla versione 4.24 è la morte definitiva del supporto WebGL, già portato avanti con poca convinzione delle versioni precedenti e ad oggi lasciato a livello di mercato principalmente a Unity, che risulta incompatibile quasi con tutte le applicazioni di terze parti e funzioni necessarie al progetto, impendendo l'utilizzo di versioni superiori richieste in alcuni casi dai plugin (ad esempio la maggior parte di quelli atti al dialogo con i database supportano ad oggi i motori 4.27 e 5.0).

VaRest è invece disponibile per molte versioni accedendo alle vecchie *Release* tramite GitHub ma utilizzando ad esempio quella per UE4.24 molti nodi risultano deprecati così come presenti nella documentazione rendendo lento e complesso lo sviluppo di Blueprint efficaci per le request necessarie.

Unity è un game engine sviluppato da Unity Technologies di larghissimo utilizzo, per anni incontrastato per lo sviluppo di giochi indie per via della sua facilità di uso e di programmazione (si basa sul linguaggio C#), curva di apprendimento, community forte con tutorial di ogni tipo online e la possibilità di sviluppare senza l'acquisto di una licenza il proprio gioco. Sono talmente tante le opere sviluppate su questo motore, in particolar modo in una fetta di prodotti di fascia di prezzo medio/bassa (senza farsi mancare nomi illustri in ambito gaming) e a partire dal suo rilascio nel 2005, da rendere la sua estetica fortemente riconoscibile, specialmente nelle ambientazioni 3D con i propri pregi e difetti ormai noti al grande pubblico. Se dal lato 3D fa parte dei prodotti divenuti ormai

mainstream (è sempre più facile e diffuso affacciarsi alla creazione di mondi virtuali tramite Blender e lo stesso Unity) è divenuto sempre più popolare per la creazione di applicazioni 2D e di conseguenza per lo sviluppo di giochi mobile. Oltre all'ambito gaming, Unity risulta spesso un'ottima soluzione per piattaforme che integrano un'interfaccia visuale (3D, 2D, VR, AR) e una programmazione atta alla risposta di input esterni. Così Unity viene spesso utilizzato nell'ambito di opere che sfruttano la tecnologia come molti casi di arte contemporanea e performance art ma anche in piattaforme visive simili a Meta Gallery in cui è richiesto un qualche tipo di interazione.

Ad oggi l'estetica di Unity risulta in parte superata, o quantomeno molto conosciuta. Non sfrutta molte delle più nuove tecniche di rendering grafico sebbene ci siano molte differenze in base alla pipeline di rendering utilizzata. Unity permette infatti di configurare il proprio progetto secondo 3 diverse pipeline, ovvero insieme di operazioni, algoritmi, procedure e librerie grafiche, con la possibilità di cambiare successivamente ma con molti accorgimenti: infatti una scelta piuttosto che un'altra impatta su numerosi aspetti, per la maggior parte legati alla configurazione dei materiali e all'illuminazione. L'opzione più vecchia e per molti anni di standard è la *Built-in Render Pipeline* che al contrario delle altre due è definita come non *scriptable*, ovvero la possibilità di controllare tramite gli script le sue feature.

L'opzione più diffusa in quanto permette una buona gestione a livello grafico mantenendo l'app leggera a livello computazionale (spesso un requisito molto importante per chi si rivolge a Unity per lo sviluppo del proprio sistema) è la *Universal Render Pipeline* (URP) mentre vengono demandate alla *High Definition Render Pipeline* (HDRP) le funzionalità più avanzate e complesse, soprattutto per ciò che riguarda l'illuminazione e la simulazione realistica delle riflessioni (uno degli aspetti più pesanti per le schede video).

Se già per una questione di performance computazionale si era deciso di operare con URP, sfruttandola per raggiungere il miglior risultato visivo, il fatto che HDRP non fosse compatibile con WebGL ha sancito che lo sviluppo di Meta Gallery avvenisse sulla Universal Render Pipeline.

Come su Unreal Engine è stata portata avanti una fase di testing su Unity per verificarne l'efficacia relativamente alle richieste del progetto partendo dalla possibilità di trasmettere dati online.

Partendo dalla struttura iniziale di prova su Amazon AWS, in particolar modo il primo test sul servizio S3 e successivamente su RDS, è stato sfruttato un SDK per Unity fornito dallo stesso AWS che include in diversi Package (insieme di file e componenti per Unity) numerosi script e scene di prova per i servizi compatibili permettendo il dialogo e lo sfruttamento delle funzioni di autenticazione, query e analytics del servizio.

Le difficoltà relative al servizio AWS poco adatto a Meta Gallery si sono palesate anche sul game engine. Gli script dell'SDK sono eccessivamente complessi per le necessità del progetto e necessitano di una preparazione dei dati online diversa da come si stava lavorando, con la creazione di nodi e punti di accesso tramite numerose regole e passaggi. Anche Firebase fornisce un kit gratuito per Unity e una documentazione multilingua semplice da seguire che permette di collegare i due servizi per sviluppare un'app. Se ciò inizialmente sembrava essere una strada indipendente dalla piattaforma di distribuzione, è stato poi evidente con le prime build di prova l'incompatibilità dell'SDK con WebGL: i numerosi script, scene e contesti di prova sono progettati per funzionare nello sviluppo mobile (specialmente per contesti classici come punteggi e classifiche nei giochi).

Per quanto riguarda l'SQL sono presenti sul mercato numerosi asset per la connessione a un DB e molti tutorial per evitare l'utilizzo di terze parti, che però risultano la soluzione più solida per accedere ai dati, con l'asset *SQLite* [21] come opzione più popolare e documentata ma a pagamento seppur a un prezzo accessibile. Sono comunque necessari numerosi passaggi per dialogare con il DB o semplicemente più di quanti ne servano nel caso di script php su un server web e di quanto sembrerebbe necessario a un progetto

come Meta Gallery. Infatti Unity si fonda sul C# e l'integrazione con degli script php, Javascript o simili necessita di una buona dose di accortezze e una certa manualità con il linguaggio, lavorando più a basso livello di quanto serve nella maggior parte dei casi a livello di programmazione Unity.

L'hosting web (in locale o su un server remoto) risulta ideale per ospitare tutto la struttura web del sistema per la creazione e login degli utenti, le loro interazioni e il caricamento, la modifica e la visualizzazione dei dati mantenendo interno ad esso il dialogo con il database strutturando una gerarchia di pagine web HTML con integrazione di script PHP. Unity, infatti, tramite le sue librerie interne permette una comoda gestione delle richieste HTTP potendo operare in Get e Post su un determinato URL. Il flusso di dati dal game engine al server può quindi avvenire tramite HTTP request (comunicate con metodo Post nel progetto) e gli script PHP possono prendere in carico le informazioni ricevute, interrogare facilmente il database presente sullo stesso server e stampare delle risposte. I dati in output possono essere letti dallo script C# che dopo la request rimane in attesa della risposta memorizzando il risultato. Queste funzionalità sono permesse su Unity dalla libreria *UnityWebRequest* [22] che si occupa di gestire i flussi di dati per la comunicazione HTTP la cui documentazione è presente sul sito ufficiale di Unity in quanto *API* interna e perfettamente compatibile con WebGL.

A livello grafico la URP risulta adeguata se sfruttata con attenzione cercando di ottimizzare le prestazioni tramite il *bake* dell'illuminazione e la riduzione della risoluzione dei vari algoritmi di simulazione grafica. Inoltre, studiando le possibilità offerte da Unreal Engine per WebGL, analizzando le esperienze virtuali più famose presenti online non ne deriva una differenza grafica che giustifichi lo sviluppo su UE: le limitazioni imposte dallo sviluppo per HTML5 causano dei vincoli da cui non poter uscire in termini di luci, ombre, interazioni e materiali.

Da queste analisi deriva la scelta strutturale definitiva per il design del sistema: sviluppo di un applicativo Unity WebGL e dialogo con un server web che ospiti le pagine per la gestione degli account e gli script per l'interrogazione di un database compatibile SQL ospitato dallo stesso web host.

3.3 Navigazione e visualizzazione quadri

Il sistema prevede da richiesta la navigazione in prima persona in un ambiente reale, simulando quindi la presenza umana su un terreno. Una prima fase di valutazione ha riguardato la modalità di spostamento all'interno dell'ambiente virtuale. L'opzione più classica è costituita dal movimento libero controllato tradizionalmente dai comandi di input (tastiera o controller esterno) potendo andare ovunque nei limiti delle superfici percorribili e degli ostacoli come muri e oggetti. L'alternativa è lo spostamento secondo *spot* selezionando dei punti evidenziati e muovendosi automaticamente verso di essi. Quest'ultima è l'opzione scelta per il progetto, nell'ambiente virtuale l'esplorazione libera non è necessaria non avendo molto altro con cui interagire oltre le opere, ciò permette una scalabilità maggiore tra dispositivi differenti e quindi diverse modalità di input.

Ciò che risulta necessario in assenza di un'esplorazione libera è una posizione privilegiata per l'osservazione delle opere esposte, la possibilità di ritrovarsi di fronte ai quadri e poterne leggere le informazioni allegate. Da qui la divisione in spot visibili sul terreno navigando tra uno e l'altro per eseguire gli spostamenti tra zone della mappa e spot posizionati di fronte alle opere accessibili cliccando nell'area di intorno al quadro.

4 Portale web per la gestione delle collezioni

Nel seguente capitolo viene descritta la struttura presente online per la gestione delle funzionalità di amministrazione delle pagine personali degli utenti e contenenti gli script di dialogo con il DB e l'applicativo. Questa parte del sistema è quindi divisa nel portale web pubblico e negli script di dialogo con il database.

Il primo è accessibile in rete e ne vengono descritte le pagine visualizzabili graficamente, ovvero l'interfaccia superiore agli script e che permette di indicare i dati da trasmettere agli stessi o fornisce i risultati, di conseguenza, a quanto eseguito dai codici.

La descrizione dei programmi presenti sul server segue la descrizione della composizione del database, necessaria alla comprensione delle query di interrogazione dei dati.

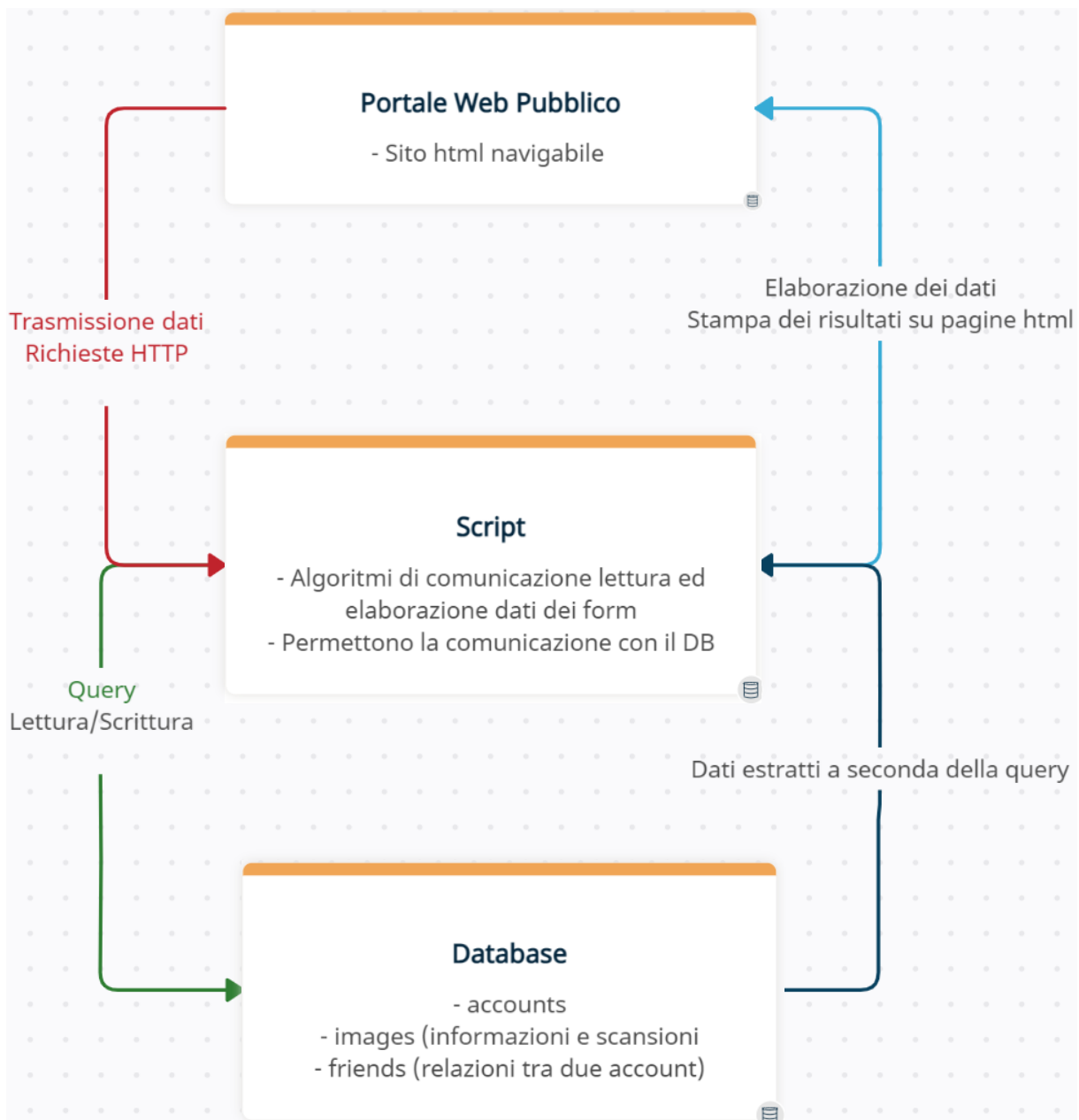


Diagramma 2: Funzionamento architettura web

4.1 Struttura e logica del database

Il database è situato su un server web offerto dall'host gestito dall'RDBMS MariaDB, la cui struttura si basa sull'algebra relazionale e quindi in una logica di tabelle: attributi (campi) distribuiti in *tuple* (righe). Le tabelle sono scritte in SQL *InnoDB*, ovvero il motore per il salvataggio dei dati che mette a disposizione le librerie per la gestione fisica dei dati per le funzionalità di lettura/scrittura, cache, indicizzazione e per definire i vincoli e le regole.

Il DB, di nome "*id19124944_metagallerymysql*", è visualizzabile e gestibile graficamente tramite il tool phpMyAdmin (Figura 4.1) e accessibile tramite username e password configurati in fase di creazione. Il server da indicare in fase di connessione negli script è "localhost" in quanto interno allo stesso portale del sito web.

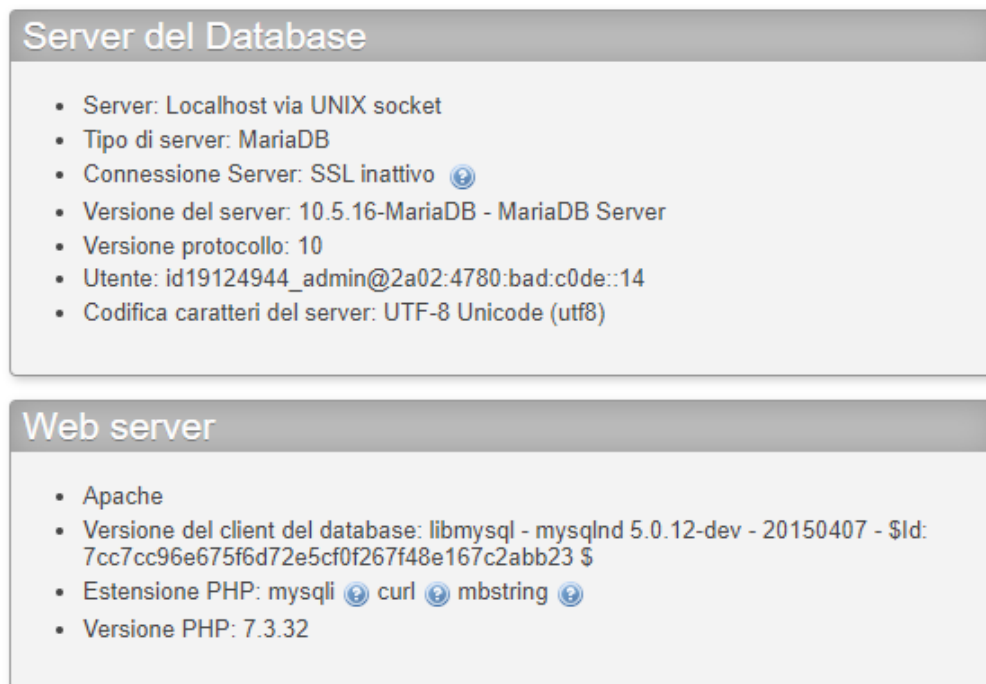


Figura 4.1: Server del Database dall'interfaccia di phpMyAdmin

La struttura del DB è molto semplice con la presenza di tre tabelle, mostrate nel Diagramma 3: *accounts*, *friends*, *images*. Ognuna di loro ha un campo definito come *chiave primaria* per identificare univocamente ogni tupla e nei casi di associazione permettono di collegare due tabelle identificando le chiavi di una come chiavi esterne dell'altra.

accounts ha un identificativo numerico incrementale come *primary key* e i campi di email e password, è presente un ulteriore campo *token* per la gestione futura delle chiavi di memorizzazione per i cookie (al momento non utilizzato).

La tabella delle immagini ha un identificativo incrementale (*imagesID*) seguito dalla chiave esterna riferita alle righe di *accounts* per associare ogni quadro a un utente. Sono presenti poi i vari account popolati con le informazioni provenienti dal form di upload. Oltre ai classici campi numerici e testuali, l'attributo *frame* è un *BIT* per contenere 0 o 1, mentre *image* è un BLOB, ovvero una variabile binaria contenente una grande quantità di bit in sequenza da interpretare successivamente come immagine o altri tipi di file. Nello

specifico il campo è un *MEDIUMBLOB* che si differenzia dallo standard per la possibilità di memorizzare un dato fino a 16MB invece che 64KB.

Infine, le amicizie sono costituite da una tabella contenente solo chiavi esterne, in cui ogni riga è l'unione dei dati di due account: il secondo in ordine da sinistra è l'utente invitato dal primo e quindi la visita può essere effettuata in questo senso, il primo account per visitare il secondo deve essere invitato a sua volta.

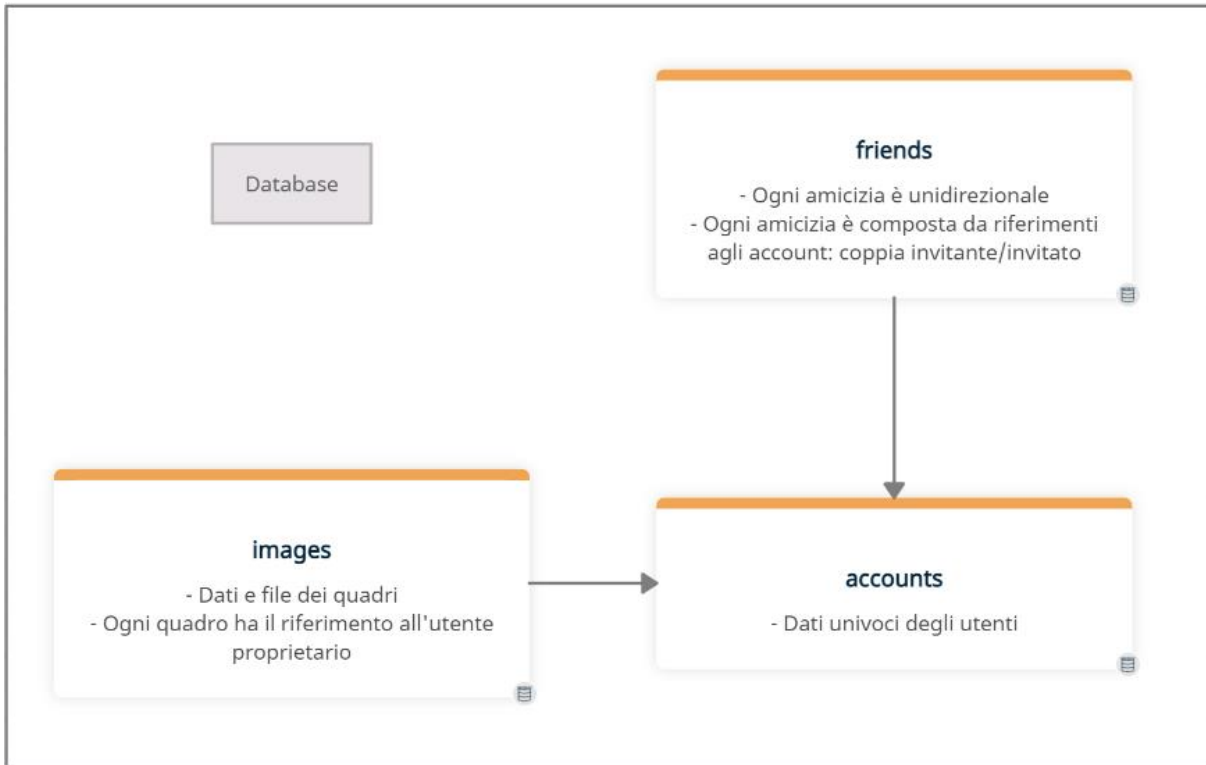


Diagramma 3: Tabelle database

id	email	password	token
2	test@gmail.com	test	.
5	vittorio@gmail.com	test	.
9	2test2@gmail.com	test2	.
10	mazzoleni@gmail.com	mazzoleni	.
11	octobergallery@gmail.com	octobergallery	.
12	t293@gmail.com	t293	.

Figura 4.2: Tabella accounts

id1	email1	id2	email2
2	test@gmail.com	5	vittorio@gmail.com
10	mazzoleni@gmail.com	11	octobergallery@gmail.com
10	mazzoleni@gmail.com	12	t293@gmail.com
11	octobergallery@gmail.com	10	mazzoleni@gmail.com
11	octobergallery@gmail.com	12	t293@gmail.com
12	t293@gmail.com	11	octobergallery@gmail.com
12	t293@gmail.com	10	mazzoleni@gmail.com

Figura 4.3: Tabella friends

imageID	id	image	name	author	width	height	year	frame	description
26	11	[BLOB - 113.8 KiB]	Light Bike	Benji Reid	1.65	1.1	2021	0	- His anti-gravitational mysteries does what fine ...
25	11	[BLOB - 320.3 KiB]	Through the Window of my Room	Brion Gysin	0.13	0.2	1962	0	An extraordinary collection of mystical, and comme...
24	12	[BLOB - 1.3 MiB]	A Lil Shard	Naohiro Utagawa	1.5	1.2	2020	1	The primary source material for Utagawa are photog...
23	12	[BLOB - 821.3 KiB]	Not yet titled (Drying Rack)	Henrik Olai Kaarstein	1.12	1.45	2021	1	The objects emerge modified in unforeseen ways. Th...
22	12	[BLOB - 1.1 MiB]	The Night	Si On	1.8	1.55	2020	1	The mental and physical engagement that the artist...
21	12	[BLOB - 2.0 MiB]	Nice View 01	Naohiro Utagawa	1.2	1.5	2020	1	Here photographs are not just created through the ...
20	12	[BLOB - 353.6 KiB]	Flower Market #1	Lorenzo Vitturi	1.34	1.82	2020	0	56% wool, 44% bamboo silk, Venetian fused glass.
19	12	[BLOB - 1.1 MiB]	This is America	Anna Park	2.24	1.52	2020	1	Anna Park's frenetic charcoal drawings range from ...
18	11	[BLOB - 198.7 KiB]	Carib	Aubrey Williams	0.51	0.38	1959	0	By applying gold leaf to the nails he creates brea...

