Portal web de jocs en línia

Szymon J. Libuszowski Borzecki

2 de juliol de 2024

Resum– En aquest projecte he desenvolupat un portal web de jocs en línia utilitzant Node.js per construir el *back end* i Vue.js pel *front end*. He dissenyat el *front end* mitjançant una Single Page Application combinat amb llibreries com Pinia, i18n o Media Query, per aconseguir la millor experiència d'usuari possible. En el *back end*, he utilitzat una base de dades relacional MySQL. Per comunicar de manera eficient el client i servidor he dissenyat una API REST amb Express.js. A més, he implementat una sèrie de tests unitaris automatitzats utilitzant Cypress i Gitlab Runners per poder executar aquests tests sempre que faci qualsevol canvi dins l'aplicació. També he realitzat *pentesting* amb l'objectiu d'aconseguir la màxima seguretat possible davant de ciberatacs com el SQL injection o el XSS. Finalment, he desplegat l'aplicació en la plataforma Azure, sota el domini *szymongames.com* i xifrant la connexió amb el protocol TLS per garantir la privacitat dels usuaris.

Paraules clau– Front end, Back end, SPA, Seguretat, API, Cookies, Node.js, Vue.js, MySQL, Cypress, Pinia, SQL injection, XSS, Internacionalització, Integració Contínua, Framework.

Abstract– In this project I have developed an online gaming portal using Node.js in order to build the *back end* and Vue.js for the *front end*. I have designed the *front end* using a Single Page Application combined with libraries like Pinia, i18n or Media Query to achieve the best possible user experiencie. In the *back end* I have used a MySQL relational database. To communicate efficiently the client and the server I have designed an API REST with Express.js. Furthermore, I have implemented some automated unit tests using Cypress and Gitlab Runners to execute these tests after doing any kind of changes in the application. Also, I have done *pentesting* so I could achieve the best possible security against cyber-attacs such as SQL injection or XSS. Finally, I have deployed my application on Azure platform, under the domain of *szymongames.com* and encrypting the connection using the TLS protocol to ensure users privacy.

Keywords– Front end, Back end, SPA, Security, API, Cookies, Node.js, Vue.js, MySQL, Cypress, Pinia, SQL injection, XSS, Internationalization, Continuous Integration, Framework.

1 Introducció - Context del treball

Al llarg del segle XXI, la indústria dels videojocs s'ha convertit en una de les més grans i importants arreu del món. Segons un article de newZoo [1], l'any 2023 hi va haver uns 3305 milions de videojugadors (Fig. 1) i s'han generat més de 180000 milions de dòlars. Només a Espanya, l'any 2022 hi va haver 18,2 milions de *gamers* i es va facturar un total de 2012 milions d'euros, segons l'informe de l'Asociación Española de Videojuegos [2]. Però, com és que tanta gent juga a videojocs? Per respondre aquesta pregunta ens hem de fixar en l'estudi Power Of Play [3], un estudi que va entrevistar a més de 12000 jugadors habituals de videojocs de tot el món. Les respostes dels enquestats donen a entendre que, a part d'ajudar a relaxar-se i divertir-se, els videojocs són una gran forma d'evadir-se dels problemes del dia a dia (64% dels enquestats) o reduir l'estrès i ansietat (71% i 61% dels enquestats, respectivament) entre altres beneficis. A més, un 73% dels enquestats opina que els videojocs milloren la creativitat de les persones, un 69% creu que milloren la capacitat cognitiva i la resolució de problemes, a part de millorar la competència lingüística i la comunicació si estem jugant amb un grup de persones.

Aquesta idea de TFG sorgeix de la meva passió pel món dels videojocs, que des de sempre m'han interessat, tant jugar-los com el món que els envolta. Fins i tot, he arribat a dissenyar-ne uns quants. De fet, vaig començar el grau d'enginyeria informàtica amb l'objectiu de profunditzar i aprendre a programar-los. Per tant, desenvolupar un portal de jocs en línia era perfecte per poder combinar la meva

• E-mail de contacte: szymon010402@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Maria Carmen de Toro Valdivia (Àrea de Ciències de la Computació i Intel·ligència Artificial)
• Curs 2023/2024

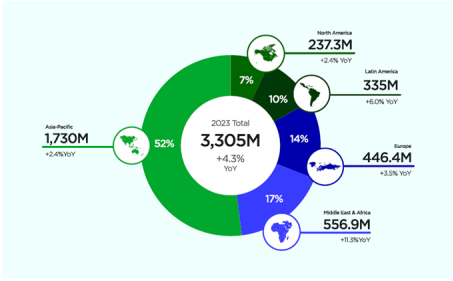

Fig. 1: Quantitat de jugadors arreu del món.

passió amb el que he après durant quatre anys i estendre els meus coneixements sobre disseny d'aplicacions webs.

Per aquest projecte he desenvolupat un portal web per poder jugar a diversos videojocs en el navegador[1]. Per fer-ho, he utilitzat *frameworks*[2] de JavaScript, tant pel disseny del *front end* com del *back end*, amb la finalitat de tenir tota l'aplicació amb JavaScript per facilitar el manteniment i desenvolupament de l'aplicació. A més a més, en la part del servidor he fet servir una base de dades relacional que permet emmagatzemar i organitzar les dades de manera estructurada en forma de taules. Per assegurar el correcte funcionament de l'aplicació he realitzat un seguit de tests unitaris compaginats amb tests de seguretat, també coneguts com a *pentesting*. La finalitat d'aquests tests és provar el correcte funcionament de la web i trobar possibles defectes o *bugs*, tant funcionals com de seguretat. Com que dur a terme aquests tests de forma manual pot arribar a ser una cosa tediosa, he realitzat una integració contínua per executar-los de manera automàtica. Finalment, he dissenyat el portal de manera *responsive* per tal d'assegurar una bona experiència d'usuari independentment des de quin dispositiu s'accedeixi a l'aplicació.

La resta d'aquest document està estructurada de la següent manera:

A continuació, hi ha l'estat de l'art, on explico la situació actual de les aplicacions web de jocs en línia, quines són les més destacades i quantes visites mensuals tenen. Seguidament, presento els objectius del projecte i la metodologia utilitzada. A continuació, la secció de disseny i implementació, on explico de manera detallada com he dissenyat l'aplicació i quines tecnologies he utilitzat i com les he utilitzat dins el projecte. Més tard, hi ha les seccions de *testing* i *pentesting*, on parlo sobre la implementació dels tests automatitzats amb Cypress i el procés de *pentesting*. Posteriorment, explico com publicar una web de forma que sigui accessible per a tot el públic. Finalment, analitzo els resultats finals del projecte comparant-los amb els objectius inicials, seguit de les conclusions.

2 Estat de l'art

A la introducció he parlat de la importància dels videojocs en l'actualitat i la gran influència que tenen en la societat. D'altra banda, però, si ens centrem exclusivament en els jocs de navegador, tema principal d'aquest treball, podem observar la gran quantitat de gent que visita aquestes pàgines (Fig. 2, 3). Les següents xifres han estat extretes del portal similarweb[3].

En l'àmbit global destaquen sobretot *poki.com* amb més de 100 milions de visites i *crazygames.com* amb gairebé 50 milions de visites al llarg del mes d'abril de 2024. També podem observar webs com *friv.com* amb 14 milions de visites o *agame.com* i *armorgames.com* amb 1,5 milions i 2 milions de visites respectivament.

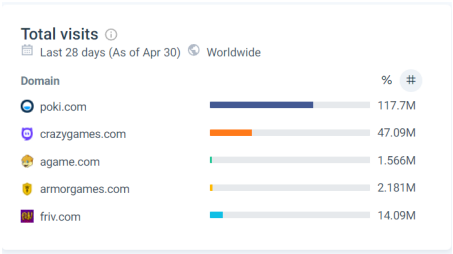

Fig. 2: Visites a webs de jocs l'abril de 2024.

Pel que fa a webs hispanoparlants, les xifres òbviament no són tan altes, però continuen sent altes. Podem trobar els portals *1001juegos.com* amb gairebé 6 milions de visites al mes d'abril, *minijuegos.com* amb gairebé 5 milions o *juegos.com* amb més d'1 milió. Després ja trobem webs com *juegosjuegos.com* , *juegosdiarios.com* o *juegosdechicas.com* que no arriben al mig milió de visites.

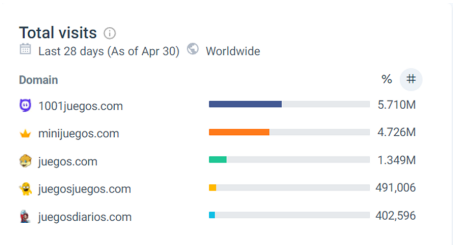

Fig. 3: Visites a portals de jocs hispans l'abril de 2024.

