Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Hernández Navarrete, Adrián; Robles Martínez, Sergi, dir. Desenvolupament d'un Mission Control Web per a un projecte de Realitat Virtual per a la formació de professorat. 2023. (Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/290108

under the terms of the BY-NC-ND license

Desenvolupament d'un Mission Control Web per a un projecte de Realitat Virtual per a la formació de professorat

Adrián Hernández Navarrete

Resum— A aquest treball es desenvolupa una aplicació web que es comunica amb un joc de Realitat Virtual creat pel projecte de recerca Didascalia orientat a la formació de professorat. Aquesta aplicació permet a l'usuari monitorar l'estat del joc i, a més, interactuar amb les diferents simulacions en curs. D'aquesta manera es millora substancialment l'experiència de l'usuari i la interacció amb la simulació des de l'exterior. També es desenvolupa una estructura de servidors backend pel seu funcionament.

Paraules clau— Mission Control, Didascalia, simulació, web, aplicació, VR, professorat, monitorar, formació, estudiant, interactuar

Abstract— In this work, a web application that communicates with a Virtual Reality game created by the Didascalia research project oriented to teacher training is developed. This application allows the user to monitor the state of the game and, in addition, to interact with the ongoing simulations. This substantially improves the user experience and the interaction with the simulation from the outside. A backend server structure is also developed for its implementation.

Index Terms— Mission Control, Didascalia, simulation, web, application, VR, faculty, monitor, training, student, interact

1 Introducció - Context del treball

L'ENSENYAMENT és un concepte molt important a la societat i es considera la base de qualsevol país, així com un dels pilars més importants de la societat contemporània. A Espanya, per poder exercir com a docent en un centre d'ensenyament, es requereix haver realitzat i aprovat el màster de formació de professorat [1]. Dins d'aquests estudis hi ha una assignatura on l'estudiant del màster ha de practicar davant d'una classe real, ja que és l'única manera de provar les capacitats assolides.

Ara bé, posar a l'estudiant del màster directament a una classe d'institut no només és impactant, sinó que també pot arribar a ser contraproduent per a l'estudiant que està atenent a classe amb un professor inexpert.

Davant d'aquesta situació, es va crear un projecte de recerca on col·laboren la UAB i la UCM, Didascalias [2], que permet crear un entorn virtual amb unes ulleres de realitat virtual que simula una aula amb una sèrie d'alumnes amb diferents personalitats. Dins d'aquesta simulació un algorisme d'IA dona vida als estudiants. També hi ha una sèrie d'esdeveniments, com ara que un alumne insulti o hi hagi una baralla. D'aquesta manera es poden posar a prova les competències de l'estudiant del màster.

No obstant, hi ha una sèrie de problemes. El projecte és unidireccional, és a dir, situa a l'estudiant del màster contra l'algorisme, deixant de banda aspectes com la interacció entre alumne i professor del màster i consulta dels diferents paràmetres de la simulació en temps real. Seria convenient que el professor del màster pugui interactuar amb la simulació i executar funcions que generen esdeveniments dins la simulació per a posar a prova a l'estudiant i, a més, cal que es pugui anar consultant en temps real l'estat del joc i la reacció de l'estudiant del màster.

A partir d'aquí sorgeix la necessitat de desenvolupar un entorn digital que mostri de manera visual estadístiques i dades en temps real d'una simulació en curs. També cal que permeti visualitzar un petit resum de la simulació i oferir al professor del màster una interacció real amb la simulació en curs.

1.1 Objectius

Un cop exposada la situació, ens proposem desenvolupar un Mission Control, una aplicació que ens permetrà visualitzar en temps real diverses estadístiques d'una simulació en curs i, per altra banda, interactuar amb aquesta simulació.

Es divideix en els següents sis objectius, enumerats en ordre de prioritat i importància:

- **O1: Analitzar els requisits**

Cercar les necessitats dels usuaris, estudiar els diferents casos d'ús i establir els requisits de l'aplicació.

- **O2: Dissenyar l'arquitectura i les interfícies del Mission Control**

Dissenyar l'estructura *backend* necessària per a fer funcionar el Mission Control. Això fa referència als servidors *backend*, la seva estructura i tecnologies i a la definició dels protocols de comunicació entre aquests.

- **O3: Desenvolupar el *frontend* del Mission Control**

Desenvolupar la part visual del Mission Control, configurant l'entorn de desenvolupament i fent funcionar un servidor per a que ofereixi a la Internet l'aplicació web.

- **O4: Desenvolupar els *backend* del Mission Control**

Desenvolupar la lògica *backend* amb els endpoints necessaris fer per funcionar les peticions del *frontend*.

- **O5: Integració i proves del projecte**

Preparar i integrar l'aplicació amb l'estructura i les necessitats del projecte Didascalias, així com realitzar proves per assegurar el bon funcionament de l'aplicació web.

- **O6: Documentació del projecte**

Per tal de que es pugui entendre amb més facilitat aquesta extensió del seu projecte, s'hauran de documentar les funcionalitats del codi i realitzar una petita formació als programadors de Didascalias per facilitar el manteniment i el relleu del sistema desenvolupat.

