

# **DOCUMENTAZIONE MAPOCALIPSE**

**By  
Ioan Robert Gruia**

<b>Cos'è Mapocalipse?.....</b>	<b>3</b>
<b>Backend.....</b>	<b>3</b>
Framework.....	3
Flusso di Lavoro.....	4
Componenti chiave.....	4
Vantaggi.....	4
Librerie.....	5
1. mysqlclient.....	5
2. geopy.....	5
3. channels.....	5
4. unicorn[standard].....	6
5. whitenoise.....	6
Struttura del Progetto.....	6
- Applicazione Mapocalipse.....	6
- Altre Applicazioni.....	7
- Authentication.....	7
- Game.....	7
- Singleplayer.....	7
- Multiplayer.....	7
<b>Frontend.....</b>	<b>7</b>
Pagine.....	8
- Login/Register.....	8
- GameMode Choice.....	9
- SinglePlayer Page.....	10
- MultiPlayer Page.....	10
- Game Page.....	12
Librerie.....	13
1. Three.js.....	13
2. GSAP(GreenSock Animation Platform).....	13
API.....	14
- Google Maps JS API.....	14
<b>Avvio del server.....</b>	<b>16</b>
<b>Idee da aggiungere in futuro.....</b>	<b>16</b>

# Cos'è Mapocalipse?

(DISCLAIMER: Il nome del progetto è stato consigliato caldamente da ChatGPT, perché io non so dare i nomi ai progetti. Se a te che leggi non piace puoi porgere le tue lamentele verso ChatGPT)

Il progetto è un web game multiplayer e singleplayer e l'obiettivo è indovinare all'incirca dove sei sulla mappa del mondo basandosi su una street view. Si passa da centri cittadini fino anche a posti più sperduti sul pianeta, come ad esempio isole in mezzo al mare.(e sì, ci sono persone in grado di indovinare la posizione in quest'ultime).

La maggior parte della logica del gioco avviene nella parte frontend della webapp. Invece nella parte backend avviene tutta la memorizzazione dei dati sul database e la gestione della comunicazione in tempo reale tra gli utenti.

## Backend

La parte backend è sviluppata utilizzando il linguaggio di programmazione python e alcuni suoi framework e librerie.

## Framework

Il framework utilizzato per lo sviluppo backend di questo progetto è django, un framework web progettato per facilitare lo sviluppo rapido e il mantenimento di applicazioni web, tutto ciò mantenendo un design pulito e rendendo il codice facilmente riutilizzabile.

Django segue l'architettura MVT (Model-View-Template), molto simile al MVC ma leggermente diversa:

- **Model (Modello):** Rappresenta la logica dei dati e l'interazione con il database, definendo la struttura del database utilizzando classi Python.
- **View (Vista):** Contiene la logica dell'applicazione. Le viste sono funzioni o classi che elaborano le richieste HTTP, interagiscono con i modelli per recuperare dati e utilizzano i template per generare risposte HTTP.
- **Template (Template):** Gestisce la presentazione dei dati. I template sono file HTML che utilizzano un linguaggio di template per incorporare i dati dinamici.

## Flusso di Lavoro

1. **Richiesta:** Quando un utente effettua una richiesta HTTP, Django utilizza il suo sistema di routing URL per determinare quale vista dovrebbe gestire la richiesta. Le URL sono mappate a specifiche viste nel file `urls.py`.
2. **Vista:** La vista associata alla richiesta elabora i dati necessari, interagendo con il modello per recuperare o manipolare i dati.
3. **Modello:** Il modello interagisce con il database per ottenere o salvare i dati, utilizzando l'ORM di Django per astrarre le operazioni del database.
4. **Template:** La vista passa i dati necessari al template, che rende il contenuto HTML con i dati dinamici forniti dalla vista.
5. **Risposta:** Il template generato viene restituito al browser dell'utente come risposta HTTP, completando il ciclo.

## Componenti chiave

1. **Routing delle URL:** Il file `urls.py` definisce le mappature tra URL e viste, utilizzando il modulo `path` per mappare gli URL alle funzioni delle viste.
2. **ORM (Object-Relational Mapping):** Permette di interagire con il database usando oggetti Python, supportando operazioni CRUD senza scrivere query SQL.
3. **Sistema di Template:** Progettato per separare la logica di presentazione dalla logica di business, utilizzando tag e filtri per manipolare i dati all'interno dei template.
4. **Admin Interface:** Fornisce un'interfaccia di amministrazione pre-costruita che permette di gestire i modelli e il contenuto del sito(da utilizzare solo se la si sa utilizzare, può creare danni al server modificare i modelli)
5. **Gestione delle Applicazioni:** Un progetto Django può ospitare diverse applicazioni, ciascuna con funzionalità specifiche, che possono essere facilmente integrate nel progetto principale.
6. **Sistema di Sessioni:** Include un sistema di sessioni integrato che permette di memorizzare e gestire dati specifici dell'utente tra diverse richieste.
7. **Protezione CSRF:** Utilizza token CSRF (Cross-Site Request Forgery) per proteggere le applicazioni da attacchi CSRF, assicurando che le richieste provengano da utenti autenticati.
8. **Gestione dei File Statici:** Django offre un sistema per gestire i file statici (CSS, JavaScript, immagini) in modo efficiente(per quanto quasi sempre non intuitivo, soprattutto in deployment). Permette di definire le directory dove risiedono i file statici e include comandi per raccogliere e distribuire questi file durante il processo di deployment.

## Vantaggi

- **Rapid Development:** Permette di sviluppare applicazioni web rapidamente grazie alle numerose funzionalità integrate.
- **Sicurezza:** Include molteplici misure di sicurezza per proteggere le applicazioni da attacchi comuni.