El portal que he dissenyat en aquest projecte[4], és un portal de videojocs amb les mateixes funcionalitats bàsiques que tots els mencionats anteriorment, però el que diferencia el meu dels altres és que els usuaris poden estar segurs de la seva privacitat, ja que no utilitzo *cookies* de tercers. Un altre punt a favor és que no hi ha anuncis.

3 Objectius

Els objectius principals d'aquest projecte són:

- Dissenyar un portal web per jugar a videojocs en el navegador utilitzant les tecnologies de Vue.js i Node.js.
- Dissenyar un portal web *responsive*.
- Permetre als usuaris registrar-se i iniciar sessió en el portal d'una forma segura.
- Aconseguir una experiència d'usuari molt positiva.

[1] https://szymongames.com

[2] Esquema o marc de treball que ofereix una estructura base per elaborar un projecte.

[3] https://www.similarweb.com/

[4] El codi font d'aquest projecte es pot descarregar i veure a: https://gitlab.com/szymon0104022/minijuegosdeSzymon

- Implementar un sistema d'integració contínua en el projecte per executar tests unitaris cada vegada que realitzi canvis en el projecte.
- Aconseguir la màxima seguretat possible davant d'atacs potencials.

4 METODOLOGIA

He fet aquest treball mitjançant una metodologia de tipus àgil com és Kanban [4]. Kanban, en japonès "rètol", és una metodologia que consisteix en un tauler (o rètol) format per columnes sobre les quals els membres de l'equip col·loquen les seves tasques en funció del seu estat.

Generalment, aquest tauler conté (com a mínim) tres columnes: pendent, en procés i finalitzat. En primer lloc, els membres de l'equip col·loquen totes les tasques en la columna de "pendent', llavors, les tasques començades passen a formar part de la columna "en procés" i, un cop finalitzes, passen a l'estat de "finalitzat". Aquesta metodologia permet controlar d'una forma molt visual i senzilla la feina a fer.

M'he decantat per utilitzar aquesta metodologia gràcies a la facilitat que proporciona a l'hora de visualitzar i organitzar les tasques. A més, considero que és una metodologia que pot adaptar-se molt bé a les necessitats del projecte gràcies a la seva gran flexibilitat per afegir tasques, ja que permet distribuir-les de la forma que consideri més adequada. Com que aquesta metodologia no es basa en *sprints*[5] permet la possibilitat d'anar afegint a poc a poc més tasques i fer-les quan sigui més convenient. Així, puc centrar-me al màxim en dur a terme la tasca de la millor forma possible sense preocupar-me de cap data límit.

5 DISSENY I IMPLEMENTACIÓ

En aquest apartat explicaré com he realitzat el projecte des de la part tècnica. Quines tecnologies he utilitzat (Fig: 4) i les decisions que he pres.

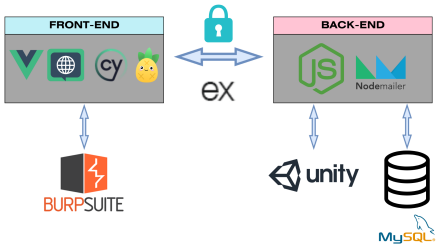

Fig. 4: Esquema de les tecnologíes utilitzades.

5.1 Disseny *front end*

El *front end* és la part visual de l'aplicació. És tot el que l'usuari veu i amb el que interactua quan accedeix a una pàgina web: els botons, els texts, els colors, el disseny, etc.

Hi ha diverses tècniques per dissenyar el *front end* d'una aplicació web, entre elles hi ha: Single Page Application, Multi-Page Application i Progressive Web Application. A continuació una petita explicació sobre aquestes tècniques:

- Single Page Application (SPA): Tipus d'aplicació web que utilitza només una pàgina que carrega la informació necessària (components) de forma dinàmica sense la necessitat de recarregar tota la pàgina. Oferint així una millor experiència d'usuari.
- Multi Page Application (MPA): Aquestes aplicacions consten d'una gran quantitat de pàgines emmagatzemades en el servidor. Quan l'usuari interactua amb la web, el servidor envia la pàgina sencera al client, qui la renderitza sencera abans de mostar-la a l'usuari.
- Progressive Web App (PWA): Aquestes aplicacions són un intermedi entre aplicacions web i aplicacions tradicionals. Són pàgines web que poden arribar a executar-se fora de la web i sense connexió Wi-Fi, assimilant-se a una aplicació de mòbil o d'escriptori.

Finalment, m'he decantat per una SPA per poder crear una web senzilla, dinàmica i proporcionar una molt bona experiència d'usuari.

5.1.1 Framework

Dins l'àmbit de construcció de front-end hi ha moltes alternatives entre les quals escollir. Els *frameworks* més populars són React, Angular i Vue. D'aquests tres llenguatges, segons una enquesta de Stack Overflow[6], el més famós i estimat per a la comunitat és React. Entre Vue i Angular, ens trobem que Angular és més famós que Vue, però menys estimat.

React és un framework molt flexible i fàcil d'aprendre que ofereix un gran rendiment. El principal inconvenient que té, és que necessita suport d'altres llibreries per algunes funcionalitats com l'encaminament, gestió de l'estat o peticions HTTP.

D'altra banda, hi ha Angular, un *framework* amb una corba d'aprenentatge molt alta i una sintaxi que pot arribar a ser molt complexa comparat amb els altres dos.

Vue, en canvi, és un framework amb la corba d'aprenentatge més baixa dels tres i que ofereix una gran quantitat de documentació, però que no arriba a ser tan flexible com pot ser React.

Finalment, m'he decantat per utilitzar Vue [5]. Principalment per la seva facilitat d'ús i l'enorme quantitat de documentació, a més que ja tenia uns coneixements bàsics.

5.1.2 Routing

Com he comentat abans, una SPA és una aplicació que només utilitza una pàgina que va recarregant el seu contingut. Per aconseguir aquesta funcionalitat, és necessari un *front end router*, una eina que treballa a la part del client i carrega el contingut necessari en el moment necessari. Pel projecte he utilitzat vue-router [6], el *router* que proporciona Vue.

El seu funcionament és molt senzill. S'ha d'especificar les rutes[7] que tindrà l'aplicació, a cada ruta se li assigna

[5] Una sessió de treball de temps fix (generalment menor a un mes) on l'equip realitza certa quantitat de tasques definides que haurien d'estar acabades quan finalitza el *sprint*.

[6] https://survey.stackoverflow.co/2023/#technology

[7] Configuracions que lliguen URLs (*entrypoints* o punts d'entrada) amb els components a carregar.

un component (contingut a carregar) i un URL. D'aquesta forma el *router* sap que quan l'usuari accedeixi a un URL específica, ha de carregar el component especificat. Un cop especificades les rutes, ja es pot començar a utilitzar el *router* dins l'aplicació.

5.1.3 Dades persistents

Hi ha alguns casos específics on és necessari guardar certa informació dins el navegador de l'usuari, ja sigui per millorar l'experiència d'usuari o el rendiment de l'aplicació. L'exemple més típic és l'inici de sessió. En aquest cas, s'ha de guardar informació sobre la sessió dins el navegador, perquè l'usuari mantingui la sessió iniciada al recarregar la pàgina o al accedir-hi un altre dia. Aquesta informació es guarda en forma de *cookies*.

Les *cookies* són una peça d'informació enviada per un lloc web i guardada en el navegador del client per poder mantenir l'estat d'una connexió. Hi ha moltes llibreries que permeten emmagatzemar informació en el navegador del client en forma de *cookies*, entre elles Vuex o Pinia [7]. Les dues llibreries són similars, Vuex va ser la llibreria oficial de Vue per mantenir l'estat, però des de fa un temps ho és Pinia, una llibreria que té les mateixes funcions que Vuex però millorades. Per aquest motiu, m'he decantat per utilitzar Pinia.

Pinia utilitza *stores*, una entitat que emmagatzema tant la informació com la lògica de l'aplicació sense estar vinculada al DOM[8], es podria dir que és aliena a l'estructura de l'aplicació. Dins aquest *store* he definit un estat (variables), que guarda la informació de l'usuari en el navegador, i unes accions (mètodes), que realitzen la lògica.

En el cas d'iniciar sessió, per exemple, en l'estat he definit una variable que guarda informació com el correu, ID o foto de perfil de l'usuari. Dins les accions, he creat mètodes per poder iniciar sessió, tancar sessió o canviar la foto de perfil.