Figura 4.4: Tabella images

4.2 Features del portale web

Tramite il *File Manager* del portale di 000webhost è possibile operare sui file che configurano la pagina web di Meta Gallery e modificarne il contenuto. All'interno della directory presente al percorso `/public_html` che risulta essere la cartella radice dentro cui caricare la pagina relativa all'homepage (`index.php`) sono archiviati i vari file `.html` e `.php` strutturati nel seguente modo: paralleli alla homepage vi sono le pagine web pubbliche, le quali rappresentano il front-end del sito, con la struttura rappresentata nel **Diagramma 4**; nella sotto-directory `scripts` invece sono archiviati i file di puro scripting back-end per le interazioni tra le varie pagine pubbliche, il contenuto del DB e la gestione delle HTTP-request da parte dell'applicativo Unity, come mostrato nel **Diagramma 5**.

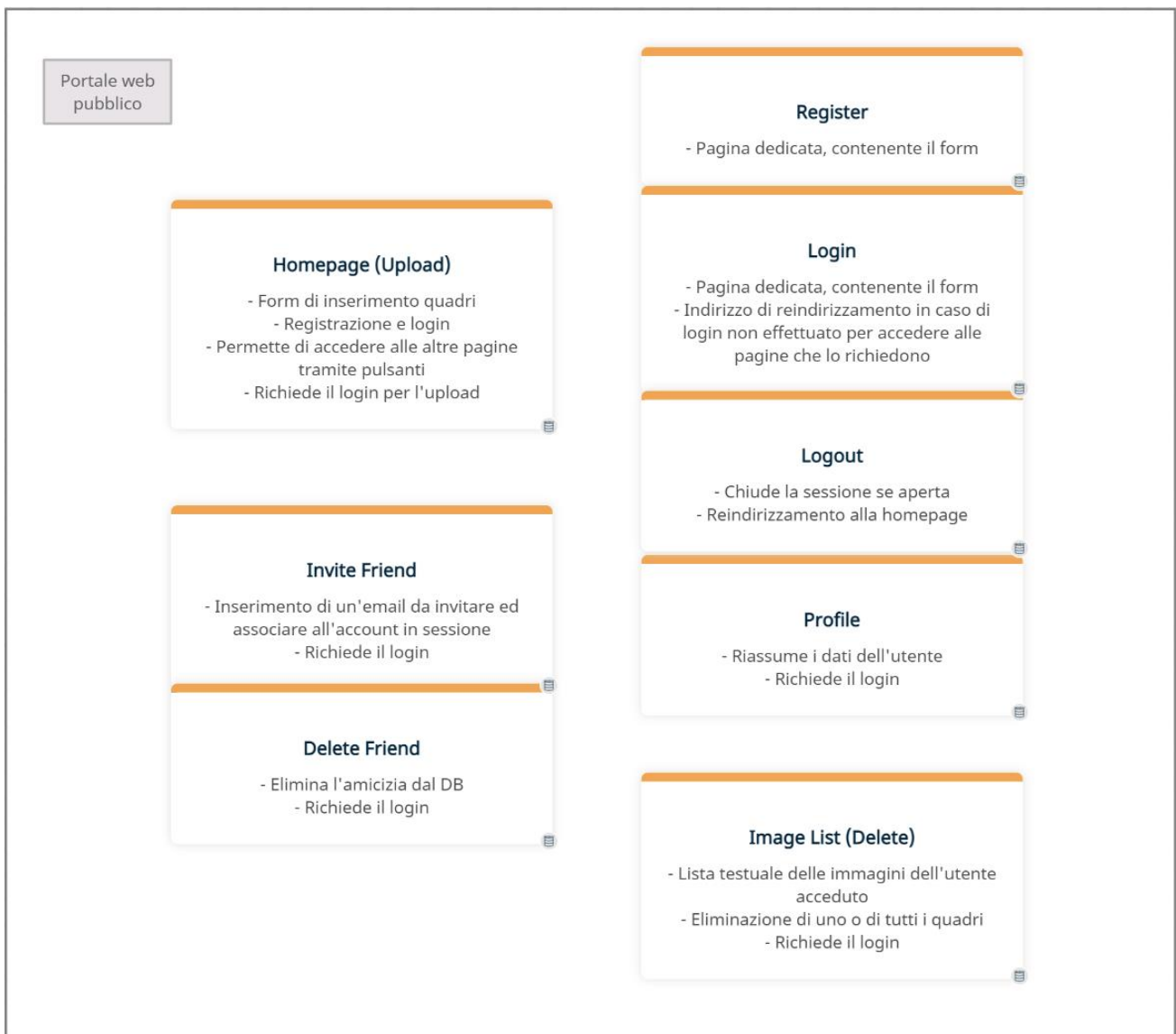


Diagramma 4: Pagine pubbliche del portale web

Numerosi script php permettono la gestione delle interazioni client-server: raccogliere i dati trasmessi in POST dai form (comprese le richieste provenienti dall'applicativo), processarli, interrogare il database, effettuare dei controlli e fornire degli output, ovvero ciò che viene poi visualizzato dal browser o letto dall'applicativo.

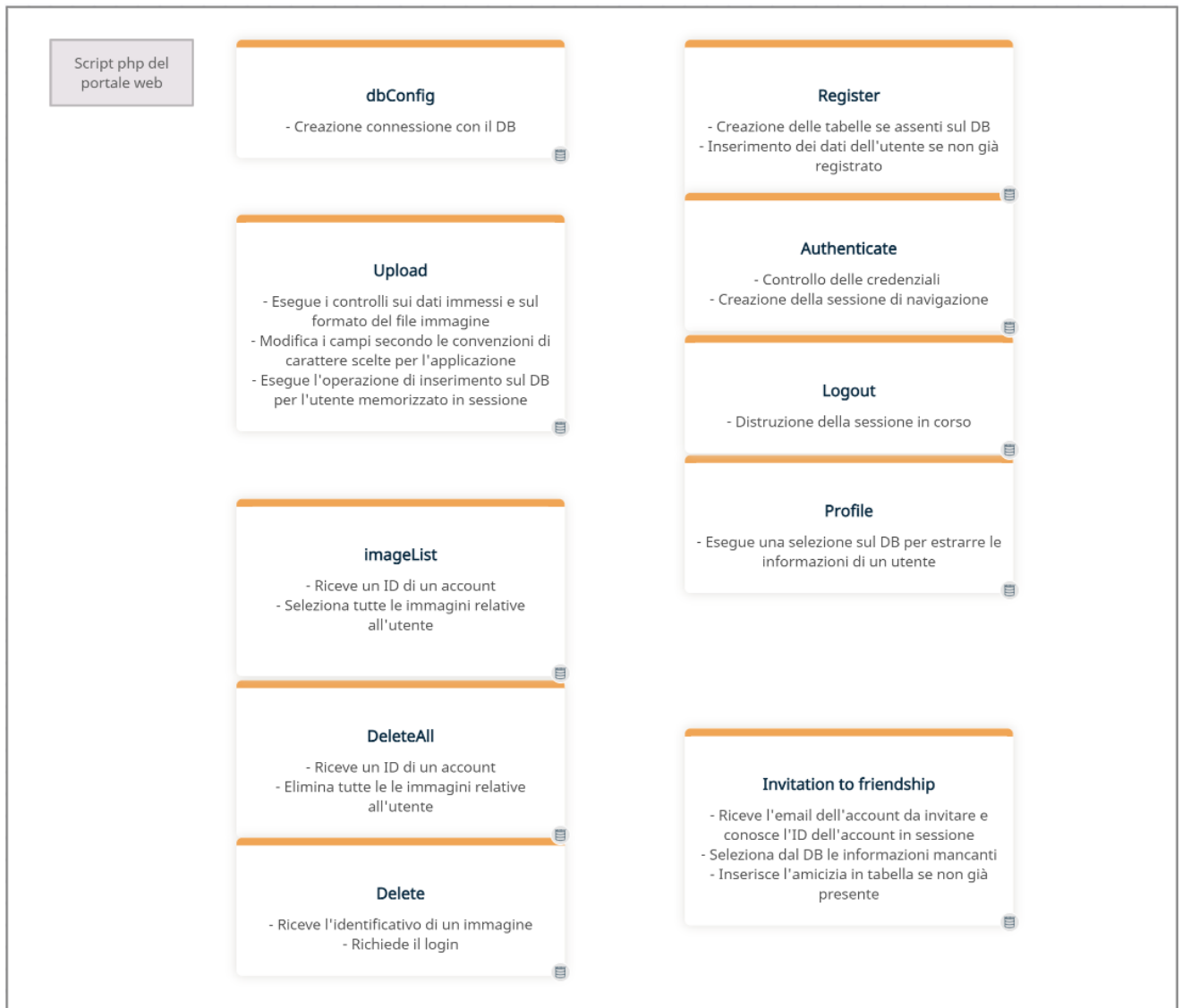


Diagramma 5: Componenti di back-end del portale web

Il servizio utilizzato per la creazione della pagina web di controllo dell'area riservata agli utenti, gestione del database e verosimilmente destinato ad ospitare la versione pubblicata dell'applicativo WebGL è denominato *000webhost* [23]. Il servizio offre numerose funzionalità già dalla sua versione gratuita risultando comodo per testare il proprio sistema, creare dei prototipi, valutando poi la migrazione verso altri servizi o l'upgrade ad una versione più performante a pagamento. Le caratteristiche della modalità standard prevedono delle limitazioni in termini di spazio di archiviazione, scelta del dominio, larghezza di banda e molte altre feature [24].

La pagina web pubblica è presente al seguente URL:

<https://metagallery.000webhostapp.com/> mentre la gestione del medesimo è accessibile dal portale fornito da 000webhost tramite cui configurare tutte le funzionalità, in particolar modo caricare e scrivere i file (File Manager) e utilizzare phpMyAdmin come tool di amministrazione del proprio DB (Gestore Database).

Le funzionalità della pagina web pubblica sono nel dettaglio la registrazione di nuovi account, il login e logout, il caricamento e rimozione dei quadri e la possibilità di invitare o rimuovere altri utenti. Rimangono alcune funzioni di debug nell'ottica di prodotto dimostrativo incentrato sull'applicativo mentre un'interfaccia funzionale ed esteticamente

apprezzabile è demandata ad un futuro sviluppo in base alle necessità (comprendere la distribuzione del potere tra gli utenti o un amministratore centrale). All'attuale livello di sviluppo del sistema sono presenti tutte le funzioni richieste in fase di design ma l'assenza di una vera e propria cura e disposizione grafica degli elementi. In ogni caso la scrittura del codice html si articola su un numero ristretto di tag e componenti permettendo una più attenta e curata definizione dello stile tramite CSS in futuro, oltre all'assenza di elementi tecnici non ancora definiti come dettagli grafici, logo, font, contratto di utilizzo in fase di registrazione e definizione di altre pagine come le tradizionali *Contatti*, *Chi Siamo...*

4.2.1 Configurazione della connessione al database

Lo script dedicato a stabilire una connessione con il db è `dbConfig.php`, esso è essenziale per tutte le operazioni che necessitano di manipolare i dati presenti nelle tabelle e deve essere chiamato come prima cosa dai programmi che intendono utilizzarlo. Dentro viene effettuata la configurazione e apertura della connessione al database in modo da poter essere incluso in maniera modulare all'inizio degli script che lo necessitano e garantire la centralità in un solo codice di tali funzioni.

Prima di impostare il DB viene abilitato il parametro `Access-Control-Allow-Origin` nell'header, ovvero una specifica richiesta da *CORS* (Cross-Origin Resource Sharing) che costituisce una funzione dei browser legata all'accesso delle risorse da parte di un dominio per dei controlli di sicurezza, specialmente da parte di sorgenti esterne (secondo la *Same-Origin Policy* abilitata di default), come può essere il client Unity. La politica CORS non impostata per permettere l'accesso bloccherebbe le richieste POST o GET da parte del client.

Successivamente vengono impostate le variabili per l'accesso al DB richieste dalla funzione di apertura della connessione. A questo punto se la connessione viene effettuata (necessari sempre i controlli del caso per gestire gli errori e stampare un feedback) si può comunicare con il server fino alla chiusura della stessa. La chiusura avviene tramite richiesta formale o semplicemente con la chiusura dello script, ad esempio navigando verso un'altra pagina.

Viene dichiarata inoltre la funzione `openSession(bool $check)` per la gestione della sessione php. Le sessioni consentono infatti di mantenere dei dati durante la navigazione web, chiusura e riapertura delle pagine, reindirizzamenti. È difatti necessaria vista la presenza di un login permettendo la conservazione dei dati dell'utente durante l'esperienza per comunicare i dati di account durante le interrogazioni al database che li richiedono.

Accedere alle variabili della sessione se questa non è stata aperta dallo script può portare ad errori o conflitti: vi è quindi nella funzione (da richiamare in caso di necessità dei vari script) la chiamata all'apertura della *session* se questa non è già in corso e, in base al valore (*true/false*) del *flag*¹⁰ passato come parametro, obbliga la navigazione a posizionarsi sulla pagina di login in caso di accesso non effettuato. Gli script che necessitano di aver proceduto con il login per continuare chiamano `openSession(true)` per assicurare l'accesso dell'utente.

In alcuni codici invece è necessario aprire la sessione ma senza il reindirizzamento al login, chiamando quindi la funzione passando come parametro *false*.

¹⁰ Variabile per la memorizzazione di un'informazione booleana (vero/falso)

4.2.2 Homepage

Molte delle feature sono direttamente accessibili dall'homepage del sito, come mostrato in Figura 4.5, ovvero nella pagina `index.php`, corrispondente all'indirizzo pubblico <https://metagallery.000webhostapp.com/> mentre per visualizzare gli altri file occorre far seguire al suddetto indirizzo il loro nome. L'estensione del file (`.php`) indica il possibile contenimento di uno script oltre al codice html, ad ora non più presente in `index.php` ma demandato ad altri componenti se non per una piccola funzione di *listener* di un evento di click per chiudere il form di login che si apre in modalità *pop-up*.

Seppure, come descritto in precedenza, l'organizzazione grafica ed estetica non è sviluppata in questo prodotto prototipale vi è una minima distribuzione degli elementi del sito.

Nella sezione detta *head* di ogni documento html oltre al titolo della scheda visualizzabile sul browser sono descritti gli stili, in modo da mantenere la consistenza degli oggetti tra le varie pagine. I tipici attributi specificati riguardano colori, dimensioni, margini, posizioni relative e assolute. Ciò permette uniformità tra i vari elementi dei form, caselle di input e bottoni.

Nel *body* vengono descritti gli elementi visualizzabili rappresentando il vero e proprio corpo della pagina a partire da una barra di intestazione contenente il nome del sito, un tasto per accedere ad una pagina contenenti i dettagli del profilo ed uno per effettuare il logout. Questi tre elementi sono collegamenti ipertestuali, ovvero testo che una volta cliccato permette di navigare tra le pagine (tag `<a>`).

Sono presenti dei form, ovvero dei moduli da compilare che una volta inviati reindirizzano la navigazione ad altre pagine trasmettendo loro dei dati tipicamente in modalità Post o Get. Tramite Get i dati vengono esplicitati nell'URL potendo essere visibili agli utenti ed è più indicato per le informazioni di configurazione del sito web; tramite Post invece i dati non sono visibili ma comunicati in un array consultabile dal server e più adatto alle informazioni degli utenti, viene utilizzato infatti quest'ultimo dai form presenti sulla pagina e dati viaggiano nascosti dal client al server.

Vi è un form per il caricamento di un quadro, uno per la registrazione di un utente e uno per il login, visualizzabile in modalità pop-up cliccando sul tasto apposito.

I vari elementi del form costituiscono gli input, i dati da trasmettere allo script specificato nell'attributo di action.

Gli input possono essere di diversa natura: testo, numero, email, password... Un metodo classico di terminare il form è inserire un tasto di invio che esegue la chiamata allo script detto *submit*.

In homepage è inoltre presente una serie di tasti di reindirizzamento alle altre pagine del portale costituiti da dei tag `<button>` rendendo la pagina il punto di accesso principale a tutta la struttura del portale web permettendo di accedere ad ogni funzione e posizionando in primo piano quella principale, ovvero il caricamento di un nuovo quadro.

META GALLERY Profile Logout

Select Image File:

Choose File No file chosen

Name:

Author:

Width (m):

Height (m):

Year:

Frame already in the image:

(a)

Upload

POST DATA

IMAGES LIST

INVITE FRIEND

DELETE FRIEND

LOGIN

(b)

Register

Please fill in this form to create an account.

Email

test@gmail.com

Password

....

Repeat Password

Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

Register

(c)

Figura 4.5: Homepage

4.2.3 Registrazione di un nuovo utente

La registrazione avviene compilando un form presente in homepage specificando la propria email e password con il tradizionale doppio inserimento di quest'ultima e la visualizzazione ad asterischi (Figura 4.6). I tre campi sono obbligatori (attributo *required* nel tag di input html) e il loro tipo permette un primo controllo della correttezza delle informazioni: l'input di tipo password nasconde i caratteri inseriti mentre quello di email richiede una stringa contenente il carattere '@'.

Con un *placeholder* nella casella di testo si indica l'azione da compiere e i vari input occupano ognuno una riga del form presentando prima di ogni elemento il nome del campo da compilare inserito nel tag `<label>` che associa tale scritta all'input corrispondente aiutando i browser a mantenere il focus sui componenti e visualizzarli in modo corretto.

Il form trasmette i dati allo script specificato nell'attributo *action*, in questo caso `/scripts/createAccount.php` dove avviene il controllo di correttezza delle informazioni ed in caso di inconsistenza viene ripresentato lo stesso form in modalità isolata, quindi non interno a `index.php` ma a `register.html` che contiene gli stessi elementi relativi alla registrazione. In tal caso l'URL visualizzato sarà comunque `/scripts/createAccount.php` in quanto `/register.html` viene integrato in tale file attraverso una funzione di richiamo del codice e non tramite un reindirizzamento.

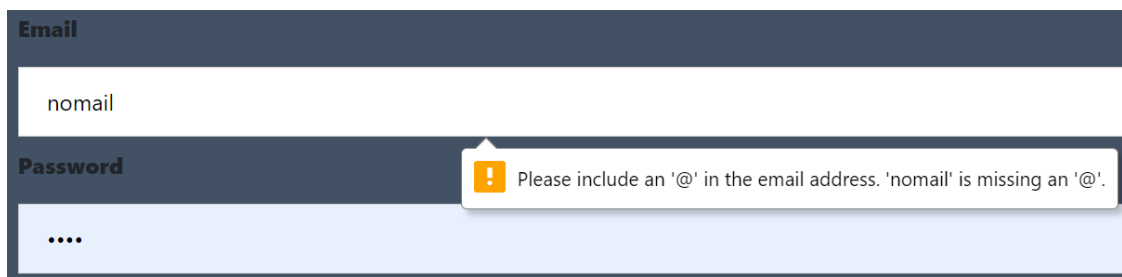


Figure 4.6(a) shows a close-up of the registration form. The 'Email' field contains the text 'nomail'. Below it, the 'Password' field is visible with masked characters '....'. A white error message box with a yellow exclamation mark icon is overlaid on the form, stating: 'Please include an '@' in the email address. 'nomail' is missing an '@'.'

(a)

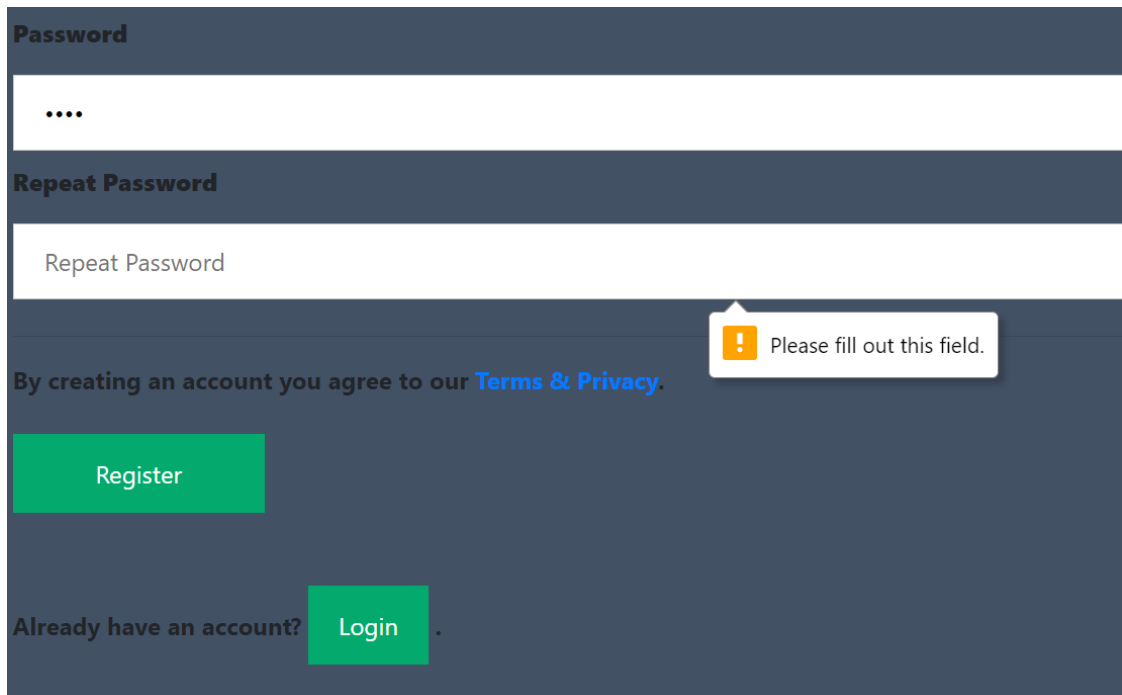


Figure 4.6(b) shows the full registration form. The 'Password' field is masked with '....'. Below it is the 'Repeat Password' field with the placeholder text 'Repeat Password'. A white error message box with a yellow exclamation mark icon is overlaid on the form, stating: 'Please fill out this field.' Below the form, there is a green 'Register' button. At the bottom, there is a link 'Already have an account?' followed by a green 'Login' button.

(b)

Figura 4.6: Form di registrazione

Dal form di registrazione i dati inseriti vengono comunicati allo script `createAccount.php` il quale esegue diverse funzioni. Stabilisce la connessione al DB tramite `dbConfig.php` ed apre la sessione senza controllo sul login essendo questa la fase di creazione dell'account.

La lettura dei dati trasmessi in POST avviene tramite l'array associativo `$_POST` il cui accesso ai campi è permesso dall'utilizzo del nome dell'input relativo inserito nel codice del form come indice del vettore.