- **Scalabilità:** Progettato per gestire progetti di qualsiasi dimensione, da piccole applicazioni a grandi siti web.
- **Flessibilità:** Altamente estensibile e può essere personalizzato secondo le esigenze specifiche del progetto.

In conclusione django è un potente framework che permette allo sviluppatore di concentrarsi sullo sviluppo delle funzionalità del server senza doversi preoccupare dei dettagli di implementazione di più basso livello(es. gestione delle sessioni, protezione CSRF, controllo sull'autenticazione).

## Librerie

Oltre al framework django, per lo sviluppo di Mapocalipse sono state utilizzate diverse librerie:

### 1. mysqlclient

- **Descrizione:** mysqlclient è un'interfaccia per MySQL scritta in Python. È un driver che permette di connettere un'applicazione Python a un database MySQL.
- **Funzionalità:** Offre metodi per eseguire query SQL, gestire connessioni al database e manipolare i risultati delle query. È compatibile con l'ORM di Django e altre librerie di database Python.
- **Utilizzo:** Comunemente utilizzato per interagire con database MySQL in applicazioni web e software che richiedono operazioni di database.

### 2. geopy

- **Descrizione:** geopy è una libreria Python per la geocodifica, ovvero la conversione di indirizzi in coordinate geografiche e viceversa.
- **Funzionalità:** Supporta diversi servizi di geocodifica, tra cui Google Geocoding API, OpenStreetMap Nominatim, e altri. Può calcolare distanze tra coordinate geografiche e gestire diverse unità di misura.
- **Utilizzo:** Utilizzato in applicazioni che richiedono la conversione di indirizzi in coordinate geografiche, calcoli di distanze o altre operazioni geospaziali.

### 3. channels

- **Descrizione:** channels è un'estensione di Django che aggiunge supporto per protocolli di comunicazione asincroni come WebSocket, HTTP2 e altre tecnologie di comunicazione in tempo reale.
- **Funzionalità:** Estende il tradizionale modello di richiesta/risposta sincrono di Django per supportare operazioni asincrone, gestione di connessioni WebSocket, e altre funzioni in tempo reale.
- **Utilizzo:** Ideale per costruire applicazioni web che richiedono funzionalità in tempo reale, come chat, aggiornamenti live e notifiche push.

#### 4. uvicorn[standard]

- **Descrizione:** uvicorn è un server ASGI ad alte prestazioni per Python. È progettato per essere veloce e leggero, supportando protocolli asincroni.
- **Funzionalità:** Implementa il protocollo ASGI, che permette di costruire applicazioni web asincrone. Il pacchetto [standard] include dipendenze aggiuntive per migliorare le prestazioni e aggiungere funzionalità come la gestione di SSL e la compressione HTTP.
- **Utilizzo:** Utilizzato come server per applicazioni web basate su ASGI, come quelle sviluppate con framework asincroni come FastAPI o Django con Channels.

#### 5. whitenoise

- **Descrizione:** whitenoise è una libreria per la gestione dei file statici in applicazioni web Python, progettata per essere semplice e senza dipendenze esterne.
- **Funzionalità:** Serve file statici direttamente dall'applicazione senza la necessità di un server web separato. Supporta la compressione dei file, la memorizzazione nella cache e altre ottimizzazioni per migliorare le prestazioni.
- **Utilizzo:** Utilizzata per servire file statici (come CSS, JavaScript e immagini) in ambienti di produzione, semplificando la configurazione e migliorando le prestazioni delle applicazioni Django e di altri framework web Python. Permette di servire file statici facilmente se si utilizzano server come uvicorn o Daphne.

## Struttura del Progetto

Il progetto ha come struttura iniziale la vera e propria applicazione del server, che porta il nome del progetto:

- Applicazione Mapocalipse
  - In questa applicazione vengono gestite tutte le impostazioni del server
  - Come file principali ha:
    - `settings.py` nel quale si trovano tutte le impostazioni del server, come i percorsi delle cartelle oppure le applicazioni installate sul server. Qui viene anche impostato il riferimento al modello dell'utente.
    - `asgi.py` nel quale viene impostato il routing asincrono per permettere l'utilizzo di strutture dati come ad esempio gli websocket per la comunicazione real time
    - `urls.py` nel quale vengono definiti i percorsi primari verso le altre applicazioni

- Altre Applicazioni
  - Hanno come file principali tutti gli stessi file:
    - `urls.py` nel quale vengono definiti i percorsi verso le sue viste
    - `models.py` nel quale vengono definiti i modelli utilizzati per le interazioni con il database
    - `views.py` nel quale vengono definite le funzionalità delle viste
- Authentication
  - In questa applicazione viene gestito il processo di autenticazione
  - Questa applicazione ospita il modello del User
- Game
  - In questa applicazione viene gestita la schermata della scelta della modalità di gioco
  - Non possiede modelli
  - Effettua i redirect verso le altre 2 applicazioni
- Singleplayer
  - In questa applicazione viene gestita la schermata di gioco singleplayer
  - Per gran parte le sue viste sono utilizzate per richieste asincrone dal frontend
  - Ha come modelli:
    - `SinglePlayerLobby`
    - `Coordinates`
- Multiplayer
  - In questa applicazione viene gestita la schermata di gioco multiplayer
  - Gran parte delle sue viste sono utilizzate per richieste asincrone dal frontend
  - Ha come modelli:
    - `MultiPlayerLobby`
    - `Coordinates`(modello che utilizza una tabella diversa rispetto al modello `Coordinates` del Singleplayer)
    - `LobbyUser`(modello utilizzato per la tabella ponte tra User e `multiplayer_lobbies`)
  - Implementa anche gli websocket per la comunicazione real time tra gli utenti:
    - `consumers.py` nel quale vengono definiti i modelli(consumers) degli websocket
    - `routing.py` nel quale vengono definiti i percorsi di routing per collegarsi al websocket dal frontend