5.1.4 Internacionalització

Internacionalitzar una aplicació significa adaptar l'aplicació als diversos idiomes, regions i cultures. Per tant, traduir els textos, el format de data, etc. Per realitzar-ho, he utilitzat I18n [8], el *plugin* d'internacionalització per Vue.

Per utilitzar aquesta llibreria he hagut de dissenyar un diccionari de tipus clau-valor per a cada idioma. Les claus (o entrades) en cada diccionari han de ser les mateixes, ja que aquesta clau és la que s'utilitza per identificar el text dins del codi. Cada clau conté un valor, que representa el text traduït a la llengua pertinent. A més d'aquest diccionari, cal especificar el *locale* (idioma en què l'usuari vol que estigui l'aplicació), el qual guardo mitjançant *cookies*.

Quan el diccionari d'internacionalització està creat, només queda utilitzar-lo en el codi JavaScript o HTML.

5.1.5 Aplicació *responsive*

Una aplicació *responsive* és aquella que és funcional i s'adapta a qualsevol dispositiu. Per aconseguir-ho, he utilitzat Media Query, una tècnica de CSS per adaptar el contingut de la pàgina a la resolució de la pantalla.

[8]Estructura en forma d'arbre dels components HTML de la web.

He creat quatre dissenys per a quatre resolucions diferents amb la finalitat d'aconseguir que l'aplicació sigui completament *responsive* (Fig. 5):

- Amplada menor a 600 píxels: Disseny pels dispositius mòbils quan estan en vertical.
- Alçada menor a 300 píxels: Disseny per adaptar els *pop ups* d'inici de sessió o registre en els dispositius mòbils quan estan en horitzontal.
- Amplada entre 600 píxels i 1300 píxels: Disseny per les tauletes.
- Amplada major a 1300 píxels: Disseny pels ordinadors.

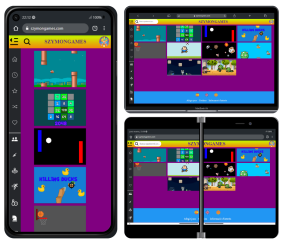

Fig. 5: Contingut de l'aplicació en diferents dispositius.

5.1.6 Seguretat en el *front end*

La ciberseguretat és una qüestió fonamental a l'hora de construir una aplicació web, ja que sinó, es pot comprometre la privacitat de milers de persones. Per garantir una bona seguretat, s'han d'implementar mesures tant en el *front end*, com en el *back end*.

En el cas del *front end*, dins de tots els camps del formulari (que no siguin la contrasenya), he especificat quins caràcters són valids i quins no a través de l'atribut HTML *pattern*.

5.2 Disseny *back-end*

El *back end* és la part de l'aplicació encarregada de què tota la lògica de la pàgina web funcioni. El *front end* accedeix al *back end* a través d'una API REST.

El back-end d'una pàgina web pot ser construït amb gairebé qualsevol tecnologia existent, es pot utilitzar Python, Java, JavaScript, Ruby, PHP, C, C++, Perl, entre d'altres. Durant molts anys el líder en tecnologies back-end ha estat PHP, però cada vegada s'està utilitzant menys degut a la seva robustesa i que pot arribar a ser difícil de mantenir. En canvi, s'estan utilitzant més tecnologies com Pyhton (Django), Ruby (Ruby on Rails) o JavaScript (Node.js [9]).

Les tres tecnologies tenen els seus punts a favor i en contra, per exemple Django és molt fàcil d'aprendre i és molt escalable, però no és tan bo pel disseny de webs com ho poden ser Ruby o Node. D'altra banda, Ruby on Rails és una tecnologia que permet generar codi de gran qualitat utilitzant el patró Model-Vista-Controlador, però pot arribar a ser molt complex. Finalment, Node.js és una tecnologia molt escalable amb un molt bon rendiment, però que pot arribar a ser complicat gestionar els fluxos asíncrons.

En el meu cas, m'he decantat finalment per Node, gràcies al seu gran rendiment i als avantatges que hi ha en tenir tot el *stack* en JavaScript, com són la reutilització de codi i consistència. A més, ja tenia una experiència prèvia amb Node.js, fet que m'ha facilitat la gestió dels fluxes asíncrons mitjançant Promises[9].

5.2.1 Base de Dades

Per poder guardar la informació relacionada amb els videojocs i els usuaris és necessària una base de dades. Com que les dades utilitzades per l'aplicació són repetitives i es poden relacionar entre elles, he decidit utilitzar una base de dades relacional de tipus MySQL [10]. D'aquesta manera, tota la informació queda guardada en forma de taules relacionades entre elles.

Dins la BD he necessitat crear 7 taules diferents (Fig. 6):

- **Users**: Guarda la informació relacionada amb els usuaris: nom d'usuari, mail, contrasenya, foto de perfil, codi de validació i si està validat o no.
- **Games**: Guarda la informació relacionada amb els jocs: nom del joc, la ruta del joc, la foto, el número de *likes*, el número de *dislikes* i si és jugable des del mòbil o no.
- **Categories**: Aquesta taula conté informació sobre les diferents categories que pot tenir un joc: l'ID del joc, el nom de la categoria juntament amb l'ID d'aquesta.
- **Game Info**: En aquesta taula hi ha les instruccions, els controls i el desenvolupador de cada joc en tots els idiomes en què estigui disponible l'aplicació. Aquests paràmetres es guarden en un fitxer JSON.
- **Recent Games**: Aquesta taula és la que emmagatzema els jocs més recents (en format JSON) que ha jugat un jugador (identificat amb un ID).
- **Comments**: Aquí hi ha la informació relacionada amb els comentaris d'un joc: ID del joc, ID de l'usuari que ha comentat, la data quan s'ha realitzat el comentari i el comentari en si.
- **Opinions**: Aquesta taula guarda la valoració que té cada usuari en cada joc que ha jugat, si li ha donat *like*, si li ha donat *dislike* o si l'ha guardat com a preferit.

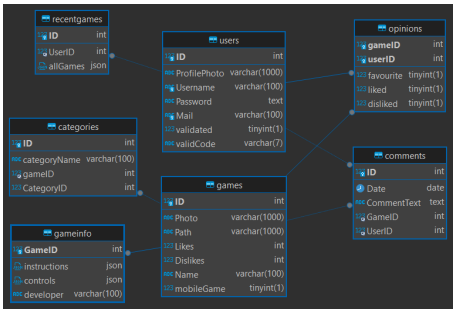

Fig. 6: Les diferents taules mySQL i la seva relació.

[9]Objecte que representa l'eventual finalització o error d'una operació asíncrona.

5.2.2 API REST

Una API (Application Programming Interface) és una peça de codi que permet a diferents aplicacions comunicar-se entre si i compartir informació sense la necessitat de saber com estan implementades, facilitant la integració d'aplicacions i actuant com una capa aïllant.

D'entre tots els tipus d'API que hi ha, he utilitzat una API REST, un tipus d'API que realitza sol·licituds HTTP (GET, POST, PUT, DELETE) per accedir i manipular els recursos a través d'*URI*. És la més adequada per escenaris que necessiten escalabilitat i simplicitat, com són les aplicacions web i mòbils.

Node.js té diferents *frameworks* que permeten implementar aquest tipus d'API en el projecte, com per exemple:

- Hapi.js: Un *framework* conegut pel seu sistema de *plugins* que facilita afegir noves funcionalitats a l'aplicació. Es centra molt en la seguretat, fent-lo ideal per aplicacions que treballen amb dades sensibles.
- Express.js: És el *framework* més famós i més utilitzat gràcies a la seva flexibilitat i facilitat d'ús.
- Nest.js: Un *framework* inspirat en Angular. Utilitza TypeScript i segueix una arquitectura modular que facilita organitzar el codi i escalar l'aplicació.

Pel meu projecte he utilitzat Express.js [11], ja que és el més fàcil d'utilitzar i el més famós de tots, per tant, trobar documentació és molt senzill.

L'API implementada consta d'una sèrie d'*endpoints* (punts on es fan les sol·licituds) on, primer comprovo que els paràmetres rebuts siguin correctes, per evitar possibles atacs com XSS o SQL injection. Posteriorment, realitzo la crida a la BD per obtenir la informació necessària, tracto la informació rebuda i l'envio al *front end* en format JSON.

5.2.3 Seguretat en el *back end*

La ciberseguretat és un concepte molt important a l'hora de dissenyar una web, ja que s'ha de tenir en compte que un ciberatac pot arribar a afectar a milions d'usuaris i comprometre la seva privacitat entre altres problemes [12]. Per exemple, si un atacant aconsegueix injectar codi maliciós en la secció de comentaris, aquell atac es cometria en el dispositiu de qualsevol usuari que accedeixi a la pàgina. Els ciberatacs més freqüents i més perillosos per a una aplicació web són el XSS [13] i SQL injection [14].