Prima di tutto è presente un controllo sulle stringhe (`strcmp()`) di password e conferma per proseguire nella registrazione. In caso di incoerenza viene stampato un feedback di errore nell'inserimento, riproposto all'utente il form di registrazione e quindi chiuso lo script (operazione `exit`).

Prima dell'inserimento delle informazioni relative all'utente il programma controlla la presenza delle tre tabelle sul database e in caso di loro assenza le crea, ciò tramite la direttiva SQL `CREATE TABLE IF NOT EXISTS`, viene quindi specificato il nome della tabella e l'elenco dei campi, questi ultimi sono seguiti da attributi che ne descrivono il tipo ed eventuali vincoli e regole.

Gli identificativi sono tutti impostati come `auto_increment` specificando la natura auto incrementale della chiave primaria (numero intero con la possibilità di decidere gli step numerici di incremento). Vengono poi specificati gli attributi testuali (`varchar`), numerici (`int` o `float`) e la loro dimensione massima. Come descritto nella struttura del database sono inoltre presenti gli attributi `BIT` e `MEDIUMBLOB`.

La chiave esterna viene invece specificata tramite un `constraint` ovvero la descrizione di un vincolo: si indica il valore di chiave e la sua *reference* e tramite la specifica `ON DELETE CASCADE` viene regolata l'eliminazione automatica delle righe della tabella figlia in caso di eliminazione della riga nella tabella genitrice puntata dalla chiave esterna. Dove necessario vengono inoltre indicati dei valori di default e il *charset*, ovvero la tabella di riferimento per i caratteri, in questo caso *Unicode*.

La query in linguaggio SQL viene eseguita tramite la funzione `php query()` fornita nella variabile contenente le procedure di connessione al database (`$db` in `dbConfig.php`). Per ogni interrogazione al database occorre controllare l'avvenuta esecuzione (la funzione restituisce `TRUE`) e gestire gli eventuali errori restituiti (`db->error`).

Oltre la `CREATE TABLE` occorre popolare il DB con i nuovi dati da registrare. Questo avviene tramite la `INSERT`: istruzione SQL che permette di specificare i campi di una tabella e i valori da inserire al loro interno.

Per evitare l'inserimento di un account già esistente viene fatta prima un'ulteriore interrogazione per controllare la presenza dell'email inserita nel form all'interno della tabella `accounts`. Tale query utilizza l'istruzione `SELECT` condizionata ad estrarre solo le righe di interesse come specificato nella clausola `WHERE` dell'istruzione.

È possibile specificare gli attributi da selezionare ma in questo come in molti casi viene effettuata la lettura di tutte le colonne (indicando ciò con un asterisco), viene poi specificata la tabella e la condizione (non obbligatoria per la `SELECT`) che permette di estrarre gli attributi desiderati (in questo caso cercare nella tabella la presenza dell'email inserita).

Se il numero di righe contenute nel risultato della query è zero vuol dire che non è stata trovata nessuna tupla con tali valori e quindi si può procedere all'inserimento dei dati, in caso contrario viene comunicato all'utente.

La connessione al DB viene chiusa formalmente tramite `$db->close()` e tramite l'espressione `require_once` vengono integrati altri codici, utilizzata quindi per accedere

alle risorse di connessione offerte da `dbConfig.php` e aggiungere in coda il contenuto dell'homepage dopo le richieste di registrazione.

4.2.4 Login, Profile, Logout

Sotto al form di registrazione è presente il bottone per accedere al modulo di login, a comparsa in sovrapposizione. Come per la registrazione, questo componente è accessibile *standalone*, ma in questo caso tramite reindirizzamento all'URL relativo `/login.php` ogni qual volta si tenti di effettuare delle operazioni che necessitano di essere acceduto: leggere le informazioni del profilo, caricare un'immagine, rimuoverla, ottenere la lista dei quadri e le operazioni sulle amicizie.

Per effettuare il login sono necessarie email e password, con la possibilità di spuntare "Remember me" per mantenere l'accesso alle riaperture successive, come mostrato in **Figura 4.7**. Gli input sono dello stesso tipo di quelli presenti nel form di registrazione con l'aggiunta di una checkbox per la spunta "Remember me", ovvero una variabile di tipo vero/falso, mentre la funzione di password dimenticata non è al momento implementata.

Figura 4.7: Form di login

L'action del login è lo script presente al percorso `/scripts/authenticate.php`, questo effettua il controllo di correttezza dei dati confrontandoli con quelli presenti sul DB stampando a video una risposta positiva o negativa seguita dalla pagina di `index.php` nel caso di accesso completato o del form di login in caso di email o password non corretta.

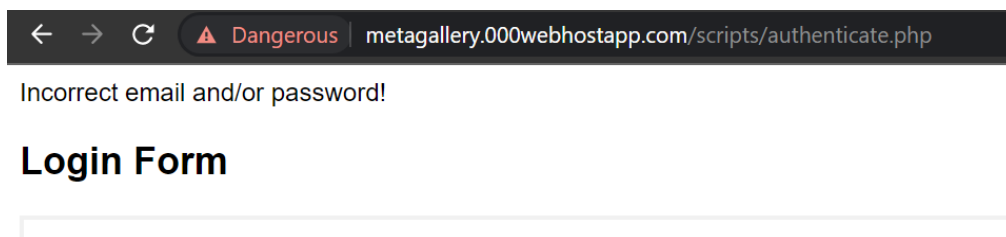


Figura 4.8: Feedback in testa alla pagina

Come gli altri elementi presenti nella navbar di intestazione della pagina, *Profile* è un collegamento ipertestuale che permette di navigare verso la pagina `profile.php`, la quale dopo il login si occupa di restituire le informazioni relative all'account presente sul DB (**Figura 4.9**), costituendo al momento più una funzione di debug la quale può essere facilmente, tramite degli aggiustamenti grafici e funzionali, trasformarsi in una classica

area riservata alla gestione del proprio account.

Da qui è possibile visualizzare l'indirizzo email con il quale ci si è registrati (Email e Username coincidono, a differenza di come operato nella prima parte dello sviluppo), la password in chiaro (chiaramente per scopi di debug ma da nascondere) e l'ID univoco assegnato ad ogni utente.

Nonostante l'email sia per sua natura un identificativo si è preferito lavorare con un dato numerico e non di tipo stringa per la maggiore facilità del suo utilizzo e per avere una chiave nascosta all'utente che si presta meglio all'utilizzo delle sessioni di login, ad una futura implementazione dei cookie e alla possibilità di essere modificata senza intaccare i dati dell'account (garantendo migliori operazioni di sicurezza).

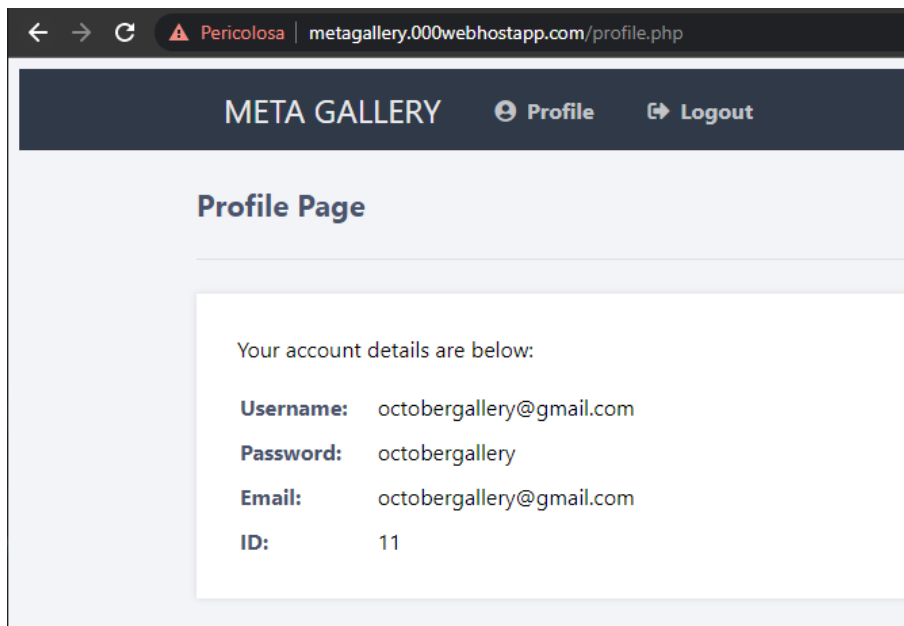


Figura 4.9: Pagina Profile

Il logout è costituito da un altro collegamento ipertestuale presente in homepage, cliccando su di esso viene chiusa la sessione in corso tramite delle funzioni descritte nei paragrafi di analisi degli script per poi essere reindirizzati nuovamente in homepage.

Il form di accesso ha come action lo script `authenticate.php` con il compito di controllare i dati immessi e la loro presenza nel DB. Tramite la funzione `isset()` viene controllato l'avvenuto inserimento dell'email e della password prima di procedere con le altre operazioni.

La query per cercare l'account viene limitata ad un risultato (`LIMIT 1`) per ottimizzare la ricerca stoppandola alla prima lettura che soddisfi la condizione in quanto ogni account è univoco in tabella.

Tramite la funzione `strcmp` che restituisce 0 in caso di stringhe uguali viene controllato che la password inserita coincida con quella letta dal DB e a quel punto vengono impostate le variabili della sessione che permettono agli altri script di accedere a tali informazioni. Questi valori sono memorizzati in un array associativo apposito chiamato `$_SESSION`.

In caso di login fallito viene riproposto il form di accesso, se invece tutto procede con successo viene proposto il contenuto dell'homepage.

Il logout, gestito da `logout.php` è un semplice reindirizzamento all'homepage preceduto dalla distruzione della sessione in corso. Se nessun login è stato effettuato apre una sessione prima di eliminarla per evitare errori.

4.2.5 Indicizzazione immagini e cancellazione

La pagina `/imageList.php` accessibile tramite l'apposito tasto in homepage, in caso di accesso effettuato restituisce la lista dei quadri posseduti dall'utente scrivendo per ognuno tutte le informazioni inserite durante l'upload precedute dall'identificativo (*ID*) univoco e incrementale assegnato dal sistema ad ogni immagine, come mostrato in Figura 4.10.

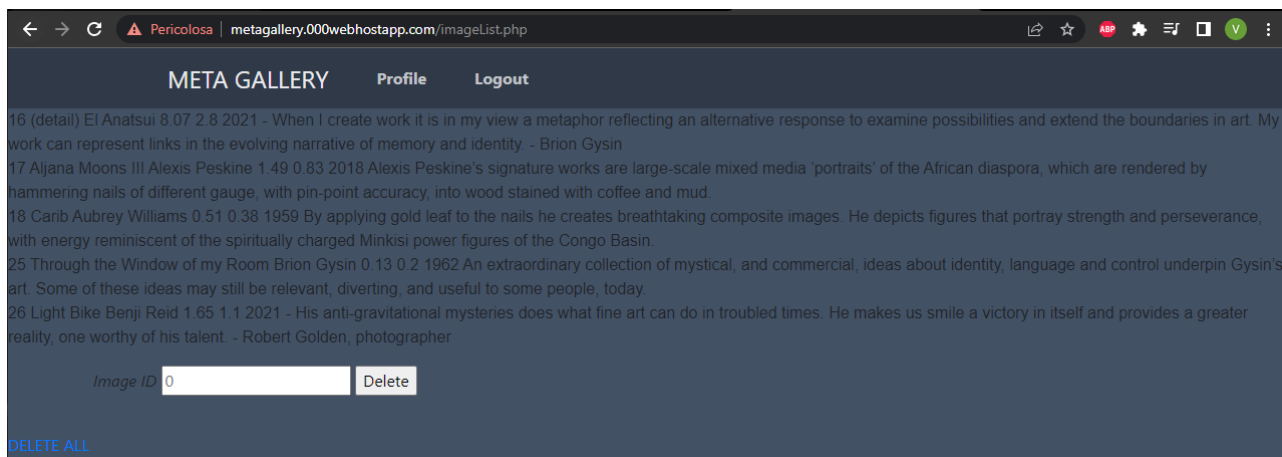


Figura 4.10: Pagina `imageList`

Subito dopo la lista è presente un form composto da un solo input oltre a quello di *submit* per inserire uno degli ID e rimuovere un'immagine tramite lo script `delete.php` indicato nell'action. Lo script esegue la cancellazione stampando un feedback seguito dal contenuto della pagina `imageList.php`.

Cliccando invece su "DELETE ALL" lo script presente al percorso `/scripts/deleteAll.php` si occupa di svuotare il record di immagini legate all'account con cui si è fatto il login scrivendo anche in questo caso un feedback seguito dal contenuto di `index.php`.

Sul portale web la lista dei quadri di un utente viene visualizzata in un documento HTML gestito dal codice `imageList.php` con la possibilità di richiamare i codici di eliminazione delle immagini. Tramite l'ID memorizzato nella sessione viene effettuata una `SELECT` di tutti gli attributi escluso quello dell'immagine. Infatti il contenuto di questo risulta lungo e non leggibile visivamente.

Ogni valore viene stampato andando a capo ad ogni quadro tramite il tag HTML `
`.

L'eliminazione delle immagini viene gestita da `delete.php` il quale legge l'ID dell'immagine da rimuovere dall'array di dati POST ed opera tramite la funzione SQL `DELETE`. Tramite `deleteAll.php` avviene invece l'eliminazione di tutte le immagini di un utente, ciò viene specificato nella condizione della `DELETE` eseguendo il confronto delle chiavi esterne della tabella *images* con l'ID dell'account acceduto.

4.2.6 Gestione degli inviti ad altri utenti

Come per gli altri bottoni presenti in homepage cliccando su “INVITE FRIEND” o “DELETE FRIEND” si viene reindirizzati alle pagine relative all’interno della gerarchia del sito web <https://metagallery.000webhostapp.com/>. Anche queste due funzionalità richiedono il login.

Alla pagina *inviteFriend.html* è presente un semplice form per l’inserimento di un’email da invitare seguita dal tasto per tornare in homepage (Figura 4.11).

Come per gli altri form di questo tipo l’action indica uno script per il trattamento dei dati trasmessi in modalità POST: */scripts/postInvite.php* il quale gestisce la richiesta associando nel database l’account invitato a quello del profilo dell’utente acceduto. A seguito del messaggio di feedback viene integrata nuovamente la pagina *index.php*.

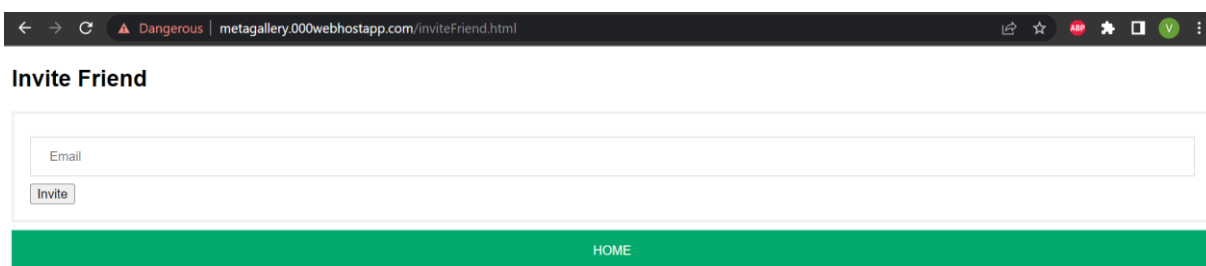


Figura 4.11: Invito ad un amico

La pagina *deleteFriend.html* è costituita da un form totalmente analogo a quello di invito il cui script di action (*deleteFriend.php*) esegue le operazioni necessarie sul database e, come operazione comune in questi casi, stampa un feedback seguito dall’homepage.

Un invito permette all’utente a cui è stato inviato di visitare l’invitante; al momento è necessario che entrambi gli account siano registrati senza una gestione tramite email o link di invito di un accesso come ospite. L’utente invitato accedendo al client visualizza una lista contenente gli altri mondi virtuali visitabili.

L’invito che l’utente crea tramite il form viene trasmesso come dato POST allo script *postInvite.php*. I dati disponibili per il codice sono l’ID dell’utente acceduto in sessione e da questa, mediante una *SELECT*, recupera nella tabella *accounts* il relativo indirizzo email. Dell’email letta dal form invece, al contrario, tramite un’altra *SELECT* si accede all’identificativo dell’utente invitato.

Avendo tutte le informazioni necessarie viene effettuata un’altra query di selezione estraendo dalla tabella *friends* le righe con le due chiavi esterne uguali ai due identificativi, nel preciso ordine previsto (invitante, invitato). Ciò serve ad evitare l’inserimento di una riga doppia contenente le stesse informazioni di invito di una scritta in precedenza, il quale oltre ad essere ridondante può portare a conflitti o visualizzazioni multiple della stessa riga a livello grafico. In tal caso viene comunicato a schermo che l’utente in questione risulta già invitato.

Se la selezione non restituisce nessun risultato si può procedere all’inserimento in quanto tupla univoca, preceduto da un ulteriore controllo sulle stringhe dei due ID (con relativo feedback grafico in caso di condizione verificata) tramite *strcmp()* per evitare di invitare se stessi (ID uguali con funzione che restituisce 0). Se i dati sono tutti corretti avviene la query *INSERT* dentro la tabella *friends* dei quattro dati disponibili.

4.2.7 Upload dei quadri ed elaborazione dei dati

Il form presente in homepage per il caricamento di un'immagine è il modulo più lungo da compilare, l'action è `/script/upload.php` e i campi sono tutti obbligatori.

Per l'immagine viene utilizzato l'input di tipo *file* che permette al browser di comunicare con il file manager del sistema operativo e selezionare un oggetto di qualunque estensione. Il nome e l'autore sono di tipologia testuale mentre le dimensioni numeriche con virgola (il numero di cifre decimali è specificato nel campo *step*), mentre l'anno è di tipo intero.

Vi è di seguito un elemento di tipo *checkbox* per permettere all'utente di specificare se nell'immagine caricata è già presente la cornice, ciò ha delle ripercussioni nel modo in cui l'eseguibile interpreterà i file scaricati applicando le immagini (texture) all'interno di una cornice digitale se questa non è presente nel file caricato. Si assume di default che la foto caricata contenga solo il quadro senza cornice inviando come valore *true* il numero 1 e come *false* uno 0.

Il valore specificato nel campo "value" della checkbox viene trasmesso solo se questa viene spuntata, altrimenti l'array associativo di dati nella posizione relativa a tale input risulta vuoto (*null*), facendo fallire eventuali operazioni su di esso. Per trasmettere il valore 0 di default quindi si è adottata la seguente tecnica: nel codice html del form, subito prima della checkbox è presente un input di tipo *hidden*, quindi non visibile graficamente, il quale ha lo stesso nome dell'input della checkbox ovvero *frame*. Il nome degli input, infatti, trasmesso in POST ad uno script permette di accedere tramite un array di dati in modo associativo ad ogni campo interessato; quindi, invece di utilizzare un indice numerico, l'indice è costituito dalla stringa contenuta nei campi *name*.

Tramite questo procedimento il browser per il nome *frame* trasmette il valore contenuto nell'input invisibile a meno che non si spunti la checkbox, sovrascrivendo il medesimo valore.

Per la descrizione non viene utilizzato il tag `<input>` per via della possibilità di creare solo caselle di dimensione ridotta senza la capacità di andare a capo, indicato più per informazioni singole e concise. Per permettere di scrivere un testo all'utente si utilizza il tag `<textarea>` che può essere inserito anche al di fuori del form purché venga specificato nell'attributo *form* l'id dello stesso, contenuto nel tag principale del modulo.

Lo script associato al form di upload è situato al percorso `/scripts/upload.php`.

Quest'ultimo esegue le operazioni di comunicazione e salvataggio sul database non prima di aver eseguito un controllo sulla natura del file trasmesso: se non è di estensione *jpg*, *png*, *PNG*, *jpeg*, *gif* questo viene comunicato, così come il successo o il fallimento del caricamento sul database.

In ogni caso la struttura visiva della pagina risulta composta da un feedback seguito dal contenuto della homepage per permettere un nuovo caricamento.

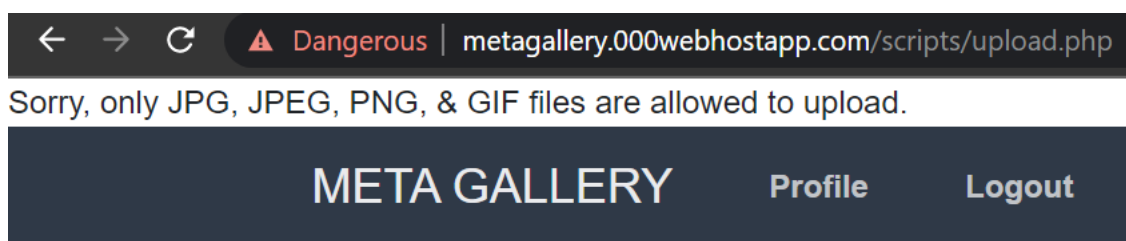


Figura 4.12: Esempio di upload fallito

Lo script `upload.php` si occupa della gestione dell'inserimento dei quadri, in particolare l'interpretazione del file binario inserito come immagine; di questo, infatti, viene subito controllata l'estensione per limitare l'inserimento di soli file immagine e preparando il file ad essere caricato.

La lettura degli altri dati POST avviene in maniera classica tramite il vettore associativo rimpiazzando il carattere `\n` presente in *description* con `\r` in caso di ritorni a capo nel testo. Questo è necessario in quanto il carattere `\n` viene interpretato dal codice di lettura dei dati nell'applicativo come separatore tra quadri diversi e non tra i singoli campi, portando a letture non corrette se non gestito.

A questo punto i campi vengono inseriti nella tabella *images*.

4.3 Dialogo con l'applicazione simulativa

L'applicazione simulativa dialoga con il server attraverso tre script php inviando le richieste HTTP insieme ai relativi dati da processare. Tali codici restituiscono i dati sotto forma di caratteri testuali ed eventuali messaggi di errore leggibili dal client.

La struttura dei codici è simile tra loro: sono infatti presenti tre programmi, come mostrato nel **Diagramma 6**, che gestiscono le richieste POST, ovvero elaborano un form proveniente dall'esterno; infatti, il server non conosce la natura del client e lavora in maniera analoga a quanto descritto con il portale web.