## Frontend

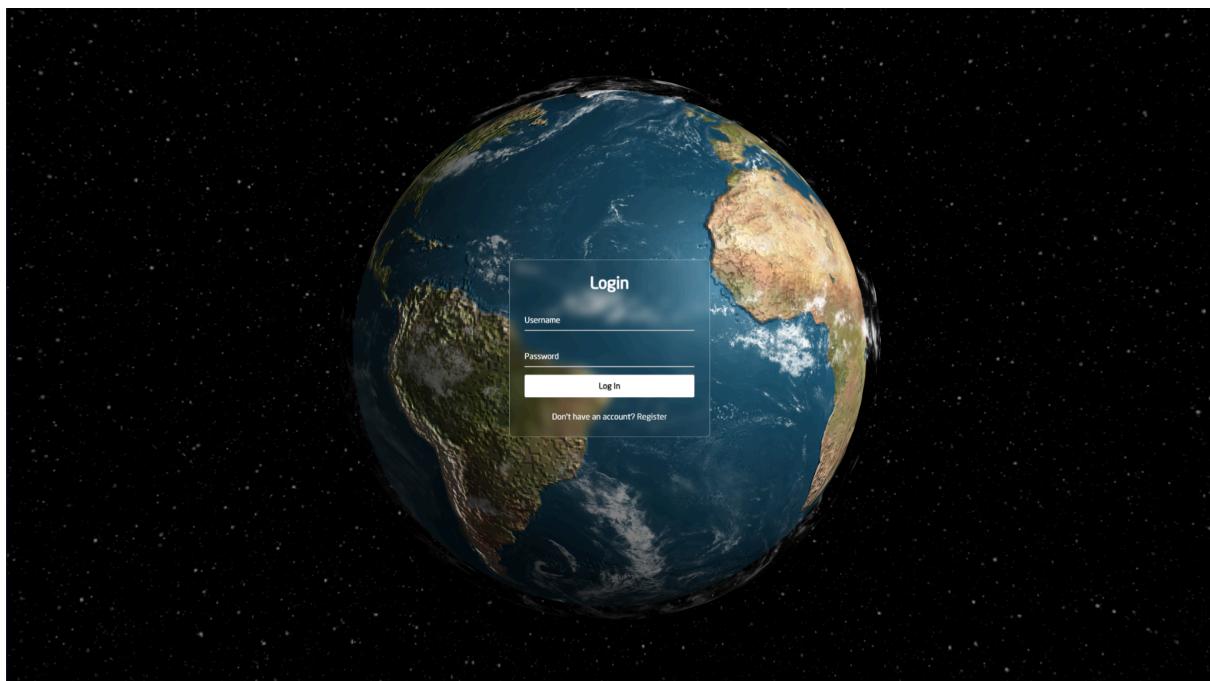
I linguaggi (di programmazione e non) utilizzati per lo sviluppo del frontend di Mapocalipse sono stati:

- HTML: per lo sviluppo della struttura della pagina web
- CSS: per lo stile grafico della pagina web
- JS: per la dinamicità della pagina web e per l'interazione della pagina web con il server

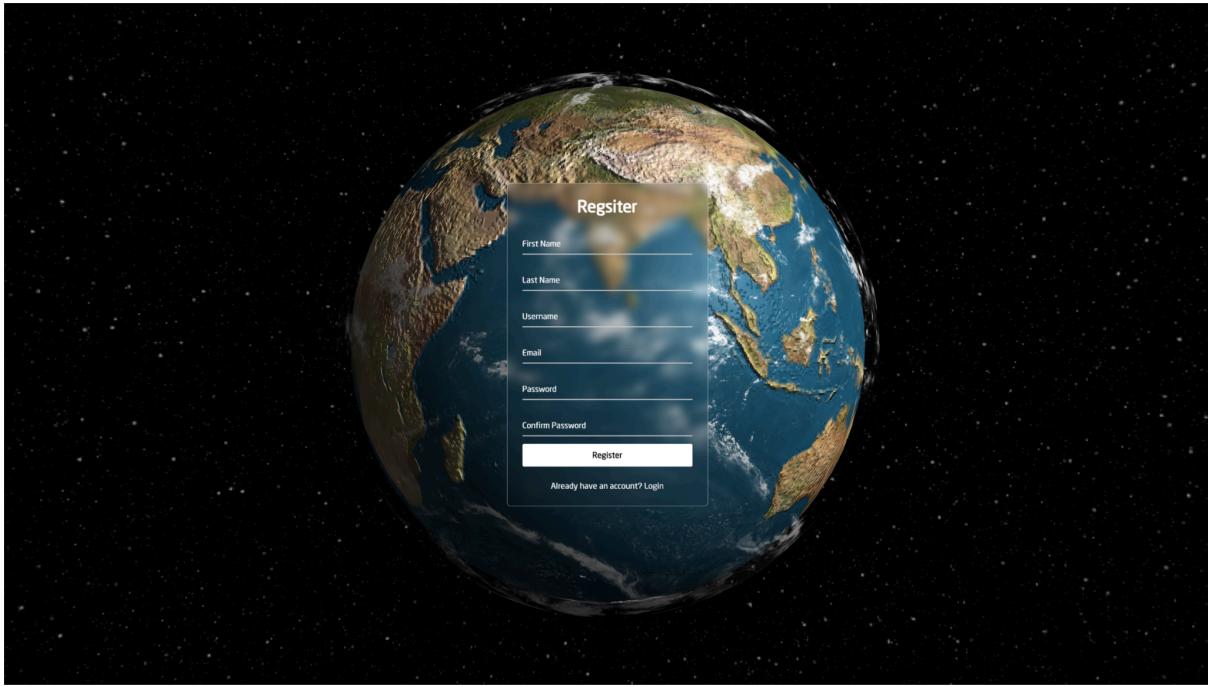
## Pagine

Il progetto Mapocalipse è composto da 5 principali pagine, una corrispondente a ciascuna applicazione del backend:

- Login/Register
  - Contiene i form di login e di registrazione, alternabili attraverso gli href su ciascuno dei 2 form. Il loro submit avviene con modalità asincrona tramite JavaScript, così in caso avvenga un errore, la pagina non si ricarica e lo mostra nel form giusto.
  - Entrambi i form hanno uno stile con sfondo trasparente leggermente sfocato
  - Come sfondo c'è un pianeta animato che gira nello spazio creato utilizzando la libreria three.js
  - Login Form:

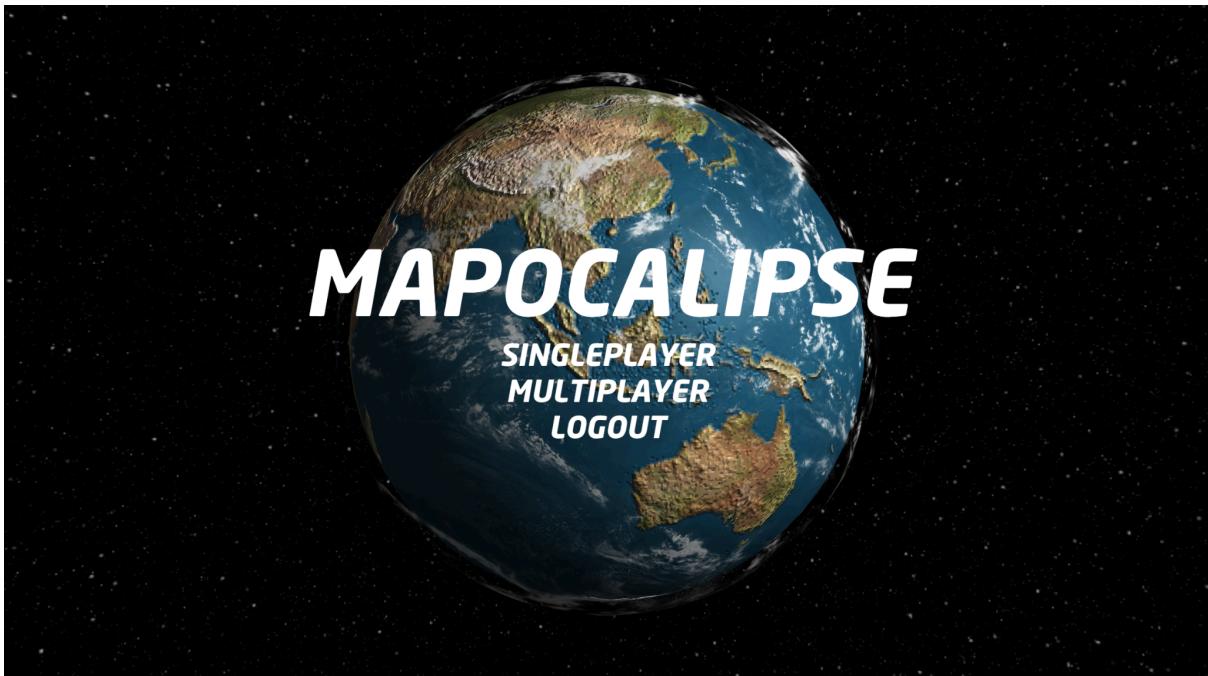


- Register Form:

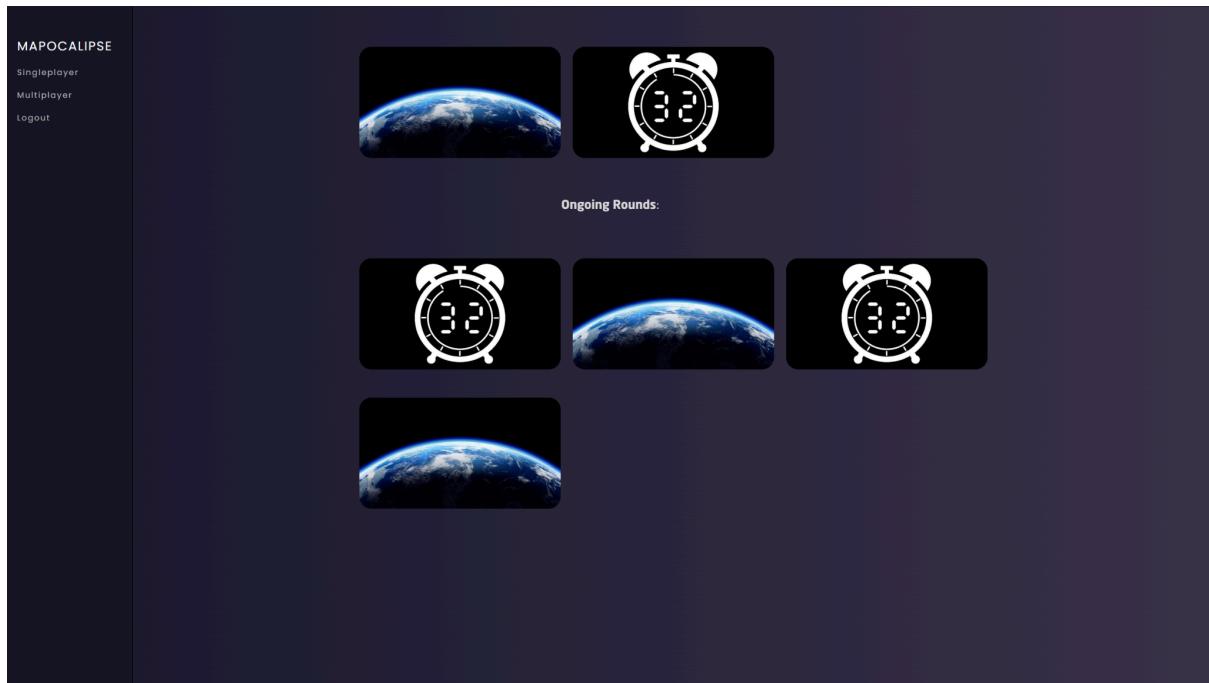


- GameMode Choice

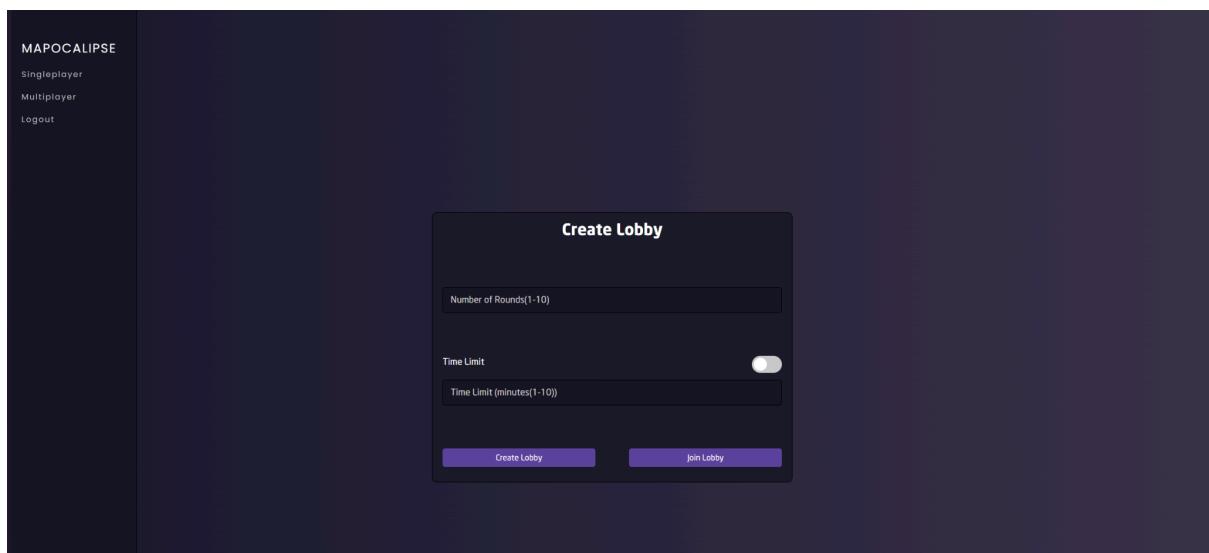
- Contiene il pulsante per accedere alle modalità singleplayer o multiplayer oppure per effettuare il logout
- Il font delle scritte è il Neo Sans Bold Italic
- Lo sfondo è lo stesso della pagina precedente



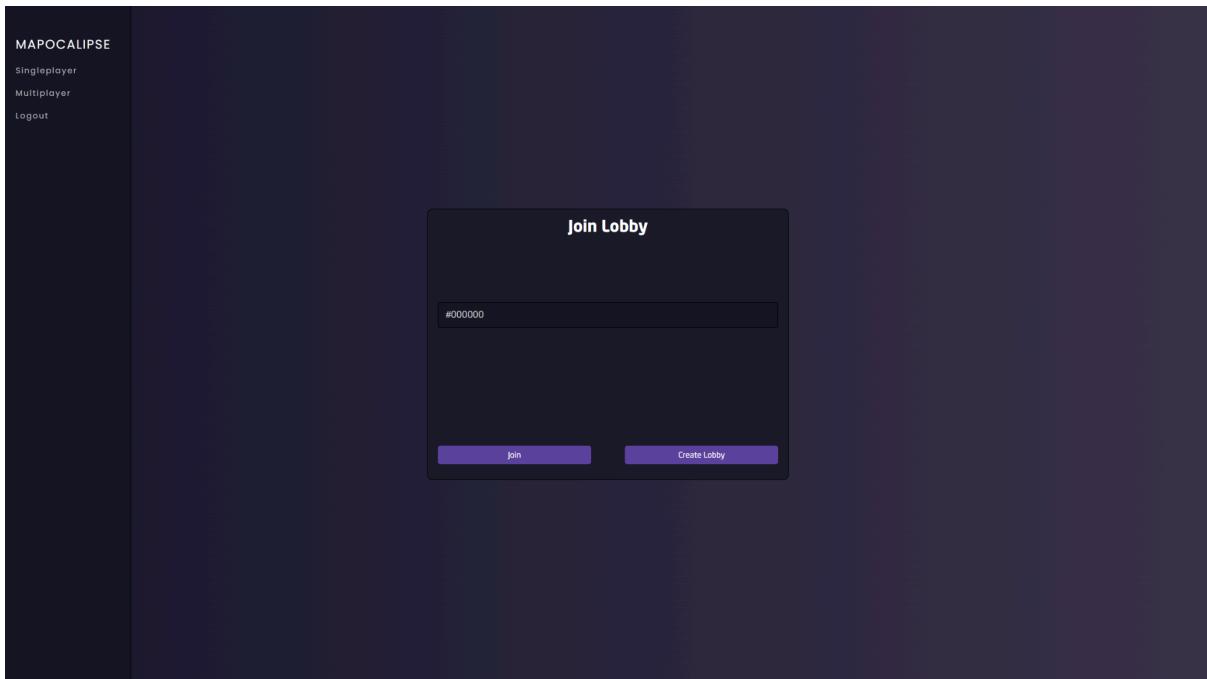
- SinglePlayer Page
  - Contiene i form per accedere alle 2 modalità di gioco disponibili e permette di visualizzare le partite lasciate in sospeso
  - Ha una sidebar che permette di accedere al multiplayer oppure di fare il logout
  - Le immagini delle due modalità di gioco funzionano in modo che, quando si passa con la freccia del mouse sopra, esca un form che descrive la modalità di gioco e ha il pulsante play(per questa animazione è stata utilizzata la libreria GSAP di JavaScript)