Mitigar aquests atacs en el servidor és molt important, encara més crucial que en el *front end*. Ja que si només realitzem la mitigació en la part del client, ens estem exposant a possibles ciberatacs via API o fins i tot que un hacker aconsegueixi sobrepassar les mesures de seguretat del *front end* i realitzar un atac exitós. Per evitar tots aquests possibles atacs he aplicat una sèrie de mesures de seguretat i comprovacions en el *back end*.

- **Prevenció de SQL injection**

  L'atac de SQL injection és un tipus de ciberatac que consisteix a inserir codi maliciós SQL per manipular la base de dades i d'aquesta forma aconseguir informació que no hauria de ser accessible com podrien ser contrasenyes, detalls sobre targetes de crèdit, informació personal dels usuaris, etc.

Un exemple d'atac seria escriure en els camps del formulari *'1 OR 1=1'*. Si la petició (en anglès *query*) a la base de dades és de l'estil *SELECT * FROM users WHERE MAIL = 'mail' AND PASSWORD = 'password'*, la petició a la que se li ha injectat codi SQL quedaría: *SELECT * FROM users WHERE MAIL = "1 OR 1=1" AND PASSWORD = "1 OR 1=1"*. Com es pot veure, les cometes que hem inserit en el formulari s'anul·len amb les que hi ha a la petició, per tant, el codi que hem escrit en el formulari deixa de ser una cadena de caràcters i passa a ser part de la petició. Llavors, podem obtenir la informació dels usuaris si el mail i la contrasenya és '1', o si '1=1'. Com que 1=1, la petició ens retorna tota la informació sobre els usuaris. Hi ha diverses formes de mitigar aquest tipus d'atacs, com per exemple validar els valors d'entrada o utilitzar *queries* parametritzades.

En el meu cas, abans de realitzar la petició cap a la BD valido que els paràmetres que han arribat siguin vàlids i, si no ho són, retorno un missatge d'error. A més a més, per fer les *queries* utilitzo Mysql2 [15], una llibreria de Node.js la qual realitza les peticions de forma parametritzada. Això significa que en lloc de posar el valor escrit per l'usuari directament, com a l'exemple de dalt, utilitza una variable. D'aquesta forma la base de dades és capaç de reconèixer quina part de la petició forma part del codi i quina és l'entrada per l'usuari.

- **Prevenció de XSS**

  El ciberatac de Cross-Site Scripting (XSS) és un altre dels atacs més freqüents sobre les pàgines web i consisteix en la introducció de codi maliciós en una pàgina web (per exemple, en una secció de comentaris) de forma que quan un altre usuari accedeixi a la web, aquest codi s'executi en el seu ordinador.

  Hi ha diversos tipus de XSS:

  1. XSS reflectit: Inserir codi maliciós a través d'un URL
  2. XSS DOM: Manipular el DOM per a modificar el contingut de la pàgina web
  3. XSS emmagatzemat: Inserir codi maliciós i emmagatzemar-lo en una BD

  Un exemple d'atac XSS (emmagatzemat) seria inserir un script maliciós en una secció de comentaris dins una pàgina web, *<script>Codi maliciós...</script>*. Si l'aplicació no ha pres mesures per evitar aquest atac, el codi s'emmagatzema tal qual dins la base de dades. Quan un usuari entri a la web, la base de dades enviarà aquest missatge cap al navegador del client, de manera que s'executarà l'*script* maliciós del comentari dins el navegador de l'usuari.

  Les formes més habituals de mitigar aquest tipus d'atacs són la validació i sanejament de les variables i missatges o utilitzar CSP (Content Security Policy), una capçalera de resposta de les sol·licituds HTTP, acceptada per gairebé tots els navegadors, que permet especificar que *scripts* externs no puguin ser carregats ni executats dins la pàgina.

  En el cas de la meva aplicació, per evitar l'atac del XSS emmagatzemat utilitzo la validació i sanejament de les variables. D'aquesta forma tots els missatges que continguin codi maliciós quedaran sanejats, ja sigui substituint els caràcters maliciosos per uns aleatoris, o eliminant el tros maliciós.

5.2.4 Autenticació d'usuari

Per evitar que els usuaris puguin registrar-se amb correus inexistents, he implementat un procés d'autenticació. Immediatament després que l'usuari s'hagi registrat, des del *back end* envio un correu electrònic amb un enllaç. Quan l'usuari entra a l'enllaç enviat, el seu compte queda validat i ja pot iniciar sessió.

Per aquesta validació primer genero un codi alfanumèric aleatori i guardo les dades de l'usuari juntament amb el codi generat dins la BD, especificant que l'usuari no està validat. Un cop guardades les dades, envio un missatge al correu proporcionat per l'usuari amb un enllaç necessari per la validació. Aquest enllaç consta del nom d'usuari i del codi assignat a l'usuari. Un cop l'usuari hi accedeix, queda automàticament validat.

El missatge de correu l'envio utilitzant la llibreria Nodemailer [16]. Per utilitzar aquesta llibreria primer s'ha de configurar el SMTP especificant quin servidor SMTP volem utilitzar, el port, si volem que la connexió sigui segura i les credencials del correu (Fig. 7). Si s'utilitza el port 587 o 25, es recomana deixar la connexió com a insegura, ja que d'aquesta forma la connexió utilitza el protocol SMTPS (SMTP Secure). En el cas d'utilitzar el port 465 sí que s'hauria de posar la connexió com a segura per poder xifrar el missatge.

Un cop configurat el SMTP, ja podem enviar correus utilitzant Node.js.

```
import nodemailer from 'nodemailer'
import dotenv from 'dotenv'

dotenv.config()
const transporter = nodemailer.createTransport({
    host: "smtp-es.securemail.pro",
    port: 465,
    secure: true,
    auth: {
        user: process.env.EMAIL,
        pass: process.env.EMAIL_PASSWD
    }
})

export { transporter };
```

Fig. 7: Configuració de Nodemailer per enviar correus.

5.3 Integració dels jocs en el portal

Tots els jocs que apareixen en el portal els he dissenyat mitjançant el motor de videojocs Unity[10]. Un cop he finalitzat els jocs els he exportat utilitzant WebGL[11], una API que permet renderitzar gràfics 2D i 3D en qualsevol navegador utilitzant JavaScript. Posteriorment, aquests jocs els he pujat al servidor perquè puguin ser accedits a través d'un *endpoint* mitjançant un URL. Finalment, per poder mostrar

[10] https://unity.com/es
[11] https://es.wikipedia.org/wiki/WebGL

aquests jocs en el *front end*, he necessitat utilitzar *iframes*, un component HTML que permet incrustar el contingut d'una altra pàgina a la pàgina actual.

Tal i com es pot veure a la Fig. 8, el funcionament és el següent:

1. Des de la pàgina del joc (p.e *szymongames.com/games/2048*), es realitza una petició al servidor especificant el nom del joc.
2. El servidor accedeix a l'*endpoint* on està el joc en format WebGL, en aquest cas "/2048".
3. Finalment, dins l'*iframe* es mostra el contingut de la ruta "/2048".

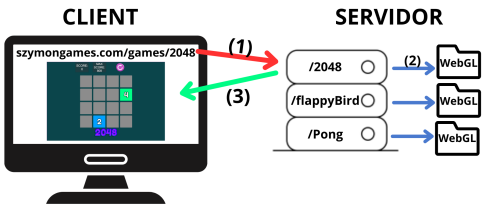

Fig. 8: Esquema d'integració dels jocs en el portal.

6 TESTING

El *testing* és una tècnica que s'utilitza per comprovar que l'aplicació funciona segons l'esperat i que no tingui errors. Però fer aquestes comprovacions de manera manual pot arribar a ser molt tediós. Per aquest motiu m'he decantat per Cypress [17], un *framework* d'automatització de tests per a aplicacions web.

Cypress és una eina que permet dur a terme tant tests unitaris de components com tests End-to-End (E2E) d'una pàgina web. És un *framework* molt fàcil d'utilitzar amb una corba d'aprenentatge molt baixa. Alguns avantatges respecte a altres eines de test són la seva velocitat, les esperes automàtiques i que treballa tant a nivell d'aplicació com de xarxa, per tant ens permet controlar i modificar les peticions HTTP que realitza la nostra pàgina web.

6.1 Tests unitaris

Per aquest projecte he dissenyat una sèrie de tests de components (o unitaris), ja que aquest tipus de test permet testejar els components a fons de forma isolada, fet que facilita trobar possibles errades o *bugs*. Gràcies al fet que Cypress permet interceptar el tràfic HTTP he pogut modificar les respostes per poder testejar qualsevol possible situació sense la necessitat de connectar-me a la base de dades.