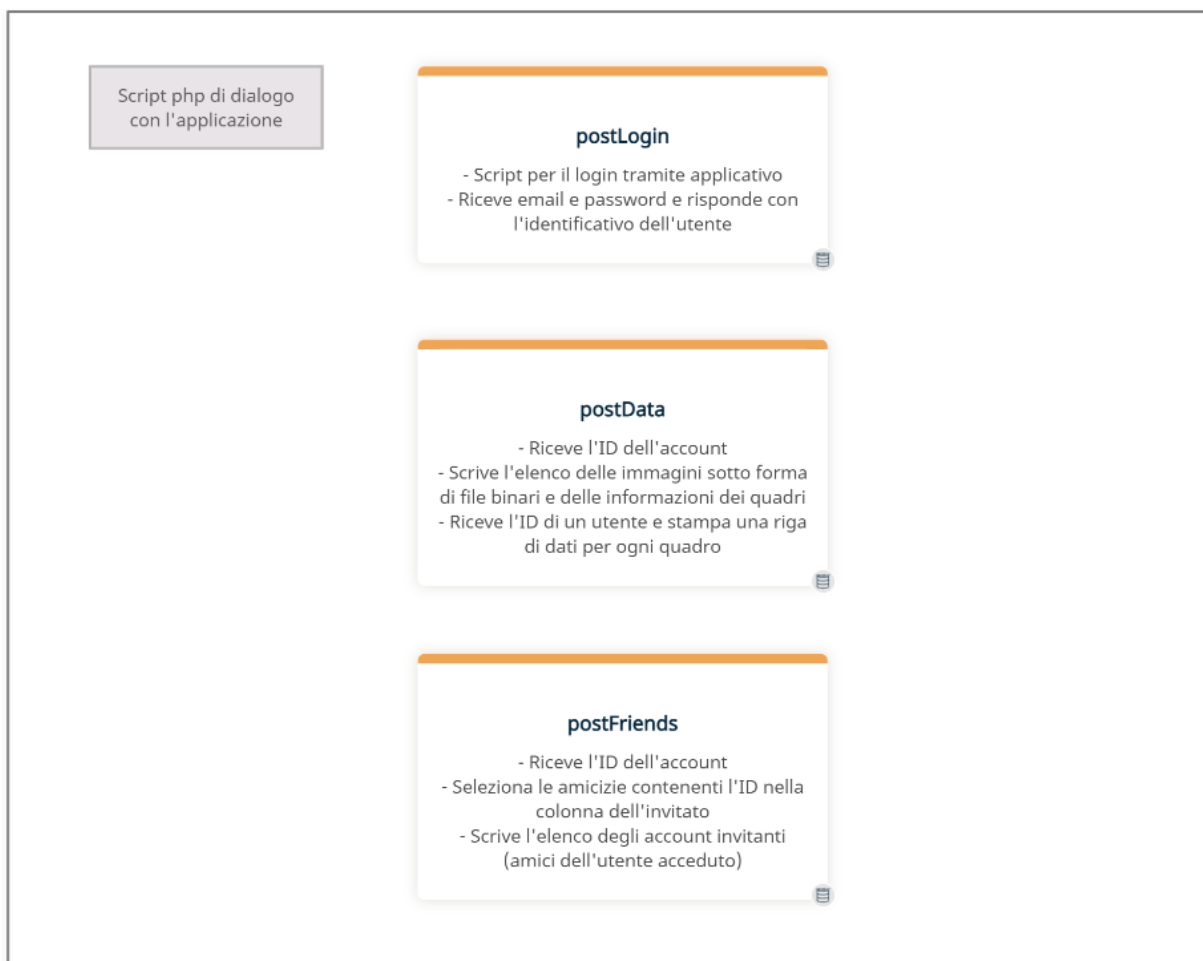


Diagramma 6: Script php richiamabili dall'applicativo WebGL

4.3.1 Script di richiesta della lista di quadri di un utente

Il bottone “Post Data” richiama la pagina `requestPostData.html` costituito da un piccolo form, mostrato in **Figura 4.13**, per l’inserimento dell’ID di un utente (funzionamento analogo all’identificativo delle immagini), il cui action è uno script (`postData.php`) che restituisce tutte le immagini legate all’account specificato nell’ID; ciò rappresenta un procedimento di debug e testing veloce per visualizzare la lista di immagini tramite richiesta POST così come farebbe l’applicativo.

Lo script `postData.php` è infatti il codice di destinazione della richiesta effettuata dall’applicazione simulativa per scaricare i quadri di un utente (inviando un form come quello presente in `requestPostData.html` tramite funzioni di dialogo con un server web offerte da Unity), risulta quindi utile poter simulare tale procedura nel sito web.

Il form mostrato in Figura 4.13 è composto da un campo di input con l'etichetta "ID:" a sinistra e un pulsante "submit" rettangolare con sfondo grigio e testo bianco, posizionato sotto il campo di input.

Figura 4.13: Form di richiesta dati dei quadri in POST

Tramite un’operazione di SELECT vengono lette le righe relative all’account confrontando l’ID richiesto con la chiave esterna della tabella `images`.

Per consistenza con il client il quale differenzia ogni campo usando come separatore lo spazio tra stringhe, vengono sostituiti gli spazi vuoti interni ad ogni valore di testo con un trattino basso. Tramite la funzione `base64_encode()` avviene l’interpretazione su base 64 bit del file binario dell’immagine per la traduzione in codici ascii leggibili dall’applicativo.

4.3.2 Script di login e di richiesta dell’elenco di amicizie di un utente

Il login effettuato sul client WebGL esegue una richiesta HTTP verso lo script `postLogin.php` inviando un form di accesso contenente email e password. Il codice in questo caso non ha necessità di aprire una session e invia come risposta uno `0` in caso di login fallito e invece in caso di successo un `1` seguito dall’ID dell’utente.

Lo script `postFriends.php` permette al client di richiedere la lista di amici dell’utente con il quale si effettua il login, ovvero selezionando tutte le righe in cui l’ID dell’utente si trova dentro il campo `ID2` e quindi tutti i casi in cui è stato invitato. Come output vengono quindi stampati l’ID e l’email dell’account invitante per ogni tupla le quali vengono lette e gestite dall’applicativo.

5 Applicazione di simulazione virtuale WebGL

Nel capitolo seguente viene approfondita nel dettaglio l'applicazione WebGL strutturata su un progetto Unity analizzando la struttura del medesimo, le feature grafiche (2D 3D), gli asset utilizzati e il codice dietro ad ogni funzione.

5.1 Architettura del progetto

La versione di Unity utilizzata per lo sviluppo della prima demo di *Meta Gallery* è la 2021.3.4f1, la compatibilità con altre distribuzioni dell'intorno di anni relativo al 2021 è garantita quasi totalmente, con alcune accortezze per quelle uscite prima del 2020 per quanto riguarda la gestione delle richieste HTTP utilizzando alcune librerie in parte deprecate per la gestione degli errori di connessione.

Il progetto ha come nome “*Meta-Gallery-URP*” e si dirama in una gerarchia di cartelle tipica di Unity, contenenti tutti gli asset utilizzati, compresi modelli e texture. Vengono inoltre utilizzati alcuni asset esterni per una più comoda implementazione di certe funzionalità.

Le scene, ovvero mondi/livelli virtuali visitabili sono due: *MainMenu* e *House*, strutturate come descritto nel **Diagramma 7**.

La prima presenta un menu di accesso per il login, alcuni controlli per le impostazioni, le amicizie, l'uscita dal programma, l'accesso al portale web e lo *start* del livello *House*, ovvero il vero e proprio mondo 3D costituito da una villa con i quadri scaricati appesi alle pareti. Un menu di gioco simile a quello principale è disponibile *in game* durante l'esperienza nella scena *House* accessibile in ogni momento tramite chiamata da tastiera.

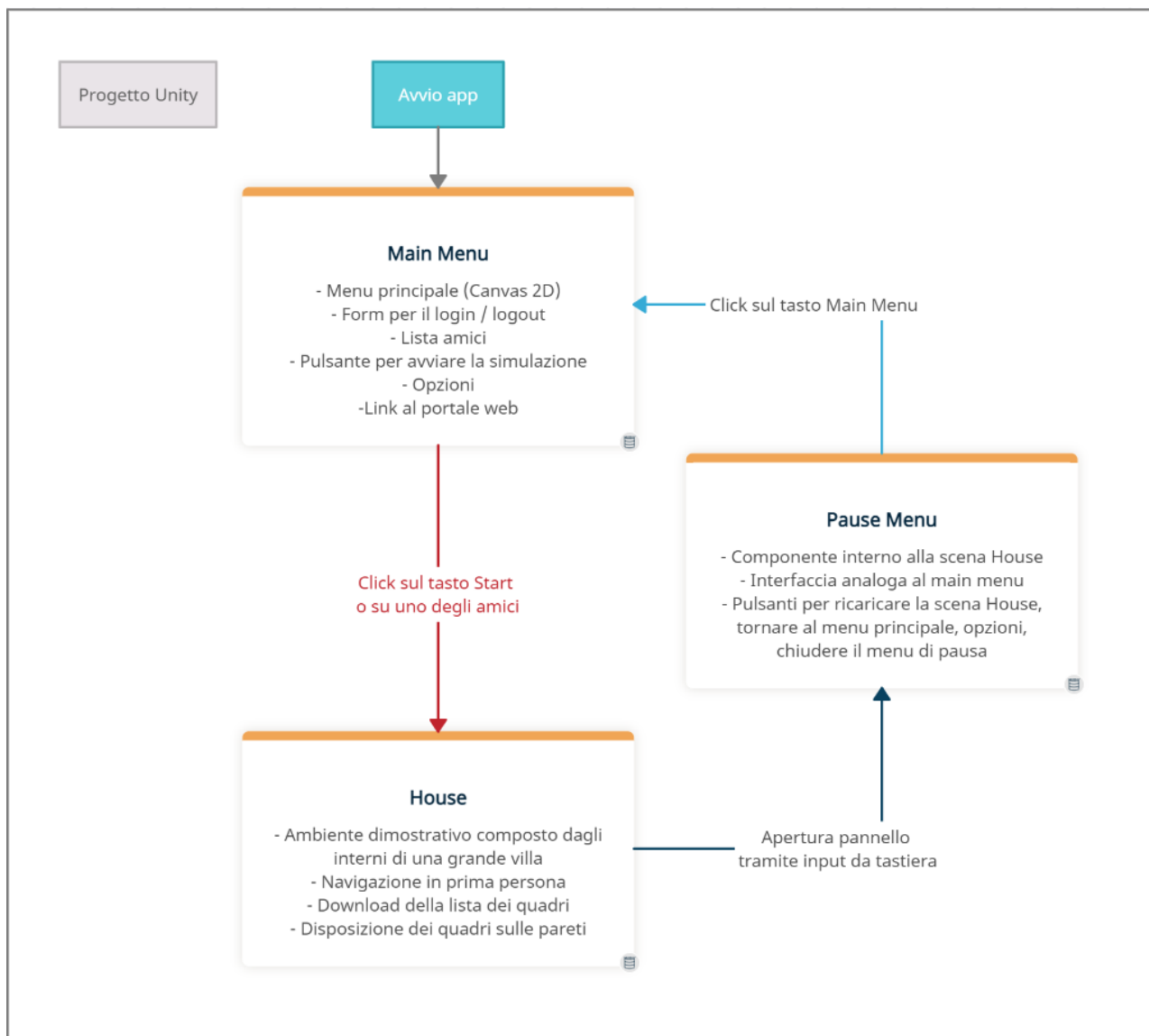


Diagramma 7: Architettura scene del progetto Unity

5.2 Struttura grafica della scena Main Menu

La scena Main Menu contiene gli oggetti dell'interfaccia 2D e i componenti per la l'interazione con la GUI (Diagramma 8). Lo stile grafico ed alcune funzionalità (con qualche operazione di adattamento del codice al contesto di Meta Gallery) sono offerte da un asset esterno chiamato *Primo* [25], personalizzato per ottenere un'interfaccia a pannelli. Il primo presenta il form di login e i tasti per avviare la scena House, aprire le opzioni ed uscire dall'app. Effettuato l'accesso è disponibile il logout e la lista di amicizie.



Diagramma 8: Elementi grafici del menu principale

5.2.1 Gerarchia degli oggetti

Nel pannello *Hierarchy* dell'ambiente di sviluppo di Unity sono visibili gli oggetti in scena, ovvero quegli elementi 2D, 3D o non visibili i quali hanno una posizione globale all'interno dell'ambiente virtuale caratterizzato da un'origine e dai tre assi vettoriali X,Y,Z. Alcuni di loro possono essere attivi o meno, cosa valida anche per i componenti.

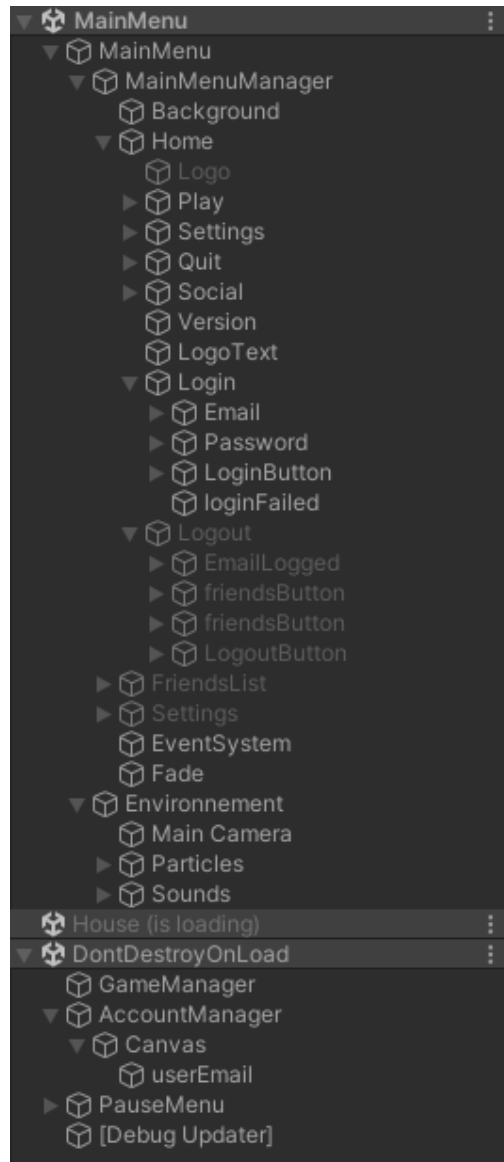


Figura 5.1: Gerarchia oggetti Main Menu

Durante lo sviluppo è possibile provare l'app tramite il tasto *Play* e nella gerarchia è visibile la sezione degli oggetti *DontDestroyOnLoad*, ovvero l'insieme di elementi da lasciare intatti in termini di componenti, variabili e *transform* (posizione, rotazione, scala).

Tra questi *Game Manager* e *Account Manager* che sono oggetti (game object) di tipo *Empty* e quindi invisibili, privi di un modello, i quali contengono informazioni comuni ad entrambe le scene, così come per il menu di gioco chiamato *PauseMenu*. È presente, inoltre, un Canvas UI testuale (*userEmail*) per visualizzare in basso l'email dell'utente dopo il login.

Il game object *Main Menu Manager* contiene i componenti di gestione dei Canvas e dei comportamenti degli elementi di Primo, parallelo a *Environment* contenente dei particellari, delle sorgenti audio di atmosfera e la camera virtuale. I figli comprendono l'immagine di sfondo, il pannello *Home* con tutti gli elementi inizialmente visibili e i pannelli con la lista delle amicizie e con le opzioni.

Per ultimi *EventSystem* i cui componenti sono necessari a Unity per gestire gli input e le selezioni (anche il *Raycasting* utilizzato nella scena seguente) e *Fade* per l'animazione di dissolvenza a nero.

Dentro *Home* ci sono tutti i bottoni del menu composti da immagine e testo e gli input del login nel formato standard di Unity che comprende una maschera per lo spazio per non uscire dallo spazio di scrittura, una stringa di placeholder e la possibilità di reagire alla selezione e focus del mouse; in più, mediante l'oggetto *loginFailed* (canva di tipo testuale), viene stampato un feedback in caso di credenziali non corrette. Nel logout di nuovo i bottoni e le informazioni relative.

Gli oggetti del menu sono tutti contenuti dentro un ulteriore empty chiamato *MainMenu* il quale contiene degli script personalizzati indipendenti dal plugin.

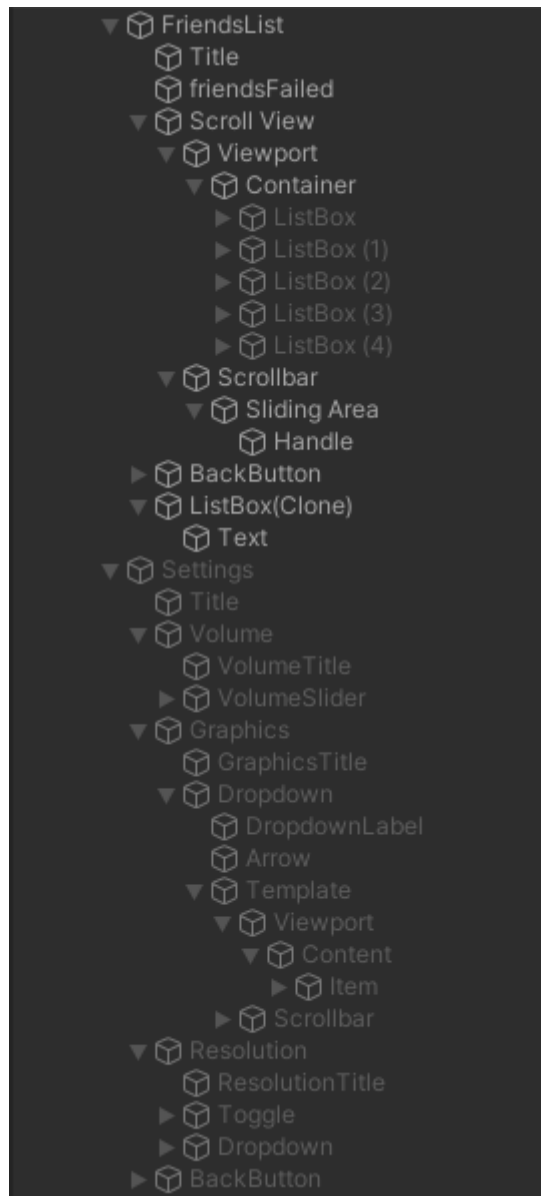


Figura 5.2: Oggetti della scroll view delle amicizie

La lista di amicizie è composta da un testo con il nome del pannello, un testo per i feedback, la barra di scroll a lato, il tasto *Back* e la *Scroll View* ovvero l'insieme di oggetti per la creazione dei box contenenti i dati delle amicizie (ogni istanza appare come *ListBox(Clone)*). *Title* e *BackButton* sono analoghi a quelli di default presenti nel pannello *Settings* il quale rimane identico a come presente nel preset di Primo, ovvero composto dagli elementi per il volume (una barra a trascinamento), la grafica (menu a tendina per la

scelta del preset di opzioni grafiche più o meno pesanti a livello computazionale) e risoluzione (menu a tendina) il quale invece rimane disattivato al momento con la sola possibilità di selezione in *Graphics*.

5.2.2 Interfaccia utente ed elementi utilizzati

Il menu è per sua natura composto da elementi 2D visibili da una camera virtuale che identifica quali oggetti visualizzare ad ogni frame. Nello specifico l'insieme di questi elementi grafici di UI vengono definiti da Unity *Canvas*, i cui componenti descrivono il comportamento di visualizzazione in qualità di ridimensionamento, disposizione e caratteristiche grafiche, relativizzandole in base alle dimensioni della finestra di visualizzazione o alle coordinate globali del mondo di gioco con le relative interazioni e distanze dalla camera. L'interfaccia è divisa in pannelli relativi alle diverse funzioni, come mostrato in Figura 5.3.

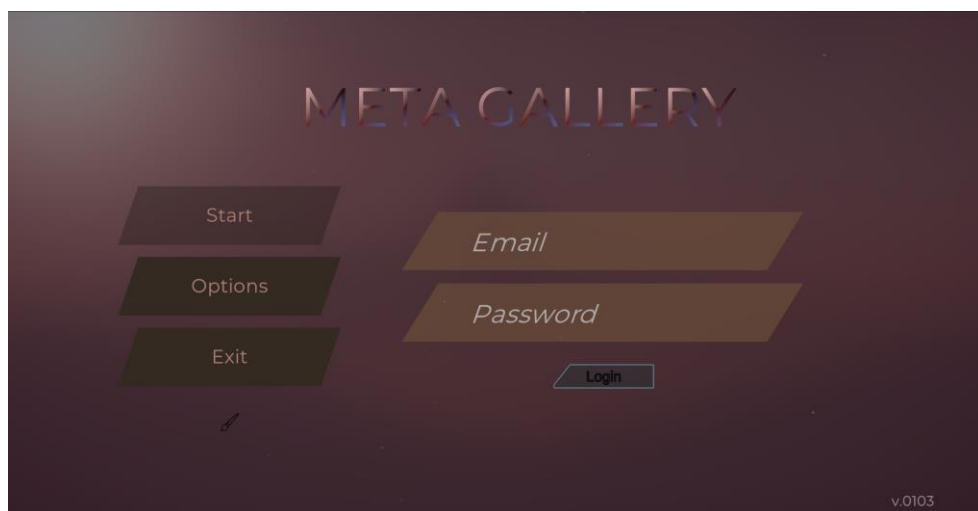
Il menu si presenta all'utente con un logo, il numero di versione del software e degli elementi di interazione. I tasti presenti si basano tutti sulla stessa base (tranne *Login* e *Logout*), ovvero un oggetto di default memorizzato che prende il nome di *prefab*, cambiando alcune caratteristiche grafiche. Il tasto di Start non è cliccabile fino all'effettuazione del login, quello di opzioni apre il pannello *Settings* come previsto dall'asset e quello di *Exit* rimane da implementare: infatti, se in un classico programma è necessario permetterne la chiusura, in un prodotto WebGL questa può essere demandata ai controlli del browser (chiusura scheda).

Il login viene visualizzato tramite caselle di input per scrivere le proprie credenziali e poi mostrare l'email utilizzata, il tasto *Friends* e abilitare il tasto start una volta proceduto con l'accesso o mostrando un messaggio di errore per riprovare in caso di email o password errate.

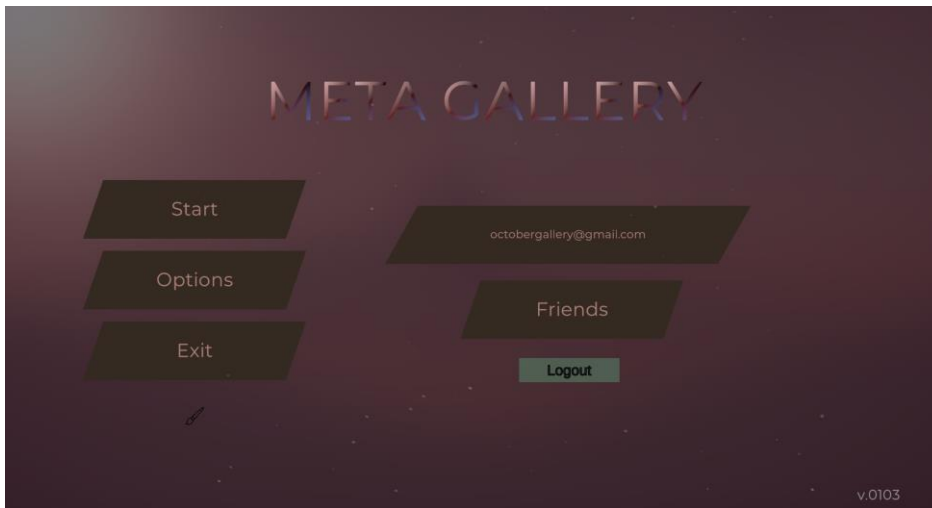
La pagina degli amici appare come un pannello da popolare con i possibili utenti visitabili, di cui viene visualizzata una lista navigabile tramite *scrollbar*.

Vi è inoltre un tasto composto dall'icona di un pennello per l'apertura del portale web in un'altra scheda.

