

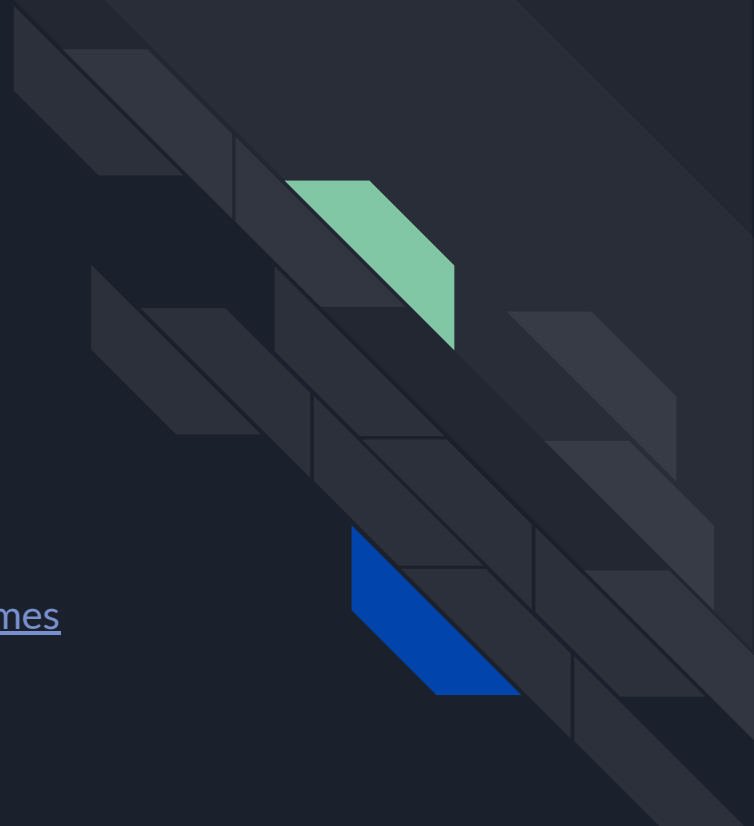
# Progetto '*RevGames*'

Componenti del gruppo :

- Carmine Marchesani
- Patryk Bialowas
- Marco Cocilova

Link Github :

<https://github.com/MrEntity303/RevGames>





# In cosa consiste...

Il progetto consiste nella realizzazione di una **Single Page Application**, che permette agli utenti di registrarsi, loggarsi, ed effettuare recensioni sui videogiochi.

Ovviamente sarà comunque possibile visualizzare tutte le recensioni esposte dal sito pur non avendo effettuato il login.

Oltre all'utente normale, sarà presente anche un utente admin, che avrà la possibilità di inserire dei videogiochi, tramite la sezione dedicata.

Nella home sarà presente un catalogo di videogiochi che l'utente potrà recensire solo una volta loggato, e un menu laterale a comparsa, con tutti i pulsanti che svolgeranno diverse azioni a seconda di cosa verrà cliccato.



# Frontend. Le tecnologie utilizzate

La tecnologia usata per il Frontend è Vue.js, un framework Javascript per la creazione di interfacce utente dinamiche.

Una delle principali caratteristiche di Vue.js è il suo approccio basato sui **componenti**. Il framework consente di creare componenti riutilizzabili e modulari, che rappresentano parti specifiche dell'interfaccia utente.

Grazie alla sua modularità i componenti possono essere facilmente combinati per creare interfacce complesse.




# Backend. Le tecnologie utilizzate

La tecnologia usata per il backend è Express.js, un framework web leggero e flessibile per Node.js, progettato per facilitare la creazione di applicazioni web e API RESTful, ovvero ad ogni richiesta al server, quest'ultimo non tiene salvato in memoria lo stato delle richieste.

Per accedere a una risorsa specifica, utilizziamo un URL univoco chiamato "**endpoint**" che identifica la risorsa desiderata a cui si vorrà accedere.

Abbiamo scelto questo framework perché fornisce metodi per il routing e numerosi middleware per gestire in modo facile le richieste e risposte HTTP.



# Principali librerie utilizzate per lo sviluppo del backend...

**'jsonwebtoken'** - una libreria usata per generare e gestire i token per l'autenticazione di un utente. La generazione dei token avviene attraverso il metodo appropriato della libreria e due chiavi segrete situate nel file `.env`.

A seguito della generazione, sono stati implementati i singoli metodi per la verifica dei token. Un vantaggio dell'uso del JWT è che la sessione specifica non viene salvata sul DB, ma sul **local storage dell'utente**.

**'bcrypt'** - una libreria che utilizza una combinazione di hashing e "salting" per aumentare la sicurezza delle password. Il salting comporta l'aggiunta di dati casuali alla password prima di fare l'hash definitivo. Noi abbiamo aumentato la sicurezza dell'hash finale concatenando la password originale con una **secret key**, situata nel file `.env`.

**'Sequelize'** - una libreria di **ORM (Object-Relational Mapping)**, che permette l'interazione con i database relazionali. Il punto di forza di questa libreria sono i metodi che fornisce agli oggetti per eseguire le query. Infatti, non ci siamo dovuti preoccupare della **SQL Injection**, poiché i metodi eseguono query chiuse correttamente, e che impediscono l'injection di script nei form di input.