A Fig. 9 podem observar un exemple de test amb Cypress. En aquest cas, estic testejant que l'inici de sessió funcioni correctament, i més concretament, el cas en el qual l'usuari escriu la contrasenya incorrecta.

En aquest test, primer inicialitzo la *store* de Pinia per especificar que no hi ha cap usuari autenticat. Seguidament, busco el camp de email i la contrasenya on hi escric els seus respectius valors, per finalment buscar un botó (el d'iniciar sessió) i prémer-lo. Quan he clicat el botó, utilitzo la funció *cy.intercept* per interceptar la petició HTTP enviada i comprovo que el cos de la petició sigui el correcte i indico quina seria la resposta del *back end*. Finalment, esperem la resposta amb el *cy.wait* i comprovem que la resposta no sigui correcta, que el camp de la contrasenya sigui de color vermell i que aparegui un missatge d'error amb el text "Incorrect Password".

```
it ('login with an incorrect password', () => {
    const authStore = useAuthStore();
    authStore.user = null

    cy.get('@email').type(correctMail)
    cy.get('@password').type(incorrectPass)
    cy.get('button').click()

    cy.intercept('POST','/bd/checkLogin', (req) => {
        expect(req.body).to.deep.equal({
            email: correctMail,
            password: incorrectPass
        })

        req.reply({
            statusCode: 418,
            body: {"msg":"IncorrectPass"}
        })
    }).as('login')

    cy.wait('@login').its('response.statusCode').should('eq', 418)
    cy.get('@email').should('have.css','border-color', 'rgb(204, 204, 204)' )
    cy.get('@password').should('have.css','border-color', 'rgb(255, 0, 0)' )
    cy.get('[data-cy="errorMessage"]').should('contain.text', 'Incorrect password!')
})
```

Fig. 9: Test amb Cypress d'un login incorrecte.

6.2 Integració contínua

Executar tots els tests de l'apartat anterior manualment cada vegada que realitzi canvis pot arribar a ser molt tediós. Per aquest motiu he implementat una tasca automàtica per executar-los. Sempre que realitzi canvis en el projecte, aquesta tasca correrà tots els tests de manera automàtica i m'enviarà un correu electrònic amb el resultat final.

Hi ha moltes eines d'integració contínua, entre elles Github Actions, Gitlab Runners [18], Jenkins, Circle CI, etc. Jo personalment m'he decantat pels Gitlab Runners, ja que tenia experiència prèvia utilitzant-los.

Per crear una tasca d'integració contínua al Gitlab només cal crear un *"Job"* dins el projecte i modificar el fitxer .gitlab-ci.yml amb el codi a executar. Un cop realitzat això, la integració contínua ja estarà integrada en el projecte.

En el meu cas en concret, dins el fitxer .gitlab-ci.yml, com es veu a Fig. 10, aixeco una imatge de Docker on correran els tests, i executo les comandes 'npm ci' per instal·lar les dependències, seguit de 'npm cypress run –component' per executar tots els tests unitaris.

```
stages:
  - test

sast:
  stage: test
include:
  - template: Security/SAST.gitlab-ci.yml

component testing:
  image: cypress/browsers:node-20.9.0-chrome-118.0.5993.88-1-ff-118.0.2-edge-118.0.2088.46-1
  stage: test
  script:
    - npm ci
    - npx cypress run --component
```

Fig. 10: Contingut del fitxer .gitlab-ci.yml.

7 *Pentesting*

Un cop he implementat les mesures de seguretat en el *back-end*, toca comprovar que funcionin correctament. Ho he fet mitjançant el *pentesting*, una tècnica que consisteix en

la simulació d'un atac a un sistema Software o Hardware amb l'objectiu de trobar vulnerabilitats per prevenir atacs externs. Com que hi ha moltes maneres diferents de realitzar atacs de XSS o SQL injection, i he de prevenir-les totes, he decidit utilitzar Burp Suite [19], una eina per a realitzar *pentesting* d'aplicacions web tant de forma manual com automàtica.

La forma de realitzar atacs amb Burp Suite és molt senzilla. Per executar atacs automatitzats s'ha d'utilitzar l'opció "Intruder", on hem d'especificar l'URL que volem atacar i la petició HTTP sencera. Dins d'aquesta petició especifiquem quines són les variables sobre les quals el "Intruder" anirà canviant l'input. En el meu cas les variables són els camps del formulari. Posteriorment, indiquem el *payload* (valors que prendran les variables escollides anteriorment). Finalment només caldrà realitzar l'atac!

Pels atacs de SQL injection he utilitzat un *payload* amb gairebé 500 possibles inputs maliciosos extrets d'un repositori de Github[12]. En canvi, pels atacs de XSS, he utilitzat un *payload* amb més de 50 diferents atacs extrets de la web Portswigger[13].

8 Publicació del projecte

Un cop he creat la web i he comprovat que tot funciona correctament, he publicat la web perquè tothom pugui accedir-hi. Per publicar una pàgina web hi ha dues alternatives principals a utilitzar:

1. PaaS (Platform as a Service): Plataforma basada en el núvol per poder crear i executar aplicacions, entre elles trobem Heroku, o Github pages, entre altres.

2. IaaS (Infrastructure as a Service): Model de servei en el núvol que ofereix recursos de computació sota demanda. Els més populars són AWS, Microsoft Azure [20] i Google Cloud Platform.

La principal diferència és que en el cas d'una PaaS, l'usuari puja l'aplicació i aquesta ja queda publicada automàticament sota un domini proporcionat amb una connexió xifrada i segura. En canvi, en el cas d'una IaaS, és el mateix usuari qui ha de crear tots els recursos necessaris i preocupar-se de publicar la web, del domini, els certificats per xifrar la comunicació, etc.

Finalment, m'he decantat a utilitzar un IaaS com és Microsoft Azure, ja que la universitat proporciona crèdits pels estudiants per crear la infraestructura.

8.1 Servidor

El primer (i mínim) que cal fer per poder publicar una aplicació en una IaaS, és crear la màquina virtual (servidor) on pujar el projecte. En el cas d'Azure, abans de crear aquesta màquina, cal crear un grup de recursos, un contenidor lògic que emmagatzema recursos relacionats entre si, com podrien ser xarxes virtuals, màquines virtuals, etc. El següent pas és crear una xarxa virtual per poder assignar-li una direcció IP a la màquina virtual. Quan ja tenim el grup de recursos i la xarxa creada, procedim a crear la màquina virtual i instal·lar totes les dependències i paquets necessaris per fer funcionar l'aplicació.

8.2 Base de Dades

Azure no només permet la possibilitat de crear màquines virtuals, sinó que també permet crear bases de dades escalables i gestionades totalment per Azure. D'aquesta forma, es podria tenir una base de dades separada i connectar-la a una o més màquines virtuals a través d'un *endpoint*, per exemple.

Aquesta era la meva idea inicial pel projecte, però a causa d'un imprevist, m'he quedat sense la meitat dels crèdits i no he pogut crear la base de dades. Per aquest motiu, he creat la base de dades dins la mateixa màquina virtual, fent que no sigui accessible des de fora.

8.3 DNS

Un cop l'aplicació està funcionant dins la màquina virtual d'Azure ja és accessible per a tothom a través de la IP pública de la màquina i el port on corre l'aplicació. Però aquesta forma no és gens segura, ja que no només estem mostrant la IP del nostre servidor exposant-nos a possibles ciberatacs com el DDoS[14], sinó que, a més a més, una persona que no conegui la IP pública mai serà capaç d'accedir a la web. Per això, fa falta associar-li un nom utilitzant el DNS.

El DNS (Domain Name System) és un sistema de nomenclatura jeràrquic i descentralitzat utilitzat pels navegadors que tradueix els noms de domini (*espn.com*, *google.es*, etc.) en direccions IP perquè els usuaris puguin accedir d'una forma molt més senzilla a les webs a través d'un nom de domini i no pas d'una adreça IP. Es podria dir que el DNS és la guia telefònica d'internet. Però perquè un nom de domini sigui vàlid i pugui ser accedit des dels navegadors, s'ha de registrar el domini mitjançant alguna empresa de registre de dominis com Nominalia[15], GoDaddy o Ionos, entre altres. Un cop registrat el domini, només cal especificar la direcció IP on està allotjada l'aplicació. D'aquesta forma, quan l'usuari busqui *szymongames.com*, el navegador demanarà a un servidor DNS i aquest, respondrà amb la direcció IP del domini (Fig. 11).