Lo sfondo è una personalizzazione in termini di colore di uno dei temi utilizzabili con l'asset in questione, è composto da un'immagine statica e dei particellari.



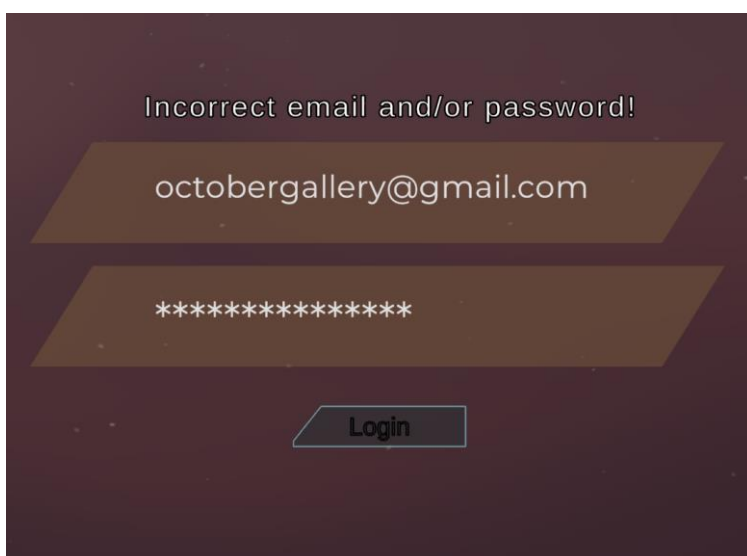
(a)



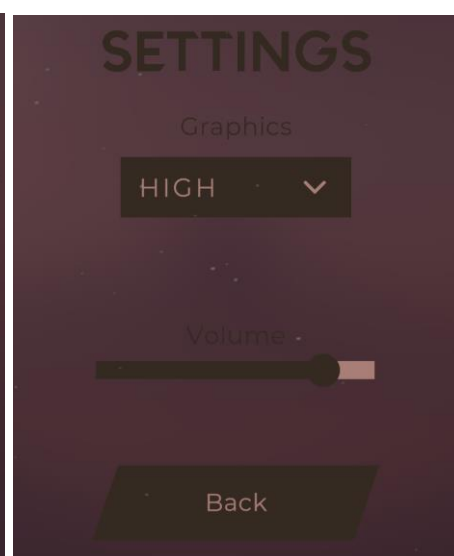
(b)



(c)



(d)



(e)

Figura 5.3: Pannelli del Main Menu

5.3 Features ed algoritmi del menu principale

I comportamenti degli elementi del menu sono regolamentati dai propri componenti, in particolare gli script C# tramite i quali si accede alle funzionalità di tutti gli oggetti mettendoli in relazione tra loro, come mostrato nel **Diagramma 9**.

Parte delle feature principali sono presenti nello script fornito dall'asset esterno Primo e il resto è stato sviluppato per adattare il preset del menu alle necessità del progetto.

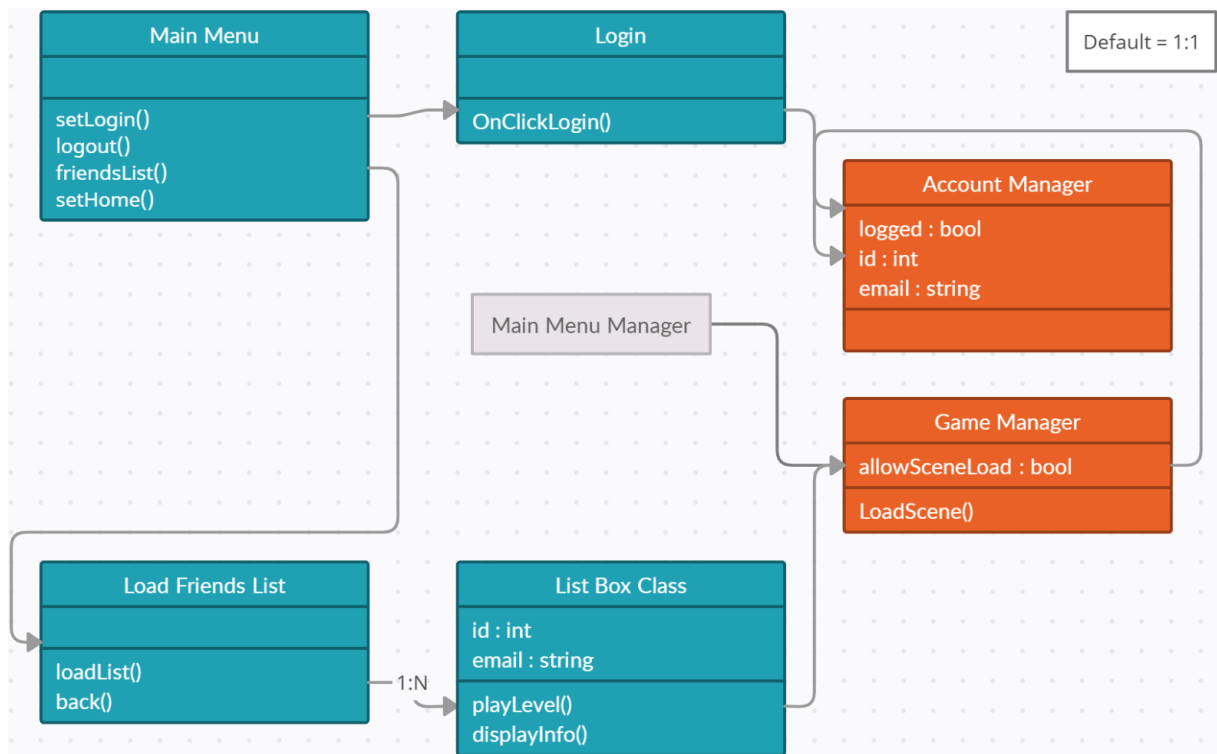
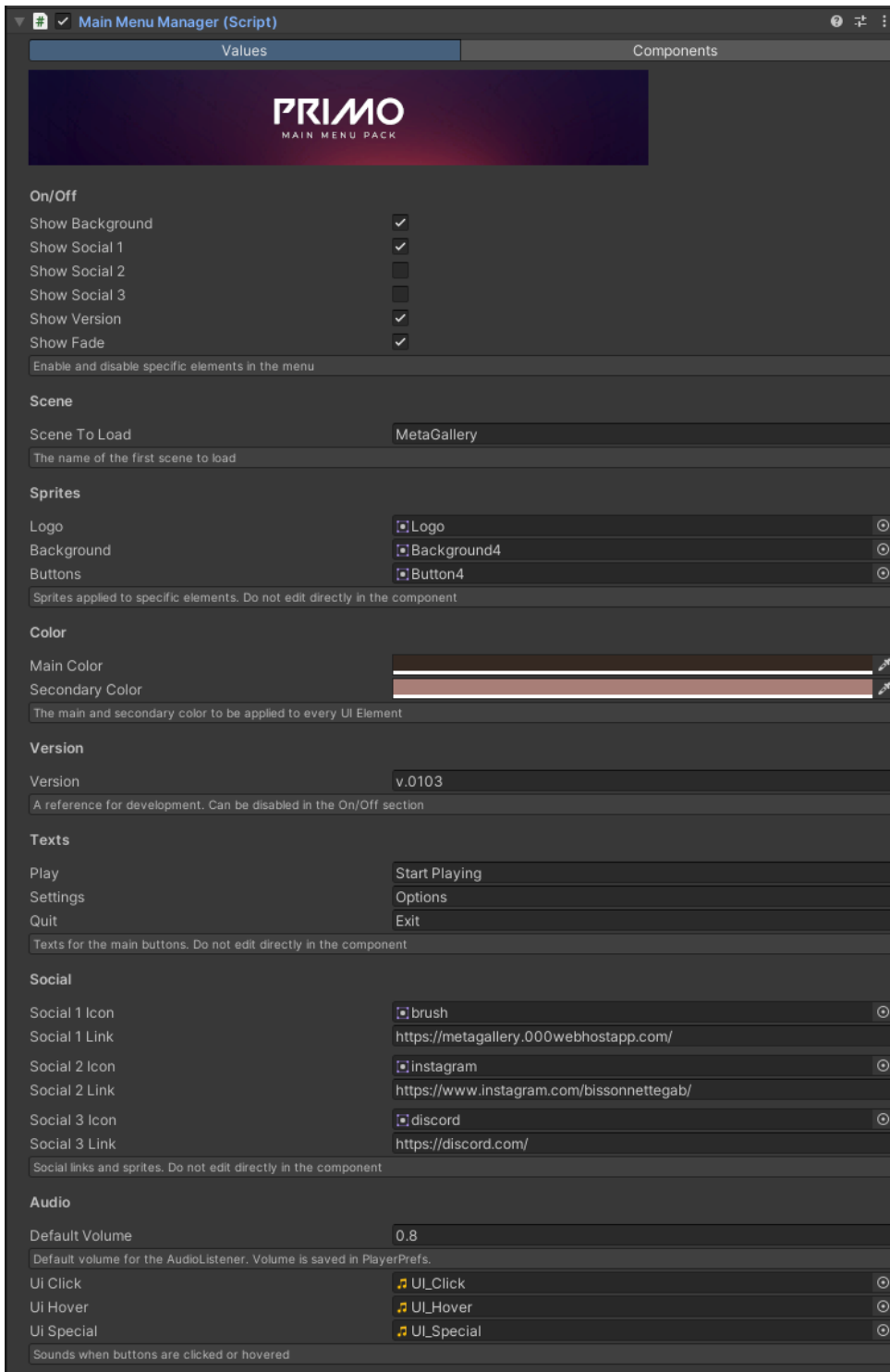


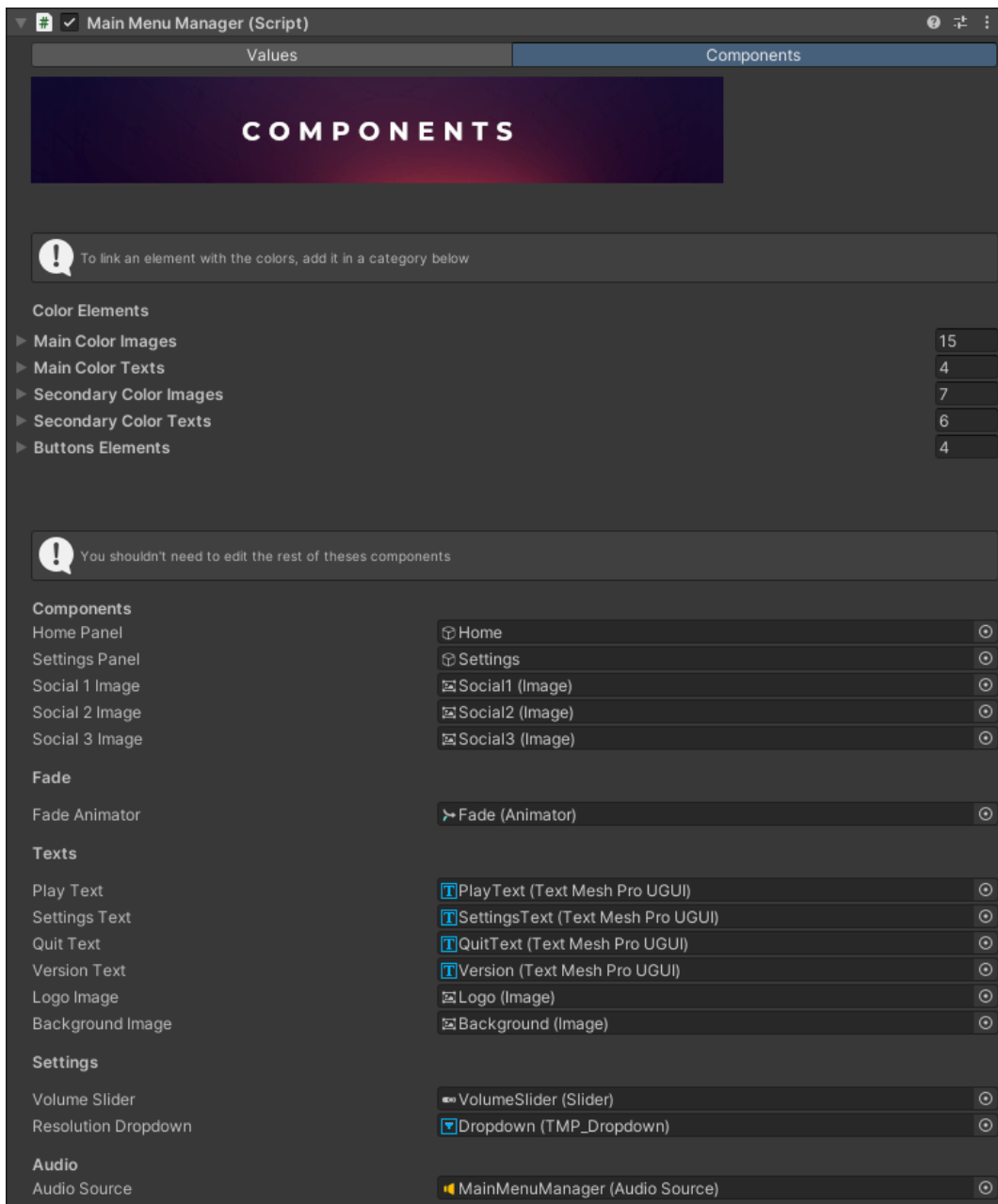
Diagramma 9: Diagramma delle classi della scena Main Menu

5.3.1 Gestione dell'interfaccia e degli elementi 2D

L'asset del menu viene gestito da un oggetto *Manager* il cui componente principale è lo script *Main Menu Manager* (utilizzando l'interpretazione di Unity dei nomi dati agli script, infatti il file è nominato in realtà *MainMenuManager.cs*) scritto in modo da essere modificabile quasi totalmente solo tramite l'interfaccia del game engine che permette di accedere alle variabili tramite la GUI specificando in fase di codice la loro natura pubblica o con un processo detto di serializzazione.



(a)



(b)

Figura 5.4: Main Menu Manager

La pagina *Components* non risulta modificata rispetto al preset mentre in quella dei valori sono stati personalizzati i colori utilizzati, la versione visualizzata in basso, il testo dei bottoni presi dal plugin, specificata la presenza di un solo social (ovvero l'icona cliccabile del portale web) e soprattutto la scena da caricare alla pressione del tasto Start. La gestione dei componenti aggiunti al preset, quindi tutto ciò che ha a che fare con il login e le amicizie, è gestito dallo script `MainMenu.cs`

Le variabili pubbliche dichiarate in testa al codice permettono di selezionare dei componenti tramite l'interfaccia dell'editor di Unity scegliendoli dalla gerarchia, mentre i componenti univoci possono essere ricercati tramite la funzione `C# FindObjectOfType`

per una gestione interna al codice e più controllata. È questo il caso dei vari script di tipo *Manager* in quanto sempre unici in scena.

Durante la creazione dell'oggetto ad inizio scena la funzione *Start* descrive alcune operazioni preliminari di configurazione e, oltre ad assegnare i componenti alle variabili da utilizzare in seguito, imposta la visualizzazione del cursore durante il *play* dell'app, in modo di mantenere attivo il puntatore del mouse a schermo, essenziale per navigare, muovere la camera ed interagire sia con gli oggetti 3D che con le interfacce 2D.

Il componente (script) *Main Menu* mette a disposizione delle funzioni pubbliche, quindi accessibili da codici esterni utili ad esempio da chiamare in caso di evento come la pressione di un tasto.

Infatti con la funzione *setLogin()* rende cliccabili i tasti *Start* e *Friends*, popola la stringa testuale dell'email dell'utente da visualizzare e attiva il pannello per il logout al posto di quello attuale.

logout() al contrario disattiva i bottoni in caso per ridare la possibilità di effettuare il login (è presente un flag in *Account Manager* per il login avvenuto o meno).

friendsList() disattiva il pannello home per cambiare totalmente l'interfaccia ed aprire quello di visualizzazione delle amicizie.

Qui sono anche presenti le funzioni per l'apertura e chiusura del menu di pausa che però in questa scena non vengono mai richiamate.

Un altro script associato al *Main Menu* è *InputNavigator.cs* per la navigazione tramite il tasto *Tab* tra un input di login e l'altro potendo premere *Invio* sulla tastiera invece che il tasto relativo con il mouse. Ciò viene continuamente controllato nella funzione *Update()* che come la *Start* è presente in ogni componente Unity e viene eseguita ad ogni frame di gioco, utile quindi per descrivere il codice da eseguire continuamente. In questo caso i controlli vengono eseguiti utilizzando le direttive offerte dalla classe *Input* di *UnityEngine*.

5.3.2 Game Manager, Start e caricamento asincrono

La più importante ottimizzazione a livello di codice, oltre a tutti gli accorgimenti grafici di rendering e di dimensione fisica della build, è la velocizzazione del tempo di caricamento della scena della casa. Il menu, infatti, è la prima scena da caricare per il browser ma non è di natura eccessivamente pesante: il tutto è molto statico e gli elementi oltre a non essere numerosi sono 2D, anche i particellari utilizzati non sono in gran numero.

Il principio su cui si basa questa ottimizzazione è quello del caricamento asincrono; infatti, l'interazione con il menu da parte dell'utente è minima e soprattutto non frenetica lasciando all'elaboratore numerosi secondi di pausa tra un input e l'altro. È così che il sistema prevede all'avvio della scena del menu il caricamento della successiva (visibile anche in gerarchia) in modo asincrono, quindi senza bloccare l'interazione e attendendo un evento per eseguire il cambio scena effettivo. La scena caricata attende quindi la pressione del tasto *Start* per essere lanciata definitivamente, ciò grazie all'interazione tra gli script del button, del menu e di *GameManager.cs*.

Il *Game Manager* è lo script di gestione delle scene, controlla e conosce dell'esperienza. Grazie al discernimento del numero di scena, oltre alle operazioni sul menu di gioco offre la funzione pubblica di caricamento della scena, da richiamare dall'esterno. In *LoadScene(int s)* viene impostata una variabile *allowSceneLoad* grazie alla quale selezionare un tipo di caricamento asincrono con successiva attesa di un evento o istantaneo. Infatti, se si è nella scena della casa e viene lanciata quella del menu (il numero incrementale relativo a tale scena è lo 0) il parametro *s* vale 0 e la variabile

diventa *true* facendo un caricamento istantaneo della scena senza un precedente caricamento.

Se si è nel menu invece sicuramente ci sarà la necessità di passare alla scena *House*, il parametro in questo caso è maggiore di 0 impostando *allowSceneLoad* a *false* prima di lanciare la Coroutine¹¹.

Nella funzione lanciata per il caricamento asincrono tramite la libreria *SceneManager* viene effettuato il load della scena fino al 90% di progresso, a questo punto, mediante un ciclo condizionato, la funzione rimane in attesa di leggere il valore *true* nella variabile *allowSceneLoad*. Essa in caso di caricamento istantaneo o pressione del tasto Start nel menu risulta positiva permettendo di terminare il caricamento e lanciare la scena.

MainMenu.cs sfrutta il comportamento delle funzioni del game manager chiamando il caricamento della scena della casa (la numero 1) che quindi attenderà il permesso di lanciarla una volta arrivato al 90% di progresso. Ciò avviene alla pressione del tasto start come impostato nel componente *Button* dello stesso nella funzione *OnClick()*.

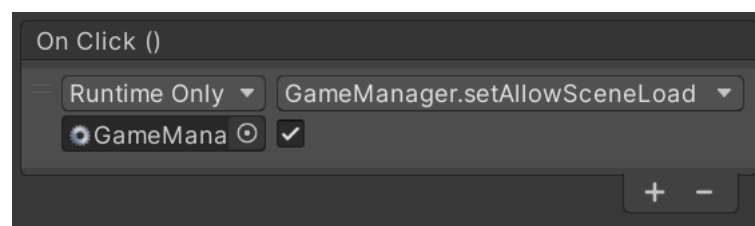


Figura 5.5: Caricamento asincrono della scena alla pressione del tasto Start

Il Game Manager come altri oggetti mantiene il suo stato al passaggio di una scena e l'altro tramite lo script associato *DontDestroy.cs*.

La funzione principale è semplicemente *DontDestroyOnLoad()* ma il codice si occupa di ricercare in ogni scena altri game object con lo stesso tag, il quale essendo univoco per gli oggetti in questione indica la presenza dello stesso elemento in scena (array di object di lunghezza maggiore di 0) procedendo alla rimozione di sé stesso.

Essendo effettuato nella funzione *Awake*, questo controllo permette la distruzione del secondo oggetto in ordine di creazione, quindi al cambio di scena (momento in cui viene eseguita la *Awake* solo nel nuovo oggetto) continuerà a sopravvivere l'originale, a patto che in entrambe le scene tale script sia presente come componente al medesimo tag.

5.3.3 Pannelli per il login e logout

Gli oggetti per l'input e la visualizzazione di stringhe di testo sfruttano le librerie *TextMeshPro* divise in sottocategorie in base alle funzioni necessarie con la possibilità di gestire tutte le caratteristiche dei testi tramite codice o interfaccia. Il comportamento di input del login è quello standard offerto da tali librerie per ciò che riguarda email e password inserite sotto la direzione dello script *login.cs* aggiunto come componente al tasto omonimo, il quale dialoga con l'*Account Manager* responsabile di mantenere memorizzate le informazione dell'account dopo il login nel passaggio tra una scena e l'altra tramite l'utilizzo di variabili pubbliche oltre che di una funzione per il setting dell'email da visualizzare nell'interfaccia in basso a schermo.

¹¹ Funzione di gestione del tempo e di ordinamento cronologico delle operazioni [29] [30]

Nello script del login viene impostata la funzione di *listener* della pressione del tasto (analogo a farlo tramite interfaccia grafica) per raccogliere le informazioni inserite pronte ad essere inviate come richiesta POST online allo script php di gestione login. La creazione del form avviene tramite l'utilizzo della libreria *UnityWebRequest* che fornisce nella documentazione degli script standard per le operazioni classiche come quelle necessarie all'applicativo.

Nel dettaglio la richiesta è formata da una *coroutine* per la gestione della sessione HTTP la quale necessita di tempi di attesa tecnici per la creazione della connessione con tutti i controlli dettati dal protocollo, in cui creare (sfruttando le funzioni e i tipi di dato offerti dalla libreria) un form, aggiungendo man mano vari campi (email e password).

È possibile inviare una richiesta HTTP di tipo POST (o GET) a un dato indirizzo URL trasmettendo il form per dialogare con il server interpretando le risposte e gestendo le eccezioni, gli errori, i warning. Vengono infatti formalmente gestiti gli errori di connessione o di protocollo stampando il messaggio ricevuto contenuto nel campo *error* della variabile di connessione.

Se la richiesta procede con successo il risultato viene scaricato sotto forma testuale e da qui può essere implementata l'interpretazione dei messaggi in accordo con quanto atteso in termini di risposta del server. Per il login è presente il controllo del primo carattere ricevuto che in caso di *1* indica che l'accesso sia avvenuto correttamente, in caso contrario le credenziali non sono corrette. Se quindi il login è valido vengono configurati i flag e le variabili necessarie, in particolar modo *id* contenente l'identificativo dell'utente da convertire prima da stringa a numero intero tramite una funzione della libreria *Convert* a 32 bit, a questo punto viene lanciata la funzione *setLogin()* del *Main Menu*.