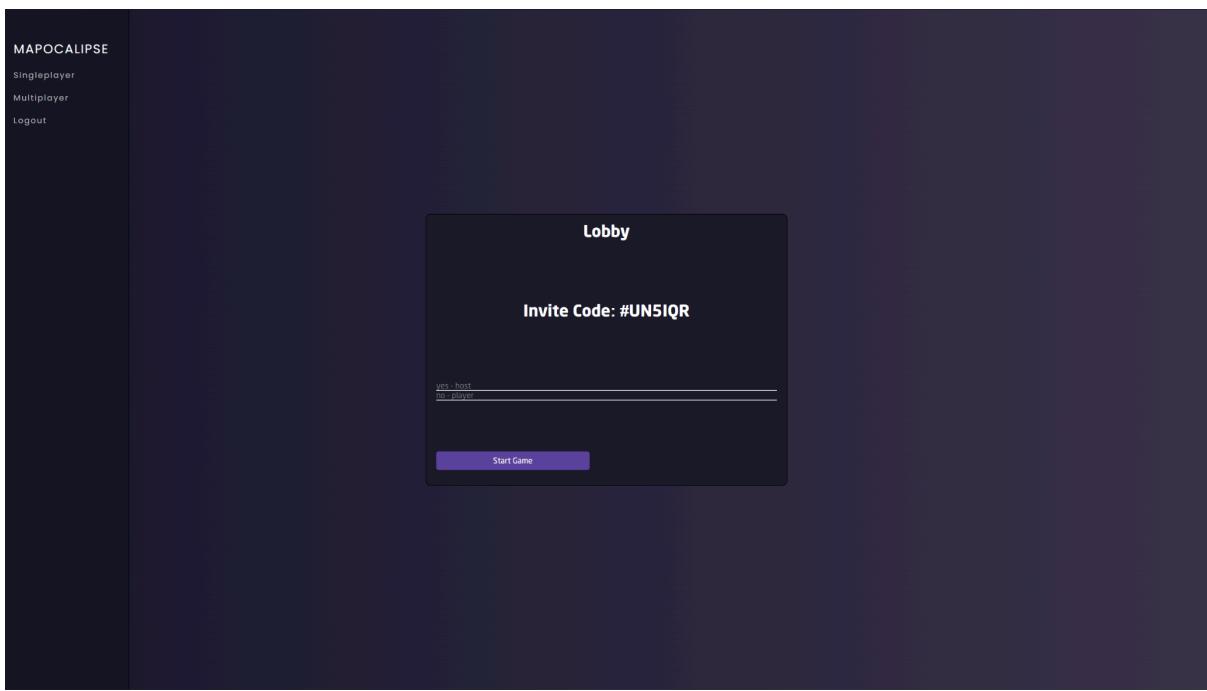
- MultiPlayer Page
  - Contiene 3 form grandi e due di loro si alternano attraverso pulsanti
    - I form sono:
      - Create Lobby