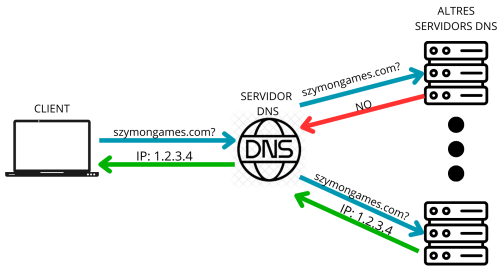

Fig. 11: Esquema sobre el funcionament del DNS.

[12] https://github.com/payloadbox/sql-injection-payload-list/blob/master/README.md

[13] https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

[14] https://www.cloudflare.com/es-es/learning/ddos/what-is-a-ddos-attack/

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

8.4 SSL

Quan ja s'ha desplegat l'aplicació i se li ha assignat un domini, ja només queda xifrar la connexió perquè cap usuari extern pugui veure els missatges entre el client i el servidor i aconseguir informació delicada. Aquest xifratge s'aconsegueix utilitzant el protocol SSL, també conegut com a TLS.

SSL és un protocol de seguretat a internet basat en el xifratge per garantir la autenticitat, privacitat i integritat de les comunicacions d'internet. Per tal de garantir aquestes característiques, al principi de la comunicació, SSL realitza el procés de *handshake*, un intercanvi de missatges on el client i servidor acorden una sèrie de variables (versió del protocol, algoritmes utilitzats, etc.) per després autenticarse utilitzant una sèrie de certificats per poder acordar una clau, només coneguda per ells, que utilitzaran per xifrar els missatges. Quan una connexió estè xifrada, a l'URL apareix el protocol "https" enlloc del "http".

Com he dit abans, en el *handshake* tant el servidor com el client s'han d'autenticar per evitar suplantacions d'identitat i poder xifrar els missatges amb una clau coneguda per ells. Per tant, he de crear uns certificats per autenticar el domini creat anteriorment. Aquests certificats els he generat utilitzant *Let's Encrypt*[16], una autoritat de certificació que proporciona certificats SSL de forma gratuïta.

8.5 Reverse proxy

Un *proxy* és un programa o dispositiu que normalment se situa davant de les màquines del client i fa d'intermediari entre les peticions d'aquest client i un servidor. Aquests dispositius permeten diverses funcionalitats com un control d'accés, anonimitat de comunicació o caché web, entre altres. Hi ha molts tipus de *proxies*, entre ells hi ha *proxies* caché, que guarden el contingut sol·licitat pel client, accelerant les futures respostes a la mateixa petició. També hi ha *proxies* NAT, encarregats de traduir les direccions IP privades dels dispositius en una direcció IP pública perquè el missatge pugui circular per internet.

Per aquesta aplicació, he utilitzat un *proxy* invers (*reverse proxy* en anglès), un tipus de *proxy* que, a diferència de la majoria de *proxies*, se situa davant del servidor redirigint les peticions del client, millorant així la seguretat, el rendiment i la fiabilitat. Els dos servidors *proxy* més populars són Apache[17] i Nginx. En el meu cas m'he decantat per utilitzar Apache.

En la configuració (Fig. 12) he hagut d'especificar dues regles, una pel port 80 i un altre pel port 443. En el cas del port 80 he indicat que faci una redirecció cap al port 443. D'aquesta forma m'he assegurat que la meva aplicació sempre utilitzarà el protocol TLS. D'altra banda, quan la connexió ja està pel port 443 he hagut d'especificar el domini de l'aplicació, els certificats SSL de *Let's Encrypt* generats anteriorment i configurar el *proxy*. Com que l'aplicació funciona pel port 3000, he hagut d'indicar que el contingut del port 3000 me'l redirigeixi al port 443, perquè l'usuari pugui gaudir de l'aplicació de forma segura.

[16] https://letsencrypt.org/es/docs/
[17] https://httpd.apache.org/

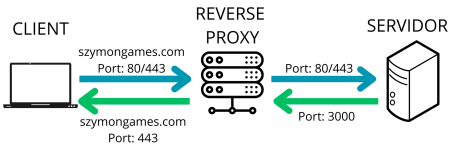

Fig. 12: Esquema de configuració del *reverse proxy*.

9 Discussió dels resultats

En aquest projecte he aconseguit dissenyar un portal de web de jocs en línia (Fig. 13) utilitzant Node.js i Vue. Paral·lelament, he implementat una sèrie de tests unitaris automàtics i de *pentesting* per assegurar el correcte funcionament de la web. Finalment, he publicat aquesta web sota el domini *szymongames.com* utilitzant el proveidor Cloud Azure. Comparant el resultat final amb els objectius proposats inicialment, puc afirmar que tots han estat completats: El portal ha estat dissenyat utilitzant Node.js i Vue. He aconseguit que sigui *responsive* amb l'ús de Media Query. L'inici de sessió i registre es realitzen de forma segura gràcies a l'autenticació i validació per correu electrònic i l'ús del protocol TLS per xifrar la connexió. A més a més, gràcies al *pentesting* amb Burp Suite he previngut els atacs web més freqüents com són el SQL injection i XSS. El sistema d'integració contínua per executar tests unitaris s'ha efectuat executant els tests de Cypress mitjançant els Gitlab Runners. Finalment, utilitzant *cookies* i internacionalitzant la web, he aconseguit que l'experiència de l'usuari sigui positiva.

A l'estat de l'art, he comentat les diferències entre el meu portal de videojocs i els altres. Tal i com he esmentat abans, encara que em falti alguna funcionalitat extra que poden aribar a tenir les webs més famoses, puc quedar molt satisfet amb el treball realitzat. Ja que en aproximadament unes 300h de treball he aconseguit implementar la gran majoria de funcionalitats que ofereix la resta de portals, però a diferència d'ells, en el meu no s'utilitzen *cookies* de tercers ni tampoc hi ha anuncis.

Quan vaig començar aquest projecte no m'esperava aconseguir tots els objectius proposats ni fer tantes funcionalitats com he acabant fent, principalment degut a que és la primera web que he dissenyat. Però gràcies als coneixements bàsics obtinguts en les assignatures de TDIW i STW, juntament amb la gran quantitat de documentació que hi ha a internet, he aconseguit superar les meves expectatives inicials. Fet que demostra que qualsevol enginyer informàtic és capaç de dissenyar una web i publicar-la, independentment dels seus coneixements sobre el disseny web.

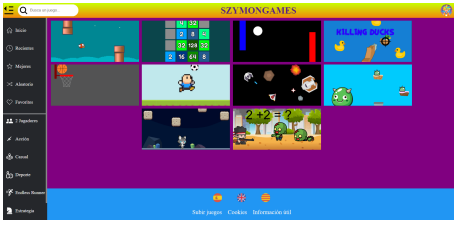

Fig. 13: Página principal de *szymongames.com*.

10 CONCLUSIONS

Després d'analitzar els resultats obtinguts, puc afirmar que aquest projecte ha estat un èxit, ja que tots els objectius han quedat completats. Durant aquest projecte no només he aprofundit molt els meus coneixements sobre el disseny web i les seves funcionalitats com podrien ser les *cookies* o la internacionalització, sinó que també he après com publicar una aplicació amb el seu propi domini públic i utilitzant el protocol TLS dins un entorn Cloud.

Encara que el projecte hagi estat un èxit, no vol dir que no sigui millorable. Hi ha algunes funcionalitats que no he implementat en el meu projecte, ja fos perquè eren molt complexes o perquè he decidit que no eren primordials. La primera d'aquestes funcionalitats seria classificar els jocs en subcategories o afegir etiquetes per facilitar a l'usuari buscar jocs, per exemple: "jocs de policies", "jocs de pingüins", "jocs de bàsquet", etc. Una altre funcionalitat a implementar és el fet de poder afegir amics dins la plataforma i poder jugar amb ells, xerrar i comparar "assoliments". D'aquesta forma es podria augmentar la quantitat d'usuaris i donar-los més motius per entrar a la web i jugar als jocs. Finalment, es podria permetre als usuaris la possibilitat de descarregar-se els jocs i jugar-los de manera local, d'aquesta forma els usuaris podrien gaudir dels jocs inclús sense connexió a internet.

En relació amb aquest últim punt, una possible idea per un futur treball seria convertir l'aplicació de Single Page Application a una Progressive Web Application, per poder accedir-hi tant des del mòbil com des del navegador. D'aquesta forma, els usuaris no haurien d'accedir-hi des del navegador, podrien jugar als videojocs sense connexió, es podrien enviar notificacions des de l'aplicació, entre altres aventatges.

REFERÈNCIES

[1] NewZoo, "Global Games Market Report", Newzoo.com. [En línia]. Disponible a: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version. [Accedit: 01-mar-2024].