Il *logout* avviene solo tramite la chiamata alla relativa funzione pubblica dei *Main Menu* alla pressione del tasto per uscire dall'accesso.

5.3.4 Pannello delle amicizie

Una volta effettuato il login può essere scaricata la lista delle amicizie da visualizzare nel pannello comunicando con il relativo codice sul server. Lo script incaricato di queste funzioni è *LoadFriendsList.cs*.

Prima di operare sugli inviti è presente la classica inizializzazione dei componenti in base ai tipi ma invece di utilizzare la *Start* questa è presente nella funzione *Awake()* la quale analogamente alla *Start* ha come compito eseguire delle operazioni preliminari ma viene chiamata ancora prima (come anche la funzione *On Enable*) evitando errori di conflitto nel momento in cui più componenti dialogano tra loro, ad esempio nel caso di oggetto inizialmente disattivato ma comunque citato in altre funzioni o inizializzazioni esterne.

È presente una funzione per tornare indietro alla pagina *Home* del menu e tramite la struttura di *download* e gestione delle eccezioni per le richieste form (sottoponendo un form contenente solo l'ID dell'account in sessione sul client) viene scaricata la lista di amici, impostata sul server come una riga per ogni utente visitabile (divisi quindi dal carattere per andare a capo *\n*) con uno spazio presente tra ogni campo. La divisione delle stringhe sulla base di un carattere avviene tramite una funzione del tipo *stringa Split()*.

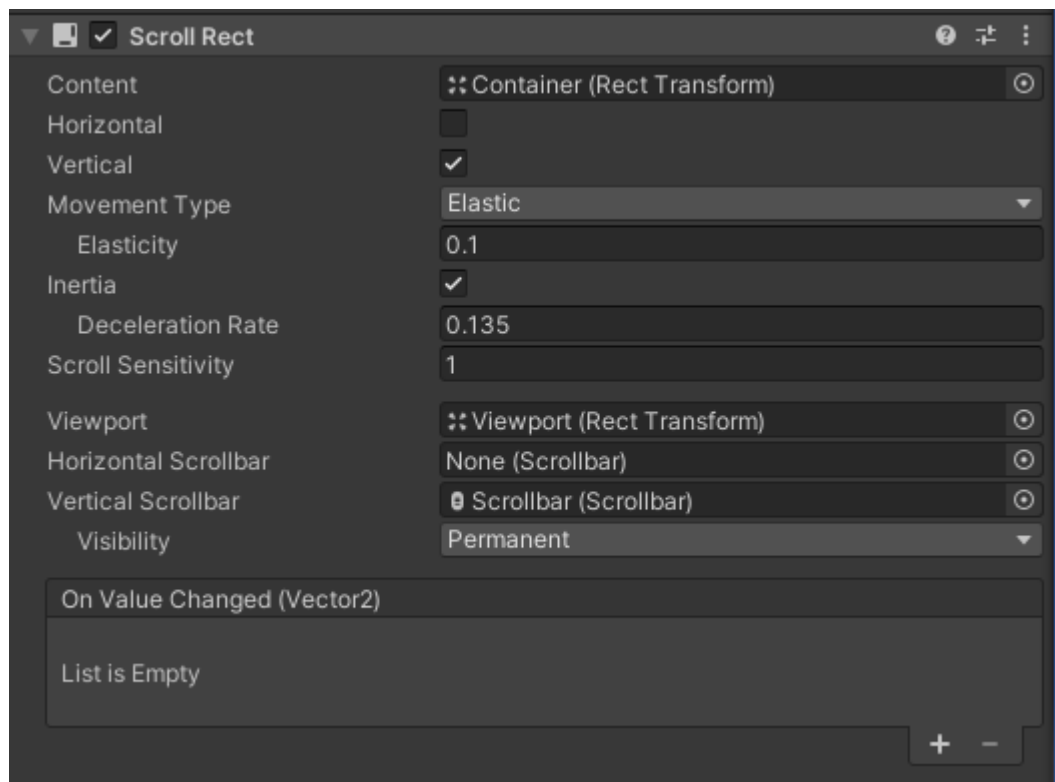
Per ogni riga scaricata viene istanziato un box da visualizzare nel contenitore *Canvas* descritto dalla funzione *instantiatedFriend*. Viene quindi creato un oggetto in scena del tipo dichiarato (il prefab *listbox*) ed effettuate le operazioni sulla *transform* per popolare man mano una lista e composti gli elementi per l'ID e l'email presenti in ogni box.

Ognuno dei contenitori è infatti descritto dalla classe `List Box Class` la quale permette la visualizzazione dell'indirizzo dell'account e dispone di una funzione di lancio della scena configurando l'account manager e liberando al fine del caricamento asincrono di House.

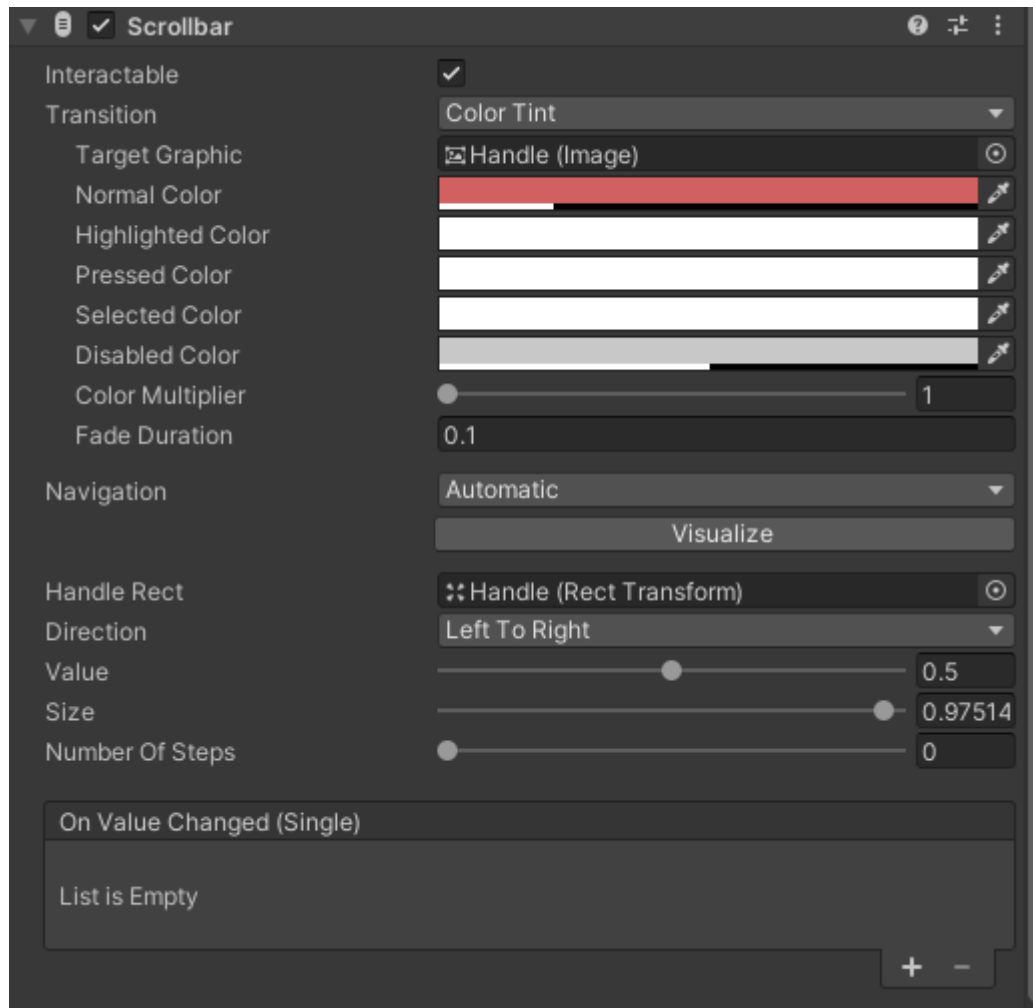
Per dare la possibilità di navigare tramite una *scroll view* quindi scorrendo una lista di lunghezza indeterminata vengono utilizzati dei componenti UI presenti in Unity senza l'utilizzo di asset esterni.

Un rettangolo esterno (*Scroll View* in gerarchia con il componente *Scroll Rect*) ha la possibilità di dichiarare la zona dove disporre la lista (*Viewport*) e la barra di scorrimento (*Scrollbar*), come mostrato in **Figura 5.6**.

La viewport ha inoltre associato il componente *Rect Mask 2D*, ovvero una maschera che rende visibili gli oggetti figli solo se all'interno dell'area descritta dalla stessa viewport, ciò permette di non visualizzare gli elementi in eccesso in caso di numerose amicizie istanziate per poi tramite lo scorrimento renderli visibili quando in posizione.



(a)



(b)

Figura 5.6: Componenti Scroll View

5.4 Menu di pausa

Il menu di gioco (o di pausa) è consultabile in ogni momento alla pressione del tasto *Esc* della tastiera bloccando il movimento di camera e personaggio sul posto (quindi in pratica il *Mouse Manager*, script di gestione dell'input e del discernimento dei comportamenti successivi a un singolo click o a due ravvicinati ovvero il doppio click). Con lo stesso tasto *Esc* o il button *Resume* è possibile tornare *in game*.

Ciò che appare all'utente è una struttura Canvas nello stesso stile del menu principale, di questo infatti utilizza in parte gli stessi asset semplificandolo ed adattandolo ad altre funzioni.

Nel dettaglio *Pause Menu* offre la possibilità ricaricare il livello in corso ritrovandosi quindi nuovamente al punto di partenza della scena della casa, le opzioni analoghe al *Main Menu* e il *button* per tornare alla prima scena ovvero quella appunto del menu principale. Anche qui è presente l'icona per il lancio del browser alla pagina del portale web. Lo script associato si occupa di partire dallo stato di menu chiuso per poi offrire le due funzioni di attivazione e disattivazione del pannello con relativo blocco del movimento (*Mouse Manager*).

5.5 Struttura grafica della scena House

La scena House contiene la maggior parte degli script C# dell'applicazione, presenta visivamente una villa con giardino la quale ospita i quadri da scaricare mettendo a disposizione delle zone dentro le quali istanziare le opere dette punti di spawn. La navigazione in prima persona permette il movimento tramite spot (checkpoint) visibili a terra mentre lo sguardo si può rivolgere liberamente in ogni direzione tramite trascinamento del mouse.

5.5.1 Gerarchia degli oggetti

Nel pannello *Hierarchy* della scena house l'elemento più corposo è quello dell'oggetto della casa (*Modern villa 2021*, Figura 5.7) esportato da Blender e composto da numerose *mesh*¹² [26] per ogni oggetto presente nella villa e nel giardino, dalle mura ai pavimenti. Se espanso mostra la sua natura complessa e strutturata, nella maggior parte delle situazioni di sviluppo può essere visualizzato come un oggetto unico rappresentando di fatto il *terreno di gioco*.

Oltre al modello importato sono visibili tutti gli oggetti presenti in scena (2D e 3D, finali o di debug come, ad esempio, quelli di tipo *DontDestroy* i quale vengono rimpiazzati dagli omonimi provenienti dalla prima scena), come mostrato in Figura 5.8.

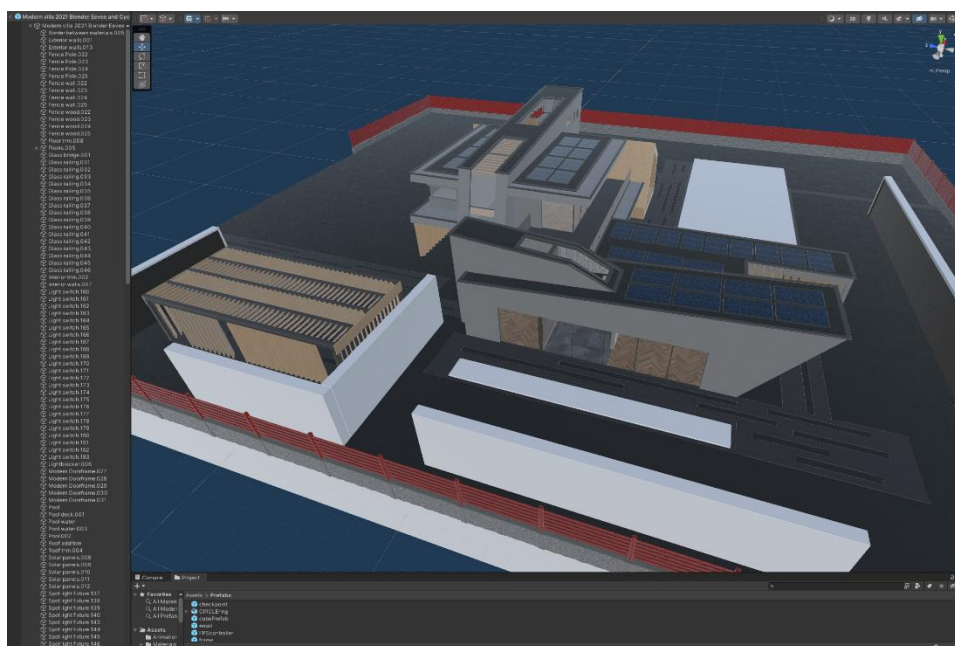


Figura 5.7: Modello della villa

¹²Struttura poligonale di un modello 3D, definisce i punti del modello nel sistema di riferimento [31]

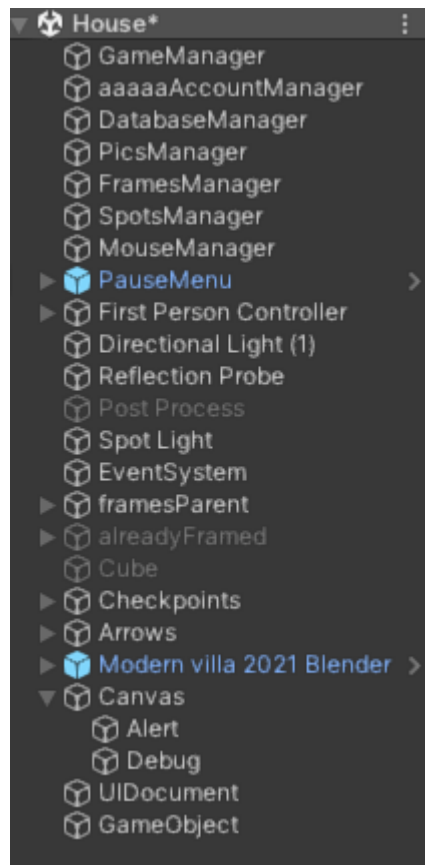


Figura 5.8: Gerarchia oggetti scena House

I *Manager* gestiscono delle macro-funzioni di programmazione e sono empty object, Game Manager e Account Manager hanno le analoghe funzioni della scena del menu. *Database Manager* è il primo punto di lavorazione dei quadri con il download dei dati dal server ma è *Pics Manager* il responsabile dell'istanziamento dei quadri e di tutte le operazioni sulla loro dimensione grazie alla comunicazione con il frame manager, il quale date le dimensioni di ogni quadro scaricato e dei punti di spawn disponibili assegna uno di tali oggetti al quadro richiesto.

Spots Manager conosce i luoghi puntati differenziando i punti di spawn dagli spot di navigazione operando sul materiale di tali oggetti a seguito della navigazione. Dialoga con *Mouse Manager*, il quale reagisce ai click e allo spostamento del mouse governando la navigazione, il movimento della camera e la visualizzazione dei dettagli del quadro.

L'oggetto *First Person Controller* è un *controller* offerto da un asset esterno molto semplice per la definizione dei parametri e script di controllo del personaggio e contiene la camera di gioco ad altezza occhi, una mesh disattivata per mantenere l'avatar invisibile e delle sorgenti audio per simulare la camminata [27].



Figura 5.9: First Person Controller

Oltre agli oggetti relativi all'illuminazione e gestione del bake grafico è presente la lista di punti di spawn (*frame*), ovvero le strutture in cui ogni elemento contiene il modello della cornice, la zona di intorno del quadro entro cui cliccare (gestita dal componente *Collider*) e con un ulteriore oggetto che indica la zona da raggiungere con la navigazione per la posizione privilegiata di visione. Per ogni quadro è presente una lista di elementi Canvas con le informazioni da visualizzare e il button per chiudere tale pannello, inoltre in posizione di visualizzazione una luce evidenzia la cornice al passaggio del mouse come feedback grafico per invitare l'utente a cliccare ed aprire il pannello informativo.



Figura 5.10: Gerarchia oggetti dei punti di spawn dei quadri

Per gli spot di navigazione è presente un ulteriore lista di oggetti (*Checkpoints*) composta dai soli modelli posizionati e visibili a terra, mentre dentro *Arrows* sono posizionati altri checkpoint non attivi per il passaggio alle aree non ancora disponibili (piano superiore).

Tramite un insieme Canvas sono infine disponibili dei campi di testo per la visualizzazione di messaggi a schermo di avviso.

5.5.2 Gestione della navigazione in prima persona

L'esperienza in Meta Gallery è in prima persona: la camera assume la posizione degli occhi del personaggio (invisibile) e i movimenti e lo spostamento dello sguardo simulano quello di una persona reale. Tramite il mouse viene controllato il proprio avatar virtuale, muovendo la camera tramite il trascinarsi durante un click ed effettuando le interazioni con un doppio click. L'utilizzo degli elementi di UI invece avviene con un solo click in quanto in quei momenti il movimento di camera e personaggio sono disabilitati.

Il mouse è quindi sempre visibile in quanto necessario per il puntamento degli oggetti con cui interagire e per effettuare le azioni.

La navigazione avviene quando puntando uno degli spot o delle zone limitrofe ad un quadro si effettua un doppio click, a quel punto l'utente viene trascinato al centro dello spot o davanti al quadro in posizione di visualizzazione.

Gli spot hanno un sistema di materiali per cui l'oggetto su cui si è posizionati diventa trasparente tramite una dissolvenza (*fade*), facendo invece un fade verso il colore quando lo si abbandona.

In posizione di visualizzazione del quadro invece viene attivata la possibilità di interagire con esso illuminandone l'intorno al passaggio del mouse (*focus*) e aprendo il pannello informativo dell'opera al doppio click sullo stesso.

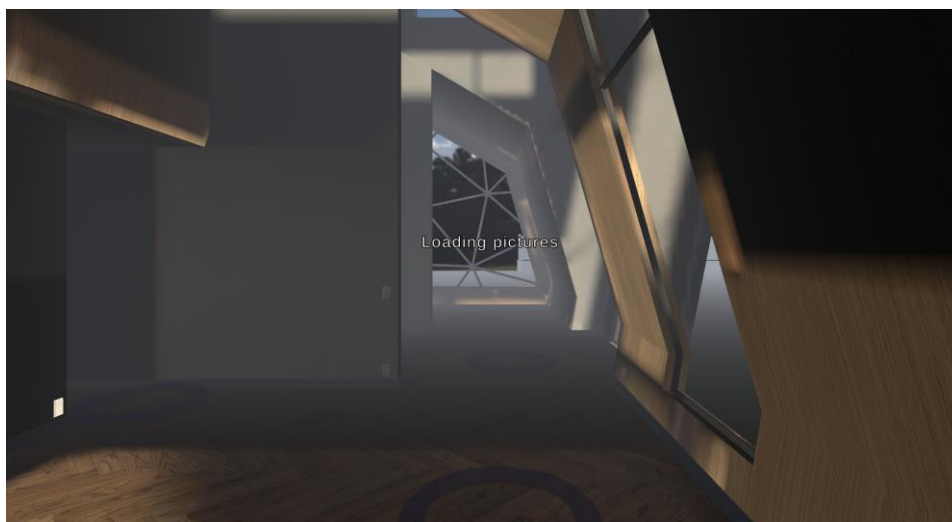


Figura 5.11: Inizio navigazione scena House

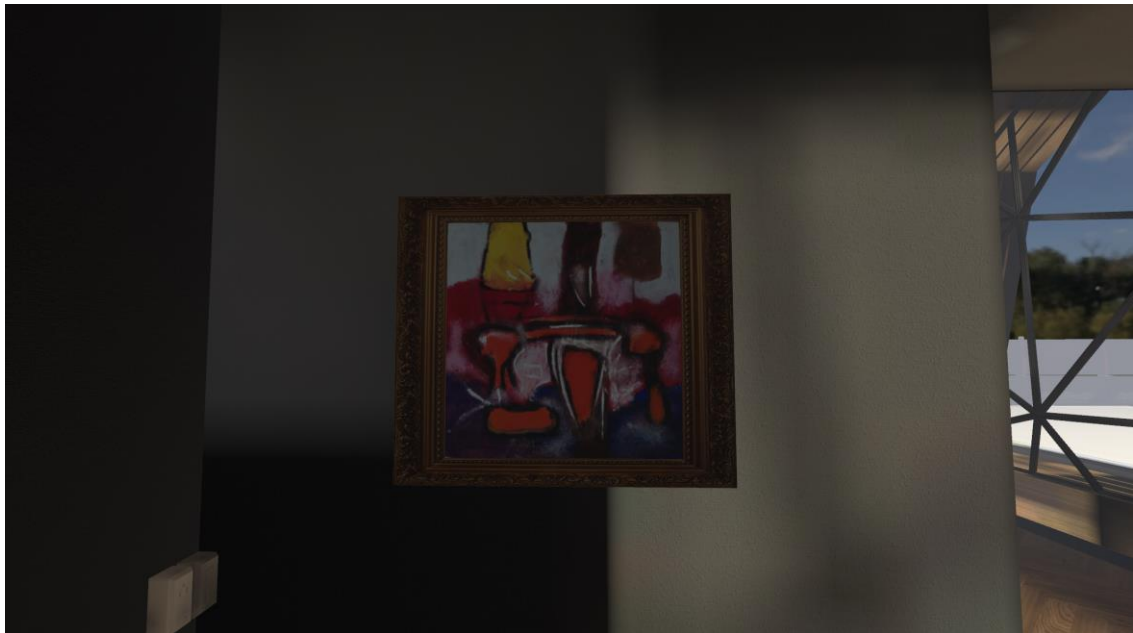
5.5.3 Feedback grafici e interfaccia 2D

La risposta grafica agli input dell'utente avviene tramite un testo da popolare visibile al centro del display con durata limitata, una volta chiamata la funzione di visualizzazione con la relativa stringa da scrivere a schermo dopo un determinato numero di secondi l'avviso scompare.