- Join Lobby



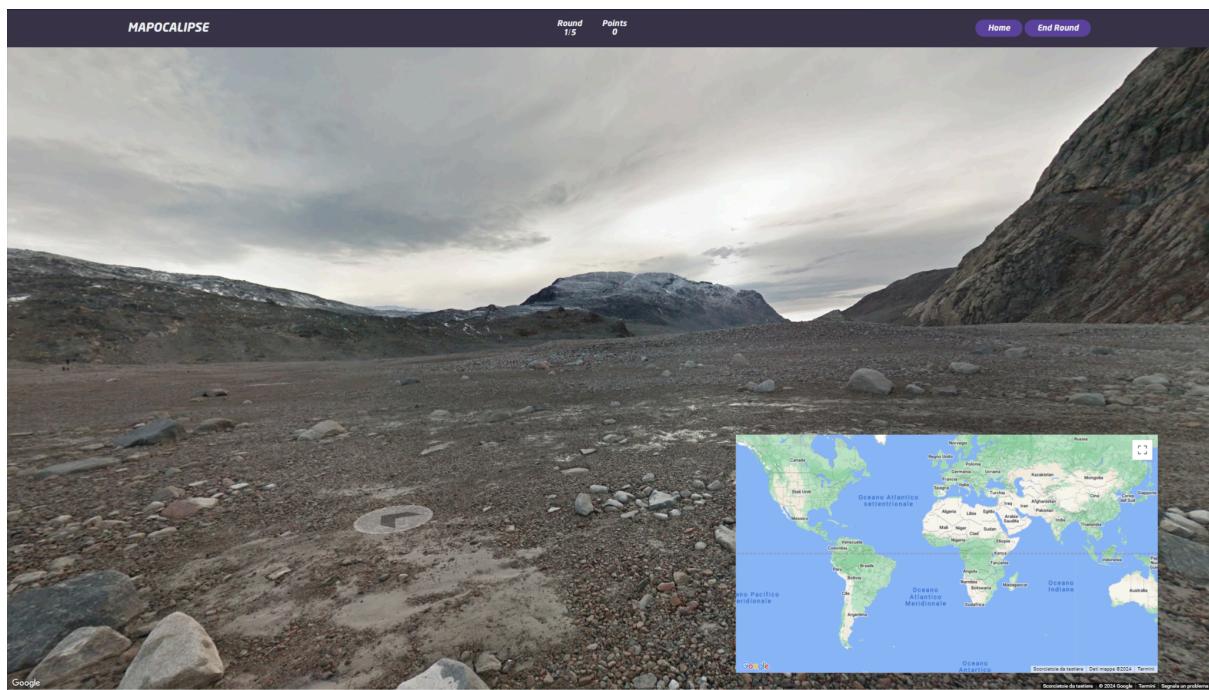
- Lobby



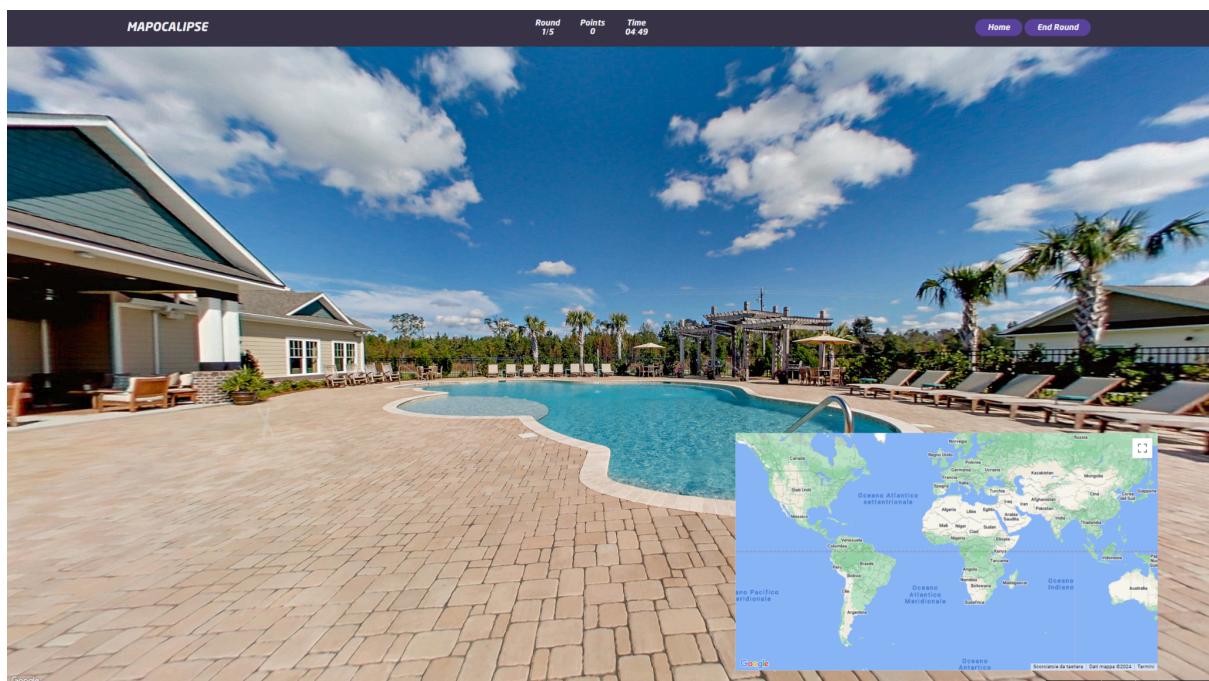
- Anche questa pagina ha la stessa sidebar della pagina precedente
- In questa pagina vengono utilizzati gli websocket tramite JavaScript per avviare la partita multiplayer(bisogna aspettare qualche secondo affinchè il collegamento avvenga prima di premere il pulsante)

- Game Page

- Lo stile di questa pagina è condiviso tra le due modalità di gioco(tranne che nel multiplayer non c'è il pulsante per sospendere la partita)
- Ha una navbar orizzontale che ha a sinistra il titolo, al centro il numero della partita, i punti e , se si gioca con il tempo limite, il timer e alla destra il pulsante per sospendere la partita o per terminarla
- Nella schermata principale abbiamo la street view nella quale ci si può muovere e una mappa in basso a destra su cui mettere il marker per poi confermare la posizione tramite pulsante. Per passare al successivo round c'è il pulsante Next che compare
- Normal Page



- TimeLimit Page



# Librerie

Per il frontend sono state utilizzate 2 librerie:

## 1. Three.js

- Libreria JavaScript per la creazione e visualizzazione di di grafica 3D nel browser. È costruita sopra WebGL(API che permette il rendering 3D su browser).
- **Funzionalità:**
  - **Scena:** Gestisce il contenuto della scena 3D, inclusi oggetti, luci e telecamere.
  - **Camera:** Definisce il punto di vista attraverso il quale la scena viene visualizzata. Supporta diverse proiezioni, come prospettica e ortografica.
  - **Mesh:** Combina geometrie e materiali per creare oggetti 3D renderizzabili.
  - **Geometria:** Definisce la forma degli oggetti 3D, come sfere, cubi e piani, così come geometrie più complesse personalizzate.
  - **Materiali:** Definiscono l'aspetto degli oggetti 3D, inclusi colori, texture, riflessioni e trasparenze.
  - **Luci:** Aggiunge illuminazione alla scena, supportando vari tipi di luci come direzionali, puntiformi e ambientali.
  - **Animazioni:** Gestisce le animazioni degli oggetti 3D, inclusi movimenti, rotazioni e trasformazioni.
  - **Shader:** Permette l'uso di shader personalizzati per effetti grafici avanzati tramite GLSL (OpenGL Shading Language).
  - **Controlli:** Offre strumenti per navigare e interagire con la scena 3D, come il controllo dell'orbita, del volo e del trackball.
- **Utilizzo:** è utilizzata per creare visualizzazioni 3D interattive, giochi, simulazioni e applicazioni grafiche avanzate nel browser. Facilita la gestione complessa di WebGL.