[2] AEVI, "La indústria del videojuego en España en 2022". [En línia]. Disponible a: https://www.aevi.org.es/web/wp-content/uploads/2023/05/Anuario-AEVI-2022.pdf. [Accedit: 29-feb-2024]

[3] AEVI, "Power Of Play, Informe Mundial 2023". [En línia]. Disponible a :https://www.aevi.org.es/web/wp-content/uploads/2023/10/POP_Version_09-10-spread.pdf. [Accedit: 29-feb-2024].

[4] J. Martins, "¿Qué es la metodología Kanban y cómo funciona?", Asana, 10-oct-2022. [En línia]. Disponible a: https://asana.com/es/resources/what-is-kanban. [Accedit: 26-feb-2024]

[5] S. B. Uzayr, N. Cloud, y T. Ambler, "Vue.js", 2019. [En línia]. Disponible a: https://vuejs.org/. [Accedit: 27-feb-2024].

[6] "Vue Router", Vuejs.org. [En línia]. Disponible a: https://router.vuejs.org/guide/. [Accedit: 01-mar-2024]

[7] "Pinia", Vuejs.org. [En línia]. Disponible a: https://pinia.vuejs.org/. [Accedit: 08-mar-2024].

[8] "Vue I18n", Intlify.dev. [En línia]. Disponible a: https://vue-i18n.intlify.dev/. [Accedit: 04-may-2024].

[9] "Node.Js", Nodejs.org. [En línia]. Disponible a: https://nodejs.org/en. [Accedido: 27-feb-2024]

[10] "Mysql", Mysql.com. [En línia]. Disponible a: https://www.mysql.com/. [Accedit: 16-feb-2024].

[11] "Express - Node.js web application framework", Expressjs.com. [En línia]. Disponible a: https://expressjs.com/. [Accedit: 02-mar-2024].

[12] C. Otero, "La Generalitat sufre un fallo de seguridad online: 5.000 usuarios expuestos", Meristation. [En línia]. Disponible a: https://as.com/meristation/2020/11/26/betech/1606402985_618977.h [Accedit: 24-jun-2024]

[13] "What is cross-site scripting (XSS) and how to prevent it?", Portswigger.net. [En línia]. Disponible a: https://portswigger.net/web-security/cross-site-scripting. [Accedit: 10-may-2024].

[14] "What is SQL Injection? Tutorial & Examples", Portswigger.net. [En línia]. Disponible a: https://portswigger.net/web-security/sql-injection. [Accedit: 10-may-2024].

[15] "MySQL2", Github.io. [En línia]. Disponible a: https://sidorares.github.io/node-mysql2/docs. [Accedit: 02-mar-2024].

[16] A. Reinman, "Nodemailer", Nodemailer.com. [En línia]. Disponible a: https://nodemailer.com/. [Accedit: 14-abr-2024]

[17] "Testing frameworks for JavaScript", Cypress.io. [En línia]. Disponible a: https://www.cypress.io/. [Accedit: 10-mar-2024].

[18] "Gitlab Runners", Gitlab.com. [En línia]. Disponible a: https://docs.gitlab.com/runner/. [Accedit: 20-mar-2024].

[19] "Burp suite - application security testing software", Portswigger.net. [En línia]. Disponible a: https://portswigger.net/burp. [Accedit: 10-may-2024].

[20] "Servicios de informática en la nube", Microsoft.com. [En línia]. Disponible a: https://azure.microsoft.com/es-es. [Accedit: 20-may-2024].

APÈNDIX

A.1 Manual d'usuari

En aquest apartat explicaré més en detall com funciona la web i les seves funcionalitats.

A.1.1 Pàgina inicial

A l'entrar a la pàgina inicial (Fig. 14), l'usuari es troba amb un llistat de jocs i quan passa el ratolí per sobre d'un d'ells, li apareix el nom del joc. Quan es clica qualsevol imatge, l'usuari accedeix a la pàgina del joc escollit.

A la part superior es troba el *header* on hi ha (d'esquerra a dreta) el botó per amagar i mostrar el menú lateral i una barra de cerca per poder cercar els jocs pel nom. La barra de cerca mostra tots els jocs que contenen la seqüència de caràcters escrita, i si no hi ha cap, no mostra cap joc. Seguidament, hi ha el nom de la web, que quan es clica redirigeix a la pàgina inicial i el botó de perfil/*log in*, en funció de si s'ha iniciat sessió o no.

A la part inferior es troba el *footer* on es pot veure diferents banderes per canviar l'idioma de la web i a sota, una sèrie de botons:

- Afegir jocs: Aquest botó porta a l'usuari a un formulari de Google on pot passar-me el seu joc perquè jo el pengi a la plataforma
- *Cookies*: Es torna a mostrar el missatge sobre les *cookies* utilitzades
- Informació d'interès: Mostra un text amb informació sobre la web

Finalment, a la part esquerra ens trobem amb el menú lateral que ens mostra les diferents categories.

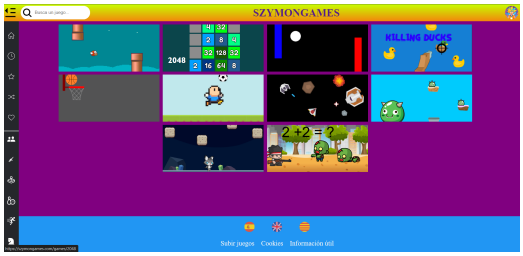

Fig. 14: Pàgina principal de *szymongames.com*.

A.1.2 Pàgina de joc

L'usuari després d'escollir quin joc vol jugar, queda redirigit cap a la pàgina del videojoc (Fig. 15), on veu (d'esquerra a dreta), un menú on es mostra tota la informació relacionada amb el joc: el nom, descripció, controls en funció de si està jugant des del mòbil o ordinador, l' *iframe* amb el joc, i una llista de jocs similars a l'actual. Quan es prem qualsevol joc similar, la pàgina et redirigeix cap a l'altre joc.

Sota de l'*iframe* hi ha la barra de valoració, on a part del nom del joc, hi ha 3 botons: els botons de "m'agrada" i "no m'agrada" amb la quantitat de vots al costat i el botó de "preferit" representat amb un cor. L'usuari només pot interactuar amb aquests botons si ha iniciat sessió, sinó, li apareixerà una finestra per iniciar-la. Quan l'usuari ha valorat qualsevol joc o l'ha afegit a preferits, llavors el botó seleccionat es torna de color negre.

Finalment, a sota de tot es situa la secció de comentaris. Els comentaris estan ordenats de més recent a més antic, i en cada comentari apareix el nom de l'usuari que l'ha escrit, juntament amb la seva foto de perfil, el dia que ha escrit el comentari i el missatge escrit. Igual que amb les valoracions, només es pot escriure comentaris si s'ha iniciat sessió.

En el cas que l'usuari estigui amb un mòbil en vertical, veuria primer l'*iframe* amb el joc i les valoracions, i si continués baixant es trobaria seguidament amb el menú d'instruccions, els jocs similars i finalment amb la secció de comentaris.

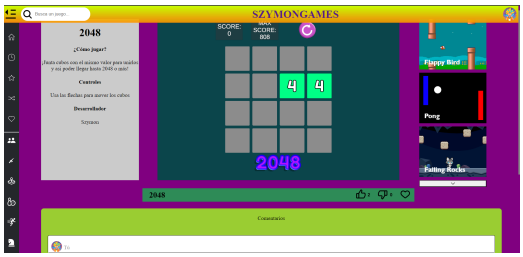

Fig. 15: Pàgina per jugar al joc "2048".

A.1.3 Inici de sessió i informació sobre el perfil

Tal i com s'ha comentat en l'apartat de "Pàgina inicial", a la part superior esquerra de la pantalla, en el *header*, es situa una icona que varia en funció de si s'ha iniciat sessió o no. El que no canvia en cap cas, és que al clicar-la, ens apareix una finestra emergent amb dues pestanyes.

Si l'usuari no s'ha registrat (Fig. 16), podrà escollir entre les pestanyes d'"Inica sessió" o "Registra't". Les dues opcions contenen un formulari que l'usuari ha d'omplir amb les seves dades. En el cas d'iniciar sessió, just sota el formulari hi ha un text, "Has oblidat la contrasenya?", que quan es prem porta a un altre formulari, però en aquest cas per restaurar la contrasenya.

Fig. 16: Finestra emergent quan l'usuari no ha iniciat sessió.

D'altra banda, si l'usuari ha iniciat sessió (Fig. 17), una de les pestanyes de la finestra emergent és "Informació de l'usuari" on es pot veure el nom d'usuari, el correu electrònic i també ofereix la possibilitat de canviar la foto de perfil clicant la imatge actual. L'altre pestanya és "Canviar la contrasenya", que ens portarà al formulari per restaurar la

contrasenya, però en aquest cas, el camp del correu ja està omplert amb el correu de l'usuari i no es pot modificar.