## ... e del frontend

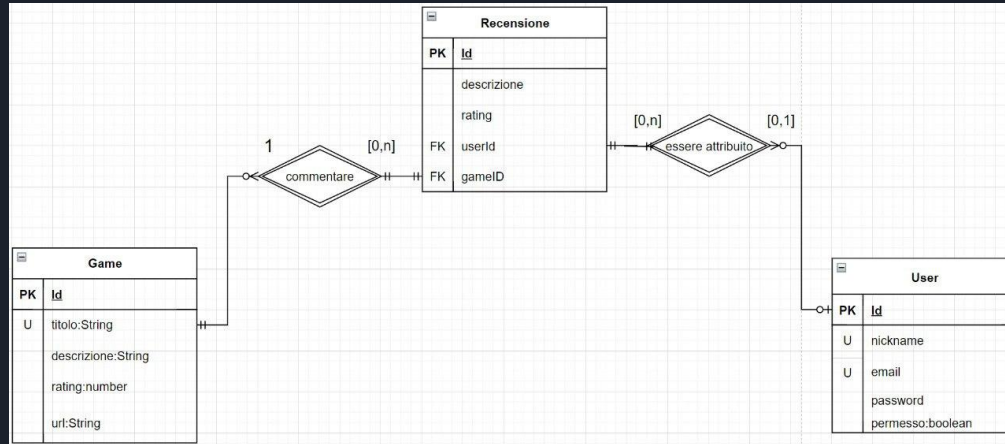
‘**vuex**’ - e’ una libreria che permette la gestione dello **stato** dell’applicazione, condiviso tra i vari componenti, in modo predeterminato e prevedibile. Nel progetto questa libreria si può notare dalla presenza di un file ‘**store.js**’.

Successivamente i componenti possono interrogare questo file per conoscere lo stato dell’applicazione.

‘**axios**’ - libreria che funge da middleware tra il client e il server. Grazie ai suoi metodi, permette di mettersi in mezzo a frontend e backend, gestendo le richieste, e favorendo una comunicazione agevole tra le due parti.

# Scelta del database

Per il popolamento dei dati, abbiamo scelto un database MySQL hostato online, ma utile al solo scopo di testing e sviluppo, e non usabile per le applicazioni in **production**, dove è consigliabile usare database più affidabili e prestanti.



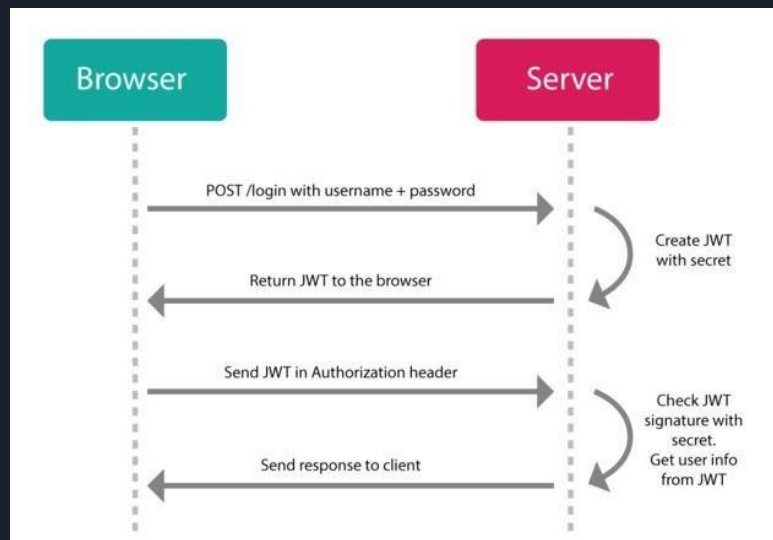
# Sicurezza

La tecnologia scelta per garantire la sicurezza degli utenti è il **JWT** (Json Web Token), che consiste nel fornire all'utente appena loggato un **access token** ed un **refresh token**.

Ogni volta che l'utente farà delle richieste a risorse protette che richiedono l'autenticazione, il server verificherà la validità dell'access token e, nel caso sia scaduto, il server genererà dei nuovi access e refresh token per quel singolo utente, passando il **refresh token** come **Header** della richiesta.

Nel frontend per garantire questo scambio di informazioni, saranno presenti degli **interceptor**, che controllano ogni richiesta, componendola del relativo **header** e reindirizzandola.

Stabilito che l'utente non è autorizzato ad accedere a determinate risorse o funzionalità, egli sarà bloccato dal **backend** e reindirizzato alla pagina di login dal lato **frontend**.







# CORS

Un problema che abbiamo riscontrato consiste nel fatto che non erano consentite le chiamate al server per via di un meccanismo di sicurezza chiamato CORS implementato dai browser, per limitare le richieste **HTTP** tra origini diverse.

Per risolvere questa problematica abbiamo dovuto specificare dal lato backend l'origine dalla quale erano permesse le chiamate verso il server, e quindi permettere il cors da quell'origine.