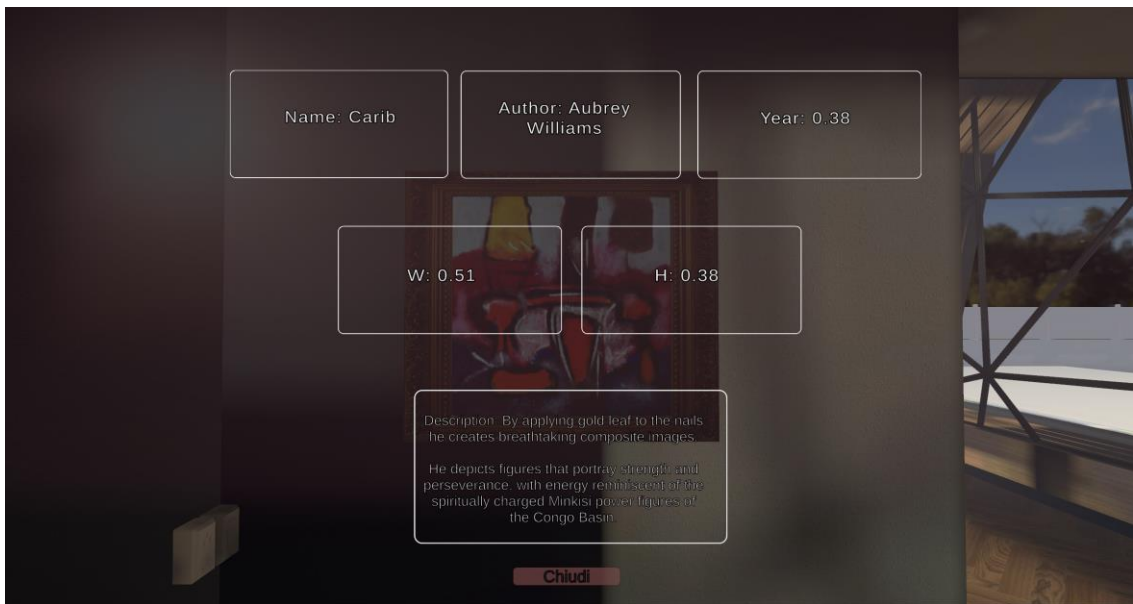
L'elemento *Alert* comunica principalmente il caricamento dei quadri ad inizio scena (in quel momento il sistema sta comunicando con il server, scaricando i quadri e posizionandoli, per poi avvisare l'utente in caso uno dei quadri dovesse essere fuori misura rispetto ai punti di spawn disponibili, in quel caso il quadro viene riscaldato in modo tale da poter essere istanziato).

In fase di osservazione di uno dei quadri (Figura 5.12), con il personaggio posizionato sull'oggetto *standUpPosition*, passando il cursore all'interno della cornice quest'ultima si illumina leggermente, attivando (e disattivando quando il mouse esce dalla zona di *focus*) una luce di tipo spot posizionata dietro al quadro e la cui intensità viene gestita tramite interfaccia del componente *Light* e regolata affinché possa essere leggera ma visibile, senza intaccare la visualizzazione dell'opera.

Con un doppio click da questo punto dell'esperienza, l'utente apre il pannello con le informazioni del quadro, ovvero un insieme di Canvas testuali popolati con le informazioni scaricate per ogni singolo quadro, ovvero il nome, l'autore, le dimensioni, l'anno e la descrizione dell'opera seguiti tutti da un tasto per chiudere tale interfaccia. Durante la visualizzazione delle informazioni la navigazione è bloccata.



(a)



(b)

Figura 5.12: Visualizzazione quadro, navigazione sull'oggetto standUpPosition

5.6 Features ed algoritmi della scena House

Sono numerosi gli script e in generale i componenti utilizzati per adempiere a tutti i compiti richiesti dalla scena (Diagramma 10), molti di essi interagiscono tra loro e si influenzano, i blocchi principali riguardano il dialogo con il server, la gestione dell'istanziamento dei quadri, la navigazione e le interazioni, oltre alle funzioni di game management analoghe a quelle previste per la prima scena.

Le principali funzioni offerte in scena permettono di disporre i quadri, navigare e interagire con l'interfaccia e con il mondo di gioco, come mostrato in Figura 5.13. A partire dai dati di login raccolti nella scena del menu principale (o dell'account amico selezionato per la visita) viene per prima cosa eseguito il flusso di download e disposizione dei quadri.

A fine allocazione può effettivamente partire l'esperienza utente tramite le funzioni di navigazione e di interazione con i punti di spawn ed i quadri per la loro visualizzazione. Il Game Manager è di nuovo il componente di gestione dello stato dell'esecuzione, il quale permette di interrompere la stessa per aprire il menu di pausa.

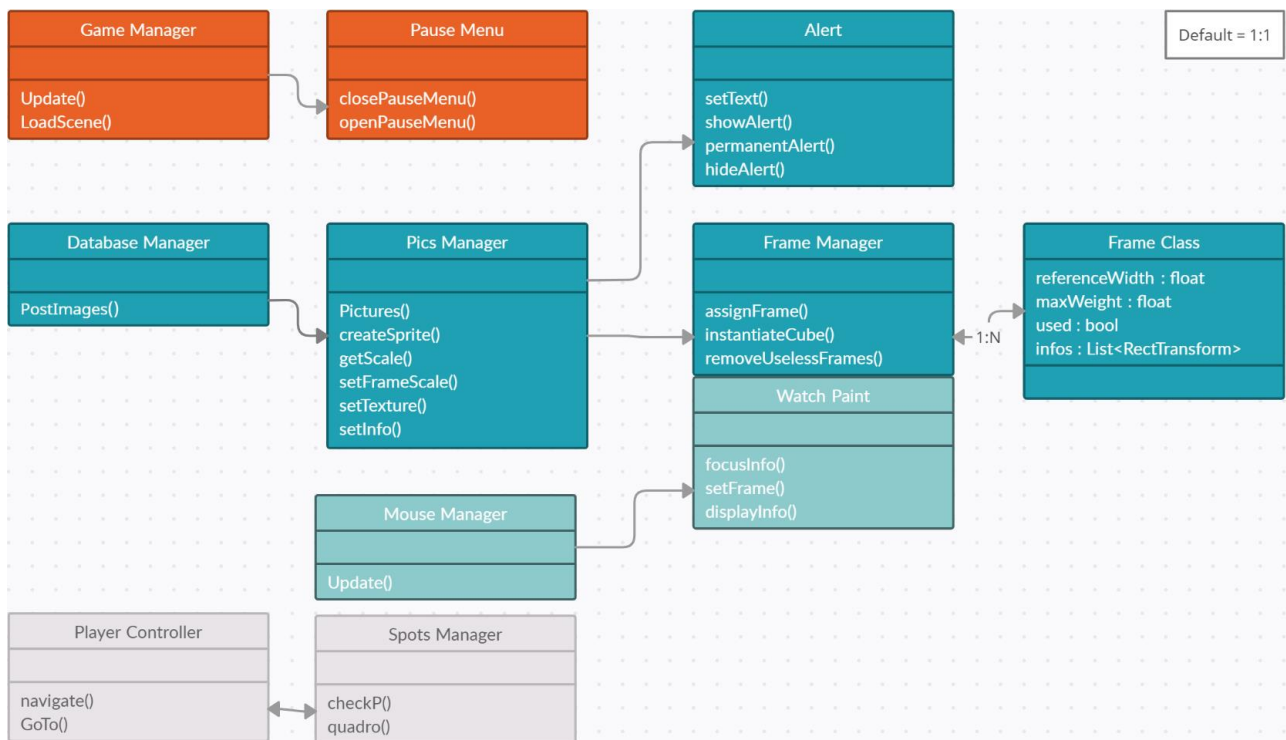


Diagramma 10: Diagramma delle classi scena House

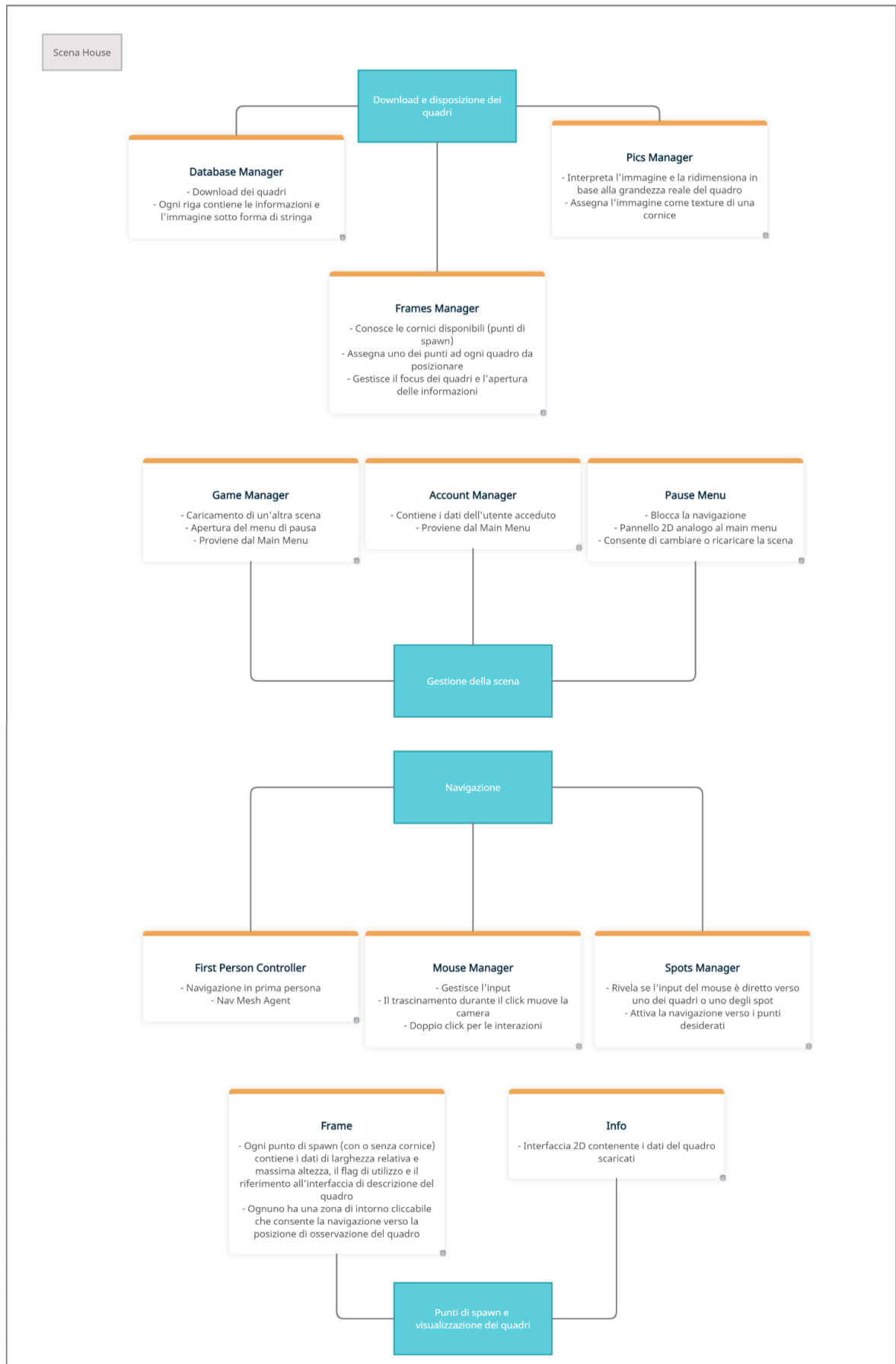


Figura 5.13: Architettura scena House

5.6.1 Database Manager per il download della lista dei quadri di un utente

In **Figura 5.14** viene mostrato il flusso di allocazione dei quadri.

Il manager del DB rappresenta il primo punto di contatto con il server per la lettura delle immagini da scaricare, primo compito da effettuare all'avvio della scena tramite la procedura delle richieste Unity utilizzate anche nel main menu, offerte dalla libreria `UnityEngine.Networking`.

Fino al termine del download dei quadri tramite lo script `postData.php` sul server su richiesta form di un determina ID (proveniente dal login nella scena del main menu) la navigazione non è permessa (mouse manager disattivato) lasciando l'utente di fronte all'avviso di caricamento in corso.

La gestione degli errori imposta la visualizzazione degli stessi sempre attraverso l'oggetto `Alert` il quale offre una funzione per la visualizzazione di avvisi temporanei ma anche la possibilità di mostrare messaggi fissi con relativa funzione di chiusura.

Se la comunicazione avviene con successo il `Mouse Manager` viene attivato e iniziano le operazioni di istanziazione dei quadri, infatti per ogni opera viene memorizzato un elemento in un vettore di stringhe e per ognuna di esse la chiamata al `Pics Manager` porta alle effettive operazioni di spawn, assegnazione e ridimensionamento.

Infine dalla lista di punti di spawn vengono rimossi quelli inutilizzati per evitare errori durante la navigazione (puntamento di aree vuote prive del quadro) tramite la funzione `removeUselessFrames()` anch'essa esterna.

5.6.2 Pics Manager per il posizionamento in scena dell'immagine su una cornice

La funzione chiamata esternamente per ogni quadro su cui operare è `Pictures()` la quale riceve come parametro la stringa scaricata nel formato offerto dal server e dirige le operazioni attraverso ulteriori funzioni in sequenza ognuna delle quali svolge un compito diverso atto alla creazione della corretta istanza. Attraverso un'operazione di `split` la stringa viene divisa nei vari campi dell'opera (divisi da uno spazio) popolando un vettore di stringhe chiamato `words` da cui estrarre subito le dimensioni di larghezza e altezza necessarie all'assegnazione di un punto di spawn (`frame`) e soprattutto l'informazione booleana sulla presenza della cornice già nella scansione (variabile `alreadyFramed`).

Viene a questo punto richiesto e memorizzato un oggetto `frame` al `Frames Manager` il quale date le dimensioni e l'informazione sulla cornice restituisce il game object su cui lavorare.

Ad ogni frame è associata la classe `FrameClass.cs` per descrivere dei valori relativi ad ogni singolo punto di spawn: oltre a memorizzare un flag che indica il fatto di essere già stato scelto, il parametro `referenceWidth` segna una larghezza in metri entro la quale il quadro possa rientrare. In fase di assegnazione viene scelto il frame disponibile più vicino in termini di larghezza senza fuoriuscire anche dall'altezza massima; se viene assegnato l'ultimo oggetto della lista dei punti di spawn (disposti in ordine crescente di `referenceWidth`) vuol dire che l'oggetto è da riscaldare perché troppo grande rispetto ai punti disponibili. In base alla larghezza relativa dell'oggetto assegnato dal `Frames Manager`, il `Pics Manager` controlla di essere nel caso di oggetto troppo grande riassegnando, se la condizione è verificata, delle dimensioni standard di 2x2 metri e richiamando la funzione per la richiesta di un punto di spawn.

Il `Pics Manager` opera con numerose funzioni sul frame assegnato con i dati scaricati dal database manager. Per prima cosa tramite `createSprite()` crea un'immagine (sprite) a partire dai dati testuali (binari) scaricati. L'immagine creata come texture 2D viene

posizionata su uno sprite per essere utilizzata correttamente ed assegnata a un materiale; in questa fase vengono memorizzate le dimensioni reali della scansione presente sul DB indipendentemente da quelle indicate nei campi *width* e *height*.

La funzione `getScale()` si occupa di estrarre due informazioni utilizzate successivamente: il ratio della texture (proporzione tra le due dimensioni) e le informazioni di scala nel mondo 3D dell'oggetto cornice per poter poi effettuare un ridimensionamento partendo dall'oggetto impostata con scala quadrata (larghezza = altezza).

In `setFrameScale()`, infatti, lo script cambia la scala della cornice sulla base della larghezza in metri inserita sul DB ma per l'altezza opera in base al ratio, così da evitare possibili errori grafici di stiramento dell'immagine presumendo un certo livello di approssimazione nella misura inserita dall'utente. Tramite questo procedimento le proporzioni delle dimensioni fisiche dell'immagine scaricata rimangono inalterate. A questo punto l'oggetto frame è pronto ad essere popolato con i dati corretti, mediante la funzione `setTexture()` viene assegnata la texture al materiale interno della cornice. In caso di cornice scansionata l'oggetto risulta essere un cubo da un solo materiale.

Infine per ogni campo di descrizione del quadro vengono modificati i Canvas del frame con le informazioni da visualizzare. Per ognuno vengono effettuati delle sostituzioni di carattere secondo la convenzione progettata con il server la quale evita l'utilizzo di spazi tra un attributo e l'altro.

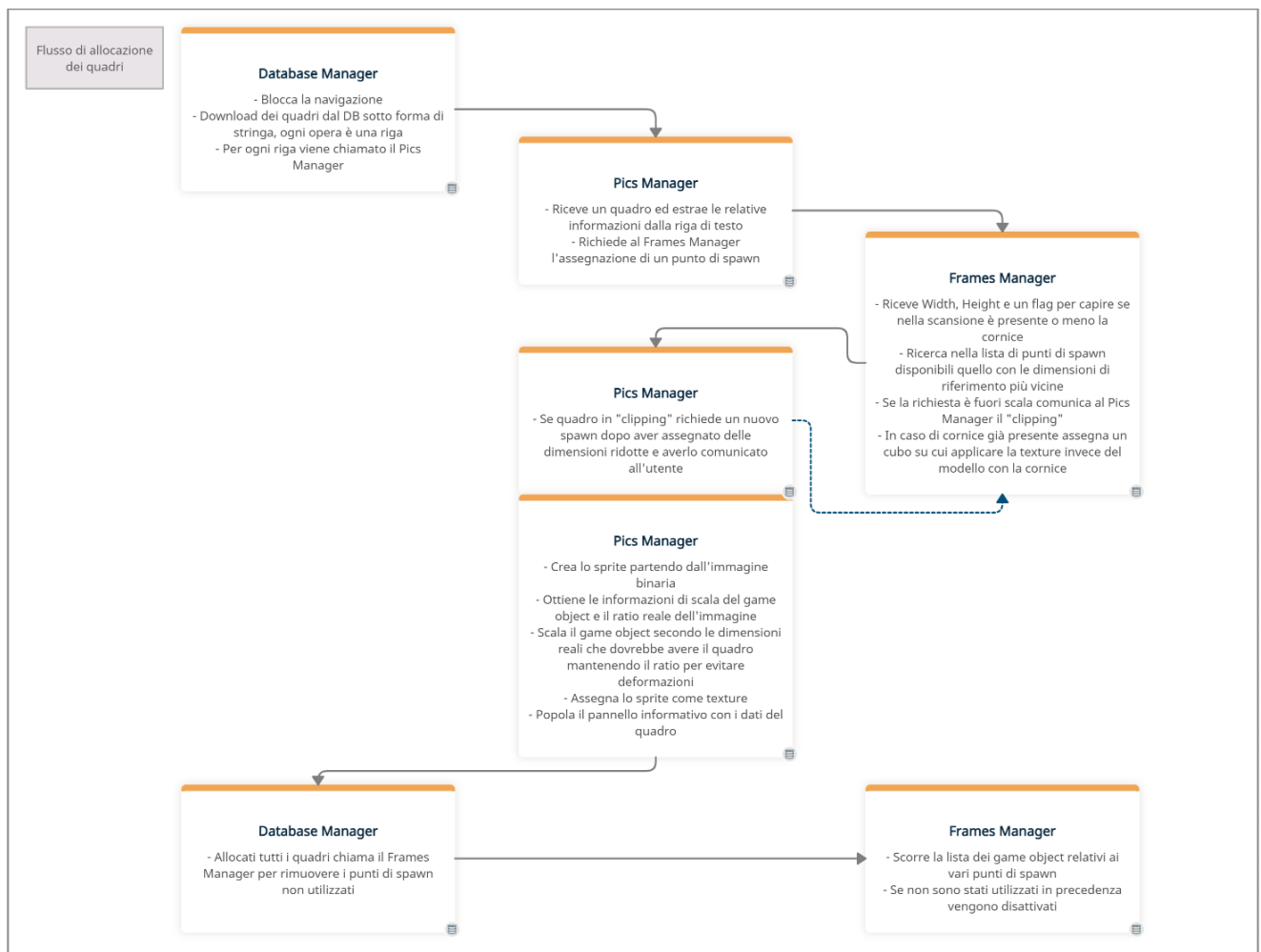


Figura 5.14: Flusso di allocazione quadri

5.6.3 Frames Manager per l'assegnazione di una cornice al quadro

Seguendo il flusso mostrato in **Figura 5.14**, il Frames Manager opera sulla lista di frame per assegnare una zona di spawn ad ogni quadro. L'assegnazione si basa sulla conoscenza della lista di oggetti presenti in gerarchia (**Figura 5.10**) disposti in ordine di larghezza crescente, rivelando, date delle dimensioni, l'elemento più vicino tramite operazioni matematiche di distanza. Qui avviene anche la creazione di un oggetto cubico invece del modello di cornice per le scansioni comprensive già di cornice, tramite i dati di posizione del frame scelto viene istanziato il cubo su cui applicare le texture nelle stesse coordinate e disposto in gerarchia nello stesso punto, permettendo la compatibilità con gli altri script.

Un ulteriore componente è presente in tale oggetto, ovvero **Watch Paint**, la cui funzione principale è la gestione dell'emissione della luce al passaggio del mouse sul quadro. Tramite le librerie di gestione della fisica di Unity è possibile, infatti, effettuare il *Raycast* ovvero la simulazione di un raggio proveniente da un'origine in una data direzione, nello specifico partendo dalla camera (ovvero gli occhi dell'utente che naviga in prima persona). Il principale utilizzo prevedere l'individuazione degli oggetti puntati con lo sguardo, è infatti possibile rilevare i collider intercettati da tale raggio. Se il raggio colpisce la cornice vuol dire che lo si sta puntando (focus) e tramite una funzione pubblica al momento della selezione dell'apertura del pannello informativo viene attivato il Canvas da visualizzare e bloccata la navigazione.

5.6.4 Puntamento, navigazione e Spots Manager

Alla base della navigazione c'è il **Mouse Manager** il quale controlla continuamente gli input del mouse chiamando la funzioni di spostamento della camera in caso di click singolo (fino al rilascio del tasto sinistro), mentre in occasione di un doppio click, quindi una successione di pressioni più breve di una frazione di tempo determinata, gestisce la navigazione o la visualizzazione delle informazioni di un quadro.

Il componente **Player Controller** si occupa della traslazione e della rotazione del personaggio, ovvero dell'oggetto detto FPS dotato di caratteristiche fisiche tra cui il peso. Al momento della chiamata del mouse manager utilizza il *Raycast* per rilevare possibili oggetti interagibili dove dirigere la navigazione, la distinzione tra le tipologie di collider spetta allo spots manager. Quest'ultimo tramite un'operazione di selezione (switch) in base al tag dell'oggetto puntato permette al controller di navigare verso tale posizione nel caso di game object significativo. La navigazione avviene attraverso la *NavMesh*, ovvero una struttura dati di intelligenza artificiale per stabilire nel mondo di gioco le superfici percorribili e gli ostacoli, automatizzando la navigazione di un oggetto detto *agent* da un punto all'altro.

Dopo la definizione di alcuni parametri di gestione dell'agent, come altezza, raggio e capacità di salto, la mappa di navigazione può essere costruita (bake) e utilizzata dal controller, infatti le chiamate al movimento specificano semplicemente il punto di arrivo desiderato.

Lo spots manager distingue il caso di spot o checkpoint da quello dei quadri. Nel primo caso prima di effettuare la navigazione permetterà l'animazione di fade in dell'oggetto sui cui si è posizionati in tale momento (quindi trasparente finché sotto il personaggio) nascondendo invece quello verso cui ci si dirige. Per i quadri, oltre al fade in se si proviene da uno spot e alla chiamata alla navigazione comunica al componente **Watch Paint** l'oggetto frame che si sta andando a visualizzare permettendogli di effettuare le proprie operazioni sul game object.