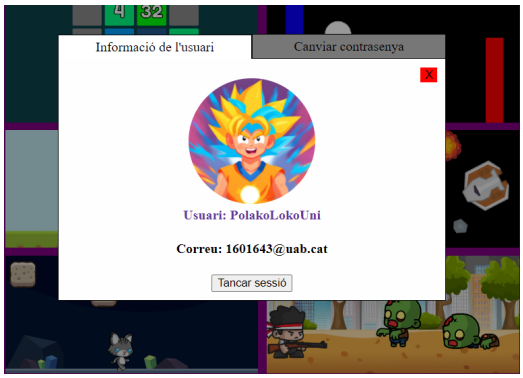

Fig. 17: Finestra emergent quan l'usuari ha iniciat sessió.

A.1.4 Menú lateral - Categories

A la part esquerra de la pantalla, hi ha el menú amb les diferents categories a les que pertanyen els videojocs (Fig. 18). Aquest menú consta de dues parts separades per una línia blanca.

La primera part, consta d'una sèrie d'opcions que poden ajudar a l'usuari a decidir quin joc jugar. Les opcions són les següents:

- Inici: Retorna a la pàgina inicial
- Últims jugats: Mostra els últims jocs jugats per l'usuari. Si ha iniciat sessió, mostra els últims jocs jugats des del seu compte, si no ha iniciat sessió, els jocs jugats més recentment des del dispositiu utilitzat.
- Millors: Aquesta opció mostra els jocs millor valorats en funció de la diferència entre *likes* i *dislikes* del joc.
- Aleatori: Aquest botó redirigeix a l'usuari a jugar un joc aleatori.
- Preferits: Mostra tots els jocs que l'usuari té marcats com a preferits. Si no s'ha iniciat sessió, aquesta opció no és accessible.

Seguidament, hi ha la segona part del menú, que mostra en ordre alfabètic totes les categories a les que els jocs pertanyen. Al prémer qualsevol d'aquestes categories, la pàgina mostra només els jocs que pertanyen a certa categoria.

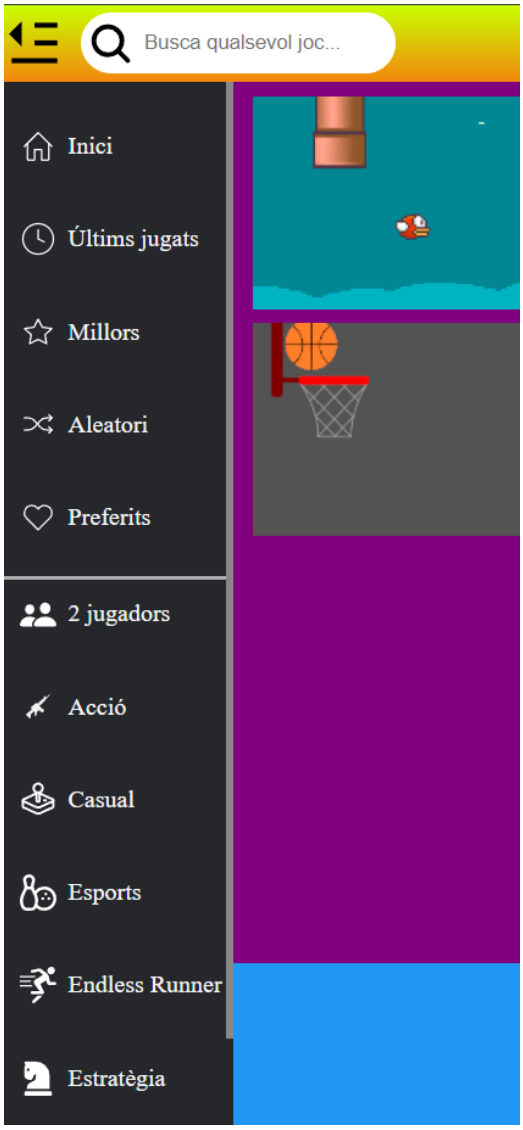

Fig. 18: Menú lateral desplegat.

A.2 *Endpoints* utilitzats

Com ja he esmentat en aquest treball, l'API REST utilitza una sèrie d'*endpoints* als quals accedeix el *front end* per aconseguir la informació del servidor. Els *endpoints* són de tipus GET, POST, PUT.

A.2.1 *Endpoints* de tipus GET

El mètode GET s'utilitza per recuperar i aconseguir informació de la base de dades a través de l'URL. Els *endpoints* que utilitzen el mètode GET són els següents:

- **/bd/getGames**: Aquest *endpoint* envia la llista de tots els jocs disponibles.

- **/bd/getSimilarGames/:id**: S'utilitza aquest *endpoint* per aconseguir la llista de jocs similars a l'actual amb l'ID enviat per paràmetre.
- **/bd/comments/:id**: Retorna una llista amb tots els comentaris del joc amb l'ID especificat.
- **/bd/:name**: Envia tota la informació sobre el joc indicat.
- **/getCategories**: Amb aquest *endpoint* s'aconsegueix una llista amb els noms de totes les categories.
- **/bd/categories/:name**: Retorna tots els jocs que pertanyen a la categoria especificada en el paràmetre "name"
- **bd/info/:name/:lang/:isMobile**: Amb aquest *endpoint* s'envien les instruccions i controls del joc especificat per paràmetre en l'idioma indicat ("lang"). El paràmetre "isMobile" s'utilitza per saber si mostrar els controls de mòbil o d'ordinador.
- **/bd/getOpinions/:id/:userId**: Retorna les valoracions (*like*, *dislike*, preferit) d'un usuari en un joc concret.
- **/bd/getFavGames/:username**: Mostra els jocs afegits a preferits d'un usuari.
- **/bd/getRecentGames/:usernameID**: Mostra els jocs recents d'un usuari.
- **/bd/addDefaultOpinions/:id/:user**: Aquest *endpoint* inicialitza els valors de *likes*, *dislikes* i preferits d'un usuari en un joc on entra per primera vegada. Tant el paràmetre "id" com "user" són valors numèrics.

A.2.2 *Endpoints* de tipus POST

El mètode POST s'utilitza per enviar dades al servidor. A diferència del mètode GET, aquest no viatja a través de l'URL, sinó que té un *body* on s'especifica la informació, per tant és ideal per enviar dades sensibles, que no es vol que ningú vegi. Els *endpoints* que he utilitzat amb el mètode GET són:

- **/bd/checkLogin**: Aquest *endpoint* compara el correu i contrasenya enviats amb els valors emmagatzemats a la base de dades. Si les dades són correctes, s'envia el nom d'usuari, correu, foto de perfil i ID al *front end*, i si no són correctes, un missatge d'error,
- **/bd/insertUser**: Afegeix un usuari nou a la base de dades, indicant que encara no està validat. Després, genera un codi aleatòri, l'emmagatzema a la base de dades i envia un correu a l'usuari per validar el compte.
- **/bd/sendCode**: Genera un codi aleatòri de 7 digits i envia un correu electrònic a l'usuari per canviar la contrasenya.
- **/bd/changePassword**: Comprova que el codi enviat és correcte i que l'usuari realment ha demanat canviar la contrasenya. Si les dues comprovacions són certes, es canvia la contrasenya de l'usuari, sinó s'envia un missatge d'error.
- **/bd/validateUser**: Aquest *endpoint* valida que el codi enviat és correcte i que l'usuari no està validat. Si és així, es valida a l'usuari i se li crea un llistat buit de jocs recents que s'anirà actualitzant a mesura que l'usuari jugui a jocs.
- **/bd/insertComment**: Primer es saneja el comentari, per evitar emmagatzemar codi maliciós, i després es guarda el comentari dins la base de dades indicant l'usuari que l'ha escrit, en quin joc, la data i el missatge enviat.

A.2.3 *Endpoints* de tipus PUT

El mètode PUT s'utilitza per afegir un nou element a la base de dades o actualitzar/substituir un d'existent. És molt similar al POST, l'única diferència és que el mètode PUT és idempotent, enviar-lo una o varies vegades té el mateix efecte, en canvi el mètode POST no ho és. Els *endpoints* utilitzats amb PUT són:

- **/bd/setRecentGames**: Actualitza els jocs recents d'un usuari guardats dins la base de dades amb un nou llistat enviat en el *body*.
- **/bd/changeValoration**: Actualitza els valors de *likes*, *dislikes* i preferits d'un usuari en un joc específic.
- **/bd/changeProfilePicture**: Canvia la foto de perfil d'un usuari i guarda dins la base de dades quina foto és.