## 2. GSAP(GreenSock Animation Platform)

- Libreria Javascript altamente ottimizzata per la creazione di animazioni ad alte prestazioni che supporta una vasta gamma di animazioni
- **Funzionalità:**
  - **Tweening:** Anima le proprietà degli oggetti (come posizione, scala, rotazione e opacità) nel tempo. Supporta un'ampia

- gamma di interpolazioni (tween) e funzioni di easing per creare animazioni fluide e naturali.
- **Timeline:** Organizza e controlla sequenze di animazioni. Permette di sincronizzare e coordinare animazioni complesse attraverso una linea temporale.
  - **Easing:** Offre numerose funzioni di easing per controllare il ritmo delle animazioni, come linear, ease-in, ease-out, e altre combinazioni.
  - **Plugins:** Estende le funzionalità di base con plugin aggiuntivi, come l'animazione di testo, la manipolazione di SVG, la fisica e il morphing.
  - **Performance:** Ottimizzata per garantire animazioni fluide e ad alte prestazioni, minimizzando l'uso della CPU e della memoria.
  - **Utilizzo:** è utilizzata per creare animazioni web complesse e coinvolgenti, migliorando l'interattività e l'esperienza utente. È una scelta popolare per sviluppatori e designer che desiderano aggiungere animazioni dinamiche ai loro progetti web.

## API

È stata utilizzata una sola API Frontend per sviluppare Mapocalypse. Questa API ha permesso di sviluppare la parte più importante della logica del gioco in frontend:

- **Google Maps JS API**
  - Libreria offerta da Google per permettere l'integrazione di Google Maps nelle applicazioni web.
  - **Vasta gamma di servizi:**
    - Integrazioni di mappe
    - Marker
    - Overlay
      - permette l'aggiunta di polilinee, poligoni o altre immagini
    - Controlli
      - esempio Zoom in/out, fullscreen, eventi Javascript come il click
    - Personalizzazione delle mappe
  - **Utilizzo e Integrazione:**
    - **API Key:** Per utilizzare la Google Maps JavaScript API, è necessario ottenere una chiave API da Google Cloud Platform. La chiave API autentica le richieste e traccia l'utilizzo del servizio.
    - **Caricamento dell'API:** La libreria viene inclusa nella pagina web tramite uno script HTML che carica il file JavaScript dell'API.
    - **Configurazione:** Le mappe vengono configurate usando oggetti JavaScript, specificando le opzioni iniziali come il centro della mappa, il livello di zoom e il tipo di mappa.
    - **Eventi:** La API supporta la gestione di eventi per rendere le mappe interattive. È possibile definire funzioni di callback per eventi come clic sulla mappa, movimento del mouse, cambiamenti di zoom e altro.

- **Vantaggi:**
  - **Affidabilità e Dati Aggiornati:** Offre accesso ai dati di Google Maps, che sono continuamente aggiornati e accurati.
  - **Flessibilità e Personalizzazione:** Fornisce numerose opzioni di personalizzazione e strumenti per adattare le mappe alle esigenze specifiche di ogni progetto.
- **Considerazioni:**
  - **Costi:** L'utilizzo della Google Maps JavaScript API può comportare costi, basati sul numero di richieste e sui servizi utilizzati. È importante monitorare l'utilizzo per evitare costi imprevisti.
  - **Limiti di Quota:** Google impone limiti di quota sull'uso delle API. È possibile richiedere aumenti di quota attraverso la console di Google Cloud Platform se necessario.
  - Import API(diciamo molto "semplice" e "intuitivo", vero Google?!?!)

```
<script>
  window.loadGoogleMapsApi = (g => {
    var h, a, k, p = "The Google Maps JavaScript API", c = "google"
    , l = "importLibrary", q = "__ib__", m = document, b = window;
    b = b[c] || (b[c] = {});
    var d = b.maps || (b.maps = {}), r = new Set, e = new
URLSearchParams;
    return () => h || (h = new Promise(async (f, n) => {
      await (a = m.createElement("script"));
      e.set("libraries", [...r] + "");
      for (k in g) e.set(k.replace(/[A-Z]/g, t => "_" + t[0]).toLowerCase()), g[k]);
      e.set("callback", "initMap");
      a.src = `https://maps.${c}apis.com/maps/api/js?` + e;
      d[q] = f;
      a.onerror = () => h = n(Error(p + " could not load."));
      a.nonce = m.querySelector("script[nonce]")?.nonce || "";
      m.head.append(a)
    }));
    d[l] ? console.warn(p + " only loads once. Ignoring:", g) : d[l]
    ] = (f, ...n) => r.add(f) && h.then(() => d[l](f, ...n))
  })({ key: "YOUR_API_KEY", v: "weekly" });
</script>
```

## Avvio del server

Per avviarlo basta avere installato Python 3.12 e avviare il file `start_http.bat` che creerà un virtual environment(.venv) e installare le librerie necessarie per poi avviare il server. Quel file può essere utilizzato per inizializzare il server siccome attua i controlli su ciò che è già installato.

Il database utilizzato è il MariaDB presente su XAMPP.

Bisogna importare il file `mapocalypse.sql` in XAMPP per avere un database pronto all'uso con 2 account già esistenti:

- username: yes — password: yes
- username: no — password: no

## Idee da aggiungere in futuro

Possono ancora essere aggiunte, secondo me, tante altre cose come ad esempio:

- Mappe per singolo paese(ovviamente quelli più conosciuti).
- Mettere sullo sfondo delle schermate del menu singleplayer e del multiplayer un'immagine leggermente trasparente che si mescola con il colore dello sfondo.
- Cambiare le grafiche della lobby in modo che mostri meglio i giocatori connessi.
- Aggiungere un'icona animata come simbolo del gioco nei menu.