6 User Test

Il prototipo di Meta Gallery a fine sviluppo è stato sottoposto a dei test d'uso sia da parte dell'azienda Motion Pixel, sia da parte di utenti tipo compatibili con i target di riferimento. Quest'ultimi vengono riportati di seguito presentando 4 casi d'uso (Diagramma 11), uno per ogni utente tipo ricoprendo le diverse metodologie di uso del sistema.

Gli attori di maggiore interesse, i quali hanno provato l'app, sono persone appassionate d'arte e vicine a tale mondo anche a livello professionale, con l'eccezione di una dipendente comunale nel caso dei test di gestione del database.

Per ognuno è stato presentato un contesto da cui partire per operare ed eseguire dei compiti sull'applicativo simulativo, sul portale web o su entrambi.

L'osservazione delle reazioni, l'ascolto dei commenti, l'analisi del flusso di operazioni compiute dall'utente è di vitale importanza per lo sviluppo futuro dell'app e per trarre le conclusioni relative alla facilità d'uso. Ne derivano infatti spunti di vario tipo e conferme rispetto ad aspetti su cui sviluppare in futuro ma anche componenti già funzionanti così come presentati, oltre a delle idee per nuove feature.

Durante i test è stata concessa massima libertà agli utenti senza guidarli o interrompere il loro flusso di lavoro. Vengono poi riportati di seguito i comportamenti di maggiore interesse, le impressioni e i commenti in ordine cronologico.



Diagramma 11: Casi d'uso dei test

6.1 Gallerista

Il primo test riguarda un utilizzo generale dell'app in tutte le sue componenti, un flusso completo dalla registrazione all'esplorazione della galleria, adatto nel caso di un gallerista che si occupa personalmente anche della gestione del database di opere.

La persona incaricata di effettuare tale test è un ragazzo di 28 anni appassionato d'arte ma non professionista, i componenti della sua famiglia pur facendo altri lavori ospitano in casa delle mostre locali (provincia di Latina) in quanto possessori di una villa molto curata in termini di architettura. L'utente in questione ha acquistato delle opere in passato di seconda mano e recuperato alcune da conoscenti non più interessati o a seguito di ristrutturazioni (un paio di quadri gli sono stati donati da una chiesa a seguito della modernizzazione dell'infrastruttura).

L'utente che verrà da qui etichettato come *gallerista* ha avuto un paio di esperienze di vendita informale a seguito di conoscenze effettuate durante le mostre ospitate.

Il test viene presentato dopo una spiegazione generica sull'app, ovvero un sistema di gestione della propria collezione, con la spiegazione delle varie feature senza mostrare come accedervi tramite interfaccia. Viene descritto lo scopo del sistema mostrandone l'utilità a cui il gallerista appare subito molto interessato.

Prima della prova sono state fornite dalla sua famiglia le foto della collezione presente in casa, sono state quindi ritagliate e consegnate all'utente per poterle caricare. Infatti, il test consiste nell'upload di 10 quadri di sua proprietà e quindi da lui conosciuti e dal successivo utilizzo di tutte le feature presenti sul portale web. È stata consegnata inoltre l'email di un account da invitare, mentre quella del gallerista è stata inserita nelle amicizie di un altro utente presente sul DB.

Successivamente all'utilizzo del portale web tramite il link alla build WebGL, l'utente naviga attraverso la propria collezione utilizzando l'app simulativa.

Qui le note di maggiore interesse di tale User Test partendo dal portale web:

- Dopo la spiegazione iniziale: “Sembra interessante anche in tema multiverso. In generale utile come biglietto da visita.”
- Creazione profilo: nota alcune mancanze in termini di feedback e tutorial, fa alcune domande per capire come procedere ma nota poi la necessità di creare un account, si aspetta un'email di conferma per poi capire di poter già fare il login.
- Interfaccia povera: prova login/logout oltre alla chiusura e riapertura del browser, trova scomodo il ritorno alla homepage dopo il login richiesto per alcune operazioni.
- A questo punto ha capito l'utilizzo del feedback nella parte superiore delle pagine. Invita, elimina l'amicizia e reinvia l'email indicata con facilità.
- L'upload procede senza problemi, trova un po' scomoda la casella per la descrizione, si aspetterebbe uno spazio più grande. Una delle descrizioni inoltre risulta troppo lunga causando l'insuccesso dell'upload, il gallerista dopo tre tentativi comprende che il motivo sia quello.
- Elimina alcune immagini dalla lista con semplicità e il caricamento dell'elenco è un po' lento in un paio di occasioni.

Da qui il gallerista si sposta sull'app WebGL:

- Va abbastanza spedito sul login per poi visitare subito l'account amico
- Nel suo caso il caricamento è un po' lento per via del web host un po' instabile in alcuni orari del giorno, segnala poi come preferirebbe un loading grafico animato siccome in quei frangenti non sa bene cosa dover fare.

- Nessun problema con il sistema di navigazione, l'utente è abbastanza abile con mouse e tastiera.
- Si muove velocemente nell'ambiente e interagisce con un quadro senza prestare troppa attenzione, ritorna al menu con facilità e prova il proprio profilo.
- Cerca per un po' di tempo un quadro specifico, vorrebbe infatti una mappa da premere con il tasto M della tastiera.
- Riguardo la visualizzazione del quadro: "...è ok, si vede che è in sviluppo ma c'è tutto quello che ho inserito".
- Chiede se c'è un modo di tornare al portale web dall'app, dopo poco trova l'icona relativa e cerca una analoga sul sito per fare il percorso contrario ma non la trova.

Alla fine, risulta come abbia provato tutti i componenti ed esprime le seguenti conclusioni:

- Ci sono tutte le funzionalità descritte, alcune in versione completa mentre di altre c'è solo lo scheletro privo di tutto il design necessario per l'esperienza utente (principalmente per quanto riguarda le pagine web).
- L'interfaccia web si vede essere una bozza, mentre quella del software simulativo funziona nel menu e può essere migliorata facilmente nella GUI di visualizzazione del quadro. Le opere si presentano bene, rispettando le dimensioni reali da lui conosciute.
- In generale ottimo potenziale, non essendo un professionista per lui è importante lo sviluppo futuro in termini di transizioni e graziosità del tutto per rendere il sistema più appetibile.
- A lui piace avere la gestione totale della propria collezione essendo in un contesto piccolo e familiare, il progetto è ottimo in tal senso ma preferirebbe un minor distacco tra app e web.

6.2 Contabile

Il test in questione riguarda la sola gestione del database, ha quindi uno scopo più commerciale rivolgendosi a chi opera a livello burocratico e contabile per conto di grandi galleristi o enti museali. La persona incaricata di eseguirlo è una ragazza di 35 anni, la quale lavora nel comune di Praia a Mare con incarichi di gestione di database anche se non relativi ai beni culturali o all'arte in generale.

La sua professione la mette quotidianamente a contatto con software gestionali la cui interfaccia viene messa in secondo piano rispetto alle performance ed il test in questione è di grande importanza per rilevare quanto il sistema di discosti dagli standard lavorativi in termini di aiuto, vincoli imposti all'inserimento e controlli di sicurezza proteggendo da bug e ambiguità.

Nella prova in questione viene impostato un contesto di primo utilizzo del portale web per conto di un gallerista; partendo dalla registrazione viene chiesto alla persona incaricata (identificata come *contabile*) di inserire una serie di 10 immagini da una cartella per creare una collezione operando su di esse, oltre alla gestione degli inviti rispetto ad un altro account di prova disposto per tali feature.

Di seguito le principali note emerse:

- La registrazione e il login funzionano bene
- Effettua l'upload di tutte le opere segnalando alcuni dubbi in caso di caratteri speciali, inoltre in un paio di casi le immagini non sono supportate per via di un formato errato, la contabile nota la cosa senza particolari problemi rinunciando al caricamento di esse.
- La gestione delle amicizie e delle immagini procede con fluidità.
- Per l'eliminazione delle immagini si aspetterebbe dei tasti o dei menu accanto agli elementi, così come per le amicizie.

Impressioni finali:

- “L'interfaccia non è troppo peggiore di quella dei programmi istituzionali o degli enti pubblici ma è comunque chiaramente da sviluppare, serve ad esempio meno libertà e più menu a tendina/modi per velocizzare soprattutto l'upload così come l'eliminazione e la modifica del contenuto.
Manca infatti la modifica dei quadri se non erro: ad ora devo eliminare e ricaricare il quadro per cambiare delle informazioni.”
- “In generale c'è un po' tutto quello che serve alla gestione di un DB del genere, inoltre come velocità è ok ma si può fare meglio in alcuni casi, soprattutto nella parte della lista delle immagini.”

6.3 Proprietario galleria

Un ulteriore test è stato pensato per un grosso gallerista la cui gestione contabile è assegnata ad altre figure professionali, il suo scopo è la sola navigazione attraverso l'applicativo simulativo.

La persona incaricata in questo caso è un appassionato di arte di 42 anni il quale ha recentemente visitato la October Gallery di Londra, viene sfruttato questo fatto per ricreare lo stesso ambiente tramite le opere realmente presenti nella galleria e dall'utente (*gallerista*) conosciute, simulando la proprietà di esse da parte dello stesso utente che quindi naviga attraverso la propria collezione.

Qui i momenti salienti:

- Date le credenziali accede semplicemente, nota che il tasto Exit è da sviluppare così come i preset grafici delle opzioni, preme Start.
- Il caricamento è veloce, finisce prima che lui possa provare qualche input. Inizia a muovere la camera per poi entrare nella casa.
- La navigazione funziona bene (“come in *Street View*”), propone una diminuzione della sensibilità di movimento della camera.
- Le dimensioni dei quadri sono realistiche rispetto alla dimensione che ricorda dalla visita.
- Le descrizioni si vede siano provvisorie e non ufficiali, infatti sono più complete e differenti quelle presenti alla vera galleria.
- Nota come serva una legenda/mappa e la disposizione chiaramente è diversa in termini di ordine delle opere dopo l'entrata nella casa.

Conclusioni:

- “Capisco la potenzialità, è interessante e rende vivo qualcosa che stavo un po' dimenticando.”
- “Pensare che sia io a possedere queste cose in remoto mi sembra molto utile. Preferirei un ambiente più da museo o forse addirittura un'unica stanza tipo corridoio con un'architettura classica.”

6.4 Possibile acquirente

L'ultimo test crea il contesto di un possibile acquirente interessato alla collezione di un gallerista, tramite invito già ottenuto (vengono fornite le credenziali e associato il suo account a quello dell'utente da visitare per visitare quindi l'ambiente tramite l'applicativo WebGL).

La persona scelta (*acquirente*) è un appassionato nonché possessore di NFT, lavora nel campo informatico e della modellazione 3D in provincia di Latina, studiando ingegneria informatica a Roma (magistrale). Non è infatti la prima volta che si trova a muoversi/creare ambienti virtuali per la presentazione di opere e prodotti legati anche alla vendita oltre che alla sola contemplazione.

Qui i momenti degni di nota e le sue impressioni:

- L'acquirente mostra una certa fluidità e capacità di muoversi all'interno dell'applicativo.
- Effettua il login, vede la sezione *Friends* ma vuole prima premere *Start*, nota che non c'è nessuna opera nel suo account. Ha già provato la navigazione, utilizzando da prima la tastiera e capendo dopo pochi secondi di necessitare del solo mouse per muoversi.
- Torna al menu e va sulla lista di amici e ne sceglie uno (ha 3 opzioni).
- Si muove agilmente con il mouse notando una sensibilità forse un po' elevata (varia leggermente in base allo schermo e al browser utilizzato).
- Nessun problema con la navigazione e l'interazione con i quadri, prova a "stressare" un po' il sistema, chiudendo e riaprendo i pannelli, tornando sui propri passi e navigando attraverso tutti gli spot premendo convulsamente col mouse per stimolare continuamente l'applicativo.

Conclusioni:

- Nota subito come l'interfaccia di visualizzazione dei quadri sia provvisoria, probabilmente la prima cosa da migliorare, oltre all'utilizzo di più ambienti magari più adatti ad una galleria ("architetture antiche stile castello") e come ci sia da aggiungere una musica contemplativa di accompagnamento.
- "...c'è però tutto quello che mi aspetto di vedere in un prodotto del genere e in realtà a livello grafico è già migliore di molti prodotti 'metaversici' presenti sul mercato, infatti la cosa che conta è che sia il prodotto (il quadro nel vostro caso) ad essere perfetto."
- "Aggiungerei la possibilità nella visualizzazione di vedere l'immagine in 2D ferma ma comunque è essenziale la parte simulativa, è molto apprezzato negli ambienti che ruotano attorno al metaverso."
- "Esteticamente è molto realistico e con le giuste accortezze diventa perfetto per quella grammatica 'Internettiana' che va di moda ultimamente, sicuramente meglio di alcuni minigiochi o ambienti di scarsa utilità creati solo per ospitare i prodotti e far sembrare reale qualcosa che non lo è, in questo caso diciamo che è la realtà ciò che avete da far vedere."

7 Conclusioni e lavori futuri

Diversi sono gli aspetti lavorativi e di progettazione delicati messi in evidenza durante le principali fasi di sviluppo. Come da richiesta, lo stato attuale di realizzazione presenta un prototipo di applicazione con la presenza stabile delle funzionalità principali, ovvero una struttura (tra portale e client simulativo) modulabile, la cui base non necessita di modifiche: nel dettaglio non hanno bisogno di interventi (anche per eventuale ampliamento o potenziamento) il sistema di login, gli script di modifica del contenuto del DB, gli algoritmi di disposizione e ridimensionamento dei quadri e il sistema di navigazione, puntamento e interazione.

Il passo successivo maggiore rispetto a quanto progettato con Motion Pixel sarà trovare un ambiente produttivo con richieste specifiche riguardo alle possibili feature di certificazione del possesso delle opere, contatto tra venditori e acquirenti, implementazione degli NFT, ideazione di una valuta e possibilità di effettuare degli acquisti interni.

In generale il destino dell'app, in base ai requisiti futuri, seguirà una strada probabilmente intermedia tra quella di essere un ambiente esplorativo e quella di essere uno spazio virtuale completo multiutente e con una propria economia. Un altro aspetto derivante da queste considerazioni è inoltre quello delle piattaforme di distribuzione, con la forte possibilità di prevedere uno sviluppo per VR, soprattutto negli ambiti museali in cui è molto apprezzato e nei casi di forte puntamento sul metaverso (così come anche delle app dedicate per smartphone).

Il progetto si presenta come una versione di prova del sistema finale, funzionante ma limitato in termini di funzionalità secondarie, variazione di contesti e abbellimento grafico. Per quanto riguarda il portale web è sicuramente necessario un lavoro di design delle pagine, lo sviluppo si è infatti concentrato nella realizzazione dei codici con la presenza di un'interfaccia provvisoria, il cui cambiamento potrà essere effettuato senza intaccare il funzionamento delle altre componenti.

A livello algoritmico il dialogo con il database e la gestione della sessione dell'utente è pronta ad essere migliorata in termini di ottimizzazione delle query (tramite accorgimenti di ordine e limitazione delle ricerche in caso di un solo risultato possibile nelle selezioni) e di sicurezza: è stato predisposto un token da utilizzare per identificare gli account attraverso una chiave indipendente dagli identificativi primari della tabella. Il sistema è ancora sprovvisto di una gestione dei cookies, informative sulla privacy e configurazione degli attestati e delle certificazioni per il caricamento delle pagine in sicurezza, permettendo la piena compatibilità con i browser.

Anche il servizio di hosting web è adatto ad un'applicazione dimostrativa ma necessita di utilizzare i piani a pagamento per un miglioramento delle performance e il caricamento di una build non eccessivamente compressa in fase di compilazione. Un trasferimento di servizio non è escluso, specialmente per la definizione di un dominio migliore a livello di marketing.

L'app WebGL permette l'esplorazione di un solo luogo ma è previsto per il futuro l'utilizzo di differenti ambienti per permettere ai visitatori (o ai collezionisti nel caso volessero forzare l'esplorazione tramite un'ambientazione precisa) di scegliere, gli algoritmi sono indipendenti da ciò e possono essere eseguiti su scene differenti se queste risultano organizzate secondo le regole interne progettate, come ad esempio l'ordine dei game object in gerarchia.

Un elemento importante per lo sviluppo futuro è fornire la possibilità di visitare un utente senza la necessità di essere registrato, sfruttare quindi link e codici di invito nell'ottica di

proporre le proprie gallerie veramente sotto forma di biglietti da visita virtuali, rivolte a persone che verosimilmente non sono registrate sulla piattaforma e non hanno motivo per farlo.

Un errore di procedimento durante la fase di testing è da individuare invece nel flusso di lavoro, in particolar modo nella scelta dell'ordine con cui provare i vari aspetti del sistema.

Come già descritto in parte durante l'analisi delle decisioni relative a quale DB utilizzare, il lavoro di progettazione con tutte le prove è proceduto in parallelo per quanto riguarda tutte le componenti, le quali però si influenzano a vicenda. Le decisioni relative alla natura web dell'applicazione e riguardanti le caratteristiche del database sono quelle che hanno portato alla determinazione del maggior numero di vincoli.

Chiudendo prima il cerchio riguardo le librerie WebGL si sarebbe virato direttamente su Unity evitando i test tramite Unreal Engine; inoltre, avrebbe reso definitivamente non percorribile la strada attraverso Firebase come sistema per l'archiviazione dei dati. Una lunga fase di studio della piattaforma è stata invece necessaria prima di accantonare il percorso tracciato attraverso i sistemi di Amazon AWS, risultati poi non propriamente adatti alla semplice struttura relazionale prevista per l'applicazione.

Oltre agli obiettivi già prefissati in fase di sviluppo molti dettagli sono emersi dagli User Test, dettando una strada più precisa soprattutto per la creazione delle interfacce. Quella del portale web, infatti, dovrà essere strutturata e guidata con vincoli ed un utilizzo intelligente delle liste, mentre per l'applicativo le prime migliorie dovranno consistere nella scelta di un ambiente di navigazione più consono al mondo dell'arte e soprattutto nella realizzazione di una GUI per la visualizzazione dei quadri riconoscibile, intuitiva e gradevole, prendendo come riferimento ciò che era stato creato nel prototipo visuale presente su 3DVista.

Oltre ad aver dato spunti in termini di macro-funzioni o dettagli, hanno quindi definito meglio le priorità delle modifiche da effettuare.

Le impressioni degli utenti sono anche importanti per il marketing e il processo di distribuzione/vendita dell'app, come creare un pitch document in grado di risultare interessante ed utile per i target di riferimento. Il futuro del progetto non può infatti non passare attraverso attori in grado di finanziare la produzione definitiva del sistema, strutturando un team di sviluppo per la gestione parallela di design, back-end informatico, struttura grafica 3D e una strategia di marketing, distribuzione e pubblicizzazione appropriata e mirata per coinvolgere il giusto target.

Bibliografia

- [1] «Unity 3D,» [Online].
Available: <https://unity.com/>.
- [2] «Motion Pixel SRL - editing video making,» [Online].
Available: <https://motionpixel.it/>.
- [3] «3DVista - Software professionale per tour virtuali,» [Online].
Available: <https://www.3dvista.com/en/>.
- [4] M. P. SRL, «Prototipo Meta Gallery,» [Online].
Available: <http://www.360vrexperience.it/metagallery/>.
- [5] denniswoo1993, «Modern Villa 2021 Blender Eevee And Cycles 3,» Sketchfab, [Online].
Available: sketchfab.com/3d-models/modern-villa-2021-blender-eevee-and-cycles-3-08e0bcb1f556419f8f386d2e15b115c1.
- [6] «blender.org - Home of the Blender project,» [Online].
Available: <https://www.blender.org/>.
- [7] «ArtSteps - Make your own VR Exhibitions,» [Online].
Available: <https://www.artsteps.com/>.
- [8] KUNSTMATRIX, «art.spaces,» [Online].
Available: <https://artspaces.kunstmatrix.com/>.
- [9] Kunstmatrix, «ART.AUGMENTED,» [Online].
Available: <https://augmented.kunstmatrix.com/>.
- [10] «SPACE - tryspace.com,» [Online].
Available: <https://app.tryspace.com/>.
- [11] S. Metaverse, «Space - Facebook,» [Online].
Available: <https://www.facebook.com/SpaceMetaverse/>.
- [12] «room - Enterprise Metaverse Solutions,» [Online].
Available: <https://www.room.com/>.
- [13] Room, «3D Visualization made easy,» [Online].
Available: <https://www.room.com/3d-view#c6550>.
- [14] A. -. Docs, «Amazon AWS - S3,» [Online]. Available:
https://docs.aws.amazon.com/it_it/AmazonS3/latest/userguide/Welcome.html.
- [15] Oracle, «What is a relational database,» [Online].
Available: <https://www.oracle.com/it/database/what-is-a-relational-database/>.
- [16] Google, «Firebase,» [Online].
Available: <https://firebase.google.com/>.

- [17] frwiki, «Aliasing,» [Online].
Available: <https://it.frwiki.wiki/wiki/Cr%C3%A9nelage>.
- [18] E. Games, «Unreal Engine,» [Online].
Available: <https://www.unrealengine.com/en-US>.
- [19] ufna, «VaRest,» [Online].
Available: <https://github.com/ufna/VaRest>.
- [20] UnrealEngineHTML5, «Documentation for the Community-supported HTML5 Platform Extension,» [Online]. Available: <https://github.com/UnrealEngineHTML5/Documentation>.
- [21] «SQLite,» [Online].
Available: <https://www.sqlite.org/index.html>.
- [22] «UnityWebRequest - Scripting API,» [Online].
Available: <https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html>.
- [23] «000WebHost,» [Online].
Available: <https://it.000webhost.com/>.
- [24] «hostinger.it - Piani di hosting,» [Online].
Available: https://www.hostinger.it/special/000webhost?utm_source=000webhost&utm_medium=frontend&utm_campaign=631958425.1665656706.
- [25] G. Bissonnette, «Primo - Minimalist Main Menu Pack,» [Online]. Available:
<https://assetstore.unity.com/packages/p/primo-minimalist-main-menu-pack-226253>.
- [26] U. Documentation, «Importing Objects From Blender,» [Online].
Available: <https://docs.unity3d.com/560/Documentation/Manual/HOWTO-ImportObjectBlender.html>.
- [27] S. Pasi, «Mini First Person Controller,» [Online].
Available: <https://assetstore.unity.com/packages/tools/input-management/mini-first-person-controller-174710>.
- [28] «svdcv,» [Online].
Available: www.google.it.
- [29] U. Documentation, «Coroutines,» [Online].
Available: <https://docs.unity3d.com/Manual/Coroutines.html>.
- [30] develop4fun.it, «Coroutine,» [Online].
Available: <https://www.develop4fun.it/unity-programmare-in-csharp-coroutine/>.
- [31] J. Petty, «What is a Polygon Mesh,» [Online].
Available: <https://conceptartempire.com/polygon-mesh/>.
- [32] I. E. | D. N. T. R. Linda Tucci, «What is Metaverse? An explanation and in-depth guide,» [Online].
Available: <https://www.techtarget.com/whatis/feature/The-metaverse-explained-Everything-you-need-to-know>.