La resta del document s'estructura de la següent manera: la metodologia, que conté una breu explicació de com s'ha planificat i organitzat; l'estat de l'art, on s'explica quines tecnologies, frameworks i paradigmes de comunicació s'han fet servir; el desenvolupament, on queda constància de les tasques realitzades i els diferents *sprints*; els resultats, on s'exposa en termes de rendiment què s'ha desenvolupat; les conclusions i, finalment, la bibliografia i l'apèndix.

2 METODOLOGIA

La metodologia seguida al desenvolupament del projecte està basada en el model espiral dividida en diversos *sprints* [3]. El procés de desenvolupament es representa en una espiral on cada volta representa un *sprint*. Dins de cada *sprint* hi ha quatre etapes: definició d'objectius del *sprint*; anàlisi de riscos i planificació; desenvolupament i proves i, finalment, avaluació i planificació per al següent *sprint*.

A la fase de planificació s'escullen els requisits que es realitzaran durant el *sprint* corresponent, assignant-los un cost determinat en forma de temps. A continuació, a la fase de desenvolupament s'implementen els requisits. Seguidament, es realitzen les proves corresponents per tal de que la implementació sigui correcta segons els requisits definits anteriorment. En acabat, es deixa quelcom preparat pel següent *sprint*.

Aquesta metodologia en aquest cas ens servirà perquè l'aplicatiu que es desenvolupa s'apropi molt més a l'esperat i, a més, es tingui en compte en tot moment quins són els objectius i les tasques que s'han de realitzar per a arribar als objectius proposats. També el fet de realitzar *sprints* permet adaptar-se amb més facilitat als canvis, la qual cosa és molt positiva tenint en compte que el projecte Didascalias també es troba en desenvolupament.

En aquest cas, es realitzaran un total de cinc *sprints*. Un primer orientat a disseny i requisits; el segon, tercer i quart en el desenvolupament i; finalment, el cinquè, enfocat al tancament.

2.1 Planificació

Per començar, extrau de cada objectiu una sèrie de tasques. A continuació, es lliga cada tasca a un cost temporal i, finalment, es mostra el diagrama de Gantt corresponent.

Primer de tot, desgranem el primer objectiu O1 que té la tasca T1,"Anàlisi de requisits", i la tasca T2, "Creació del prototipus". A continuació, l'objectiu O2 té la tasca T3, "Dissenyar l'arquitectura", i la tasca T4, "Dissenyar els endpoints i estructures de comunicació". L'objectiu O3 conté la tasca T5, "Oferir el *frontend* a la Internet", i la tasca T6, "Programar el *frontend*". L'objectiu O4 conté la tasca T7, "Dissenyar els endpoints del backend", la tasca T8, "Oferir el *backend* a la Internet" i la tasca T9, "Programar el *backend*". L'objectiu O5 conté la tasca T10, "Integració amb Didascalias", i la tasca T11, "Mesura del rendiment". Finalment, l'objectiu O6 conté la tasca T12, "Creació de documentació", i la tasca T13, "Formació a Didascalias".

O1	T1	Anàlisi de requisits
	T2	Creació del prototipus
O2	T3	Dissenyar l'arquitectura
	T4	Dissenyar els endpoints i estructures de comunicació
O3	T5	Oferir *frontend* a la Internet
	T6	Programar el *frontend*
O4	T7	Dissenyar els endpoints del *backend*
	T8	Oferir el *backend* a la Internet
	T9	Programar el *backend*
O5	T10	Integració amb Didascalias
	T11	Mesura del rendiment
O6	T12	Creació de la documentació
	T13	Formació a Didascalias

A cada tasca se li ha assignat un cost en forma de temps. Al diagrama de Gantt a l'apèndix A es pot apreciar la quantitat de setmanes que ocupa cadascuna i aquesta associació entre objectius i tasques.

Les primeres tasques T1, T2 i T3 ocupen un mes cadascuna perquè s'ha de determinar amb precisió els diferents requisits, a més de triar una arquitectura idònia. La tasca T4 és prou llarga, atès que es contempla anar dissenyant les estructures i tecnologies de comunicació segons l'evolució del projecte Didascalias, que s'està desenvolupant en paral·lel. Les tasques T5 i T8 són gairebé idèntiques i curtes, així que no cal més de dues setmanes. Pel que fa a la programació del *frontend*, tasca T6, és la més llarga de totes, doncs és la tasca troncal del projecte. La tasca T7 es realitza en paral·lel al disseny de les estructures de comunicació i tecnologies. La tasca T9 de programació del *backend* també és prou llarga, atès que és una altra de les tasques essencials del projecte. La tasca T10 ha de ser llarga per naturalesa, doncs allò que es desenvolupa ha d'anar integrant-se contínuament amb el projecte Didascalias. Finalment, les tasques T11, T12 i T13 són curtes i conformen el tancament de projecte.

- *E-mail de contacte: adrian.hernandezn@autonoma.cat*
- *Menció realitzada: Tecnologies de la Informació*
- *Treball tutoritzat per: Sergi Robles Martínez (DEIC)*
- *Curs 2023/24*

3 Estat de l'Art

A continuació s'expliquen les tecnologies emprades i quins beneficis presenten. Primerament, allò relacionat amb Didascalias. Segonament, tecnologies del Mission Control. Seguidament, es tractarà el *backend* i, finalment, els protocols i tecnologies de comunicació.

Primer de tot, s'ha de conèixer l'entorn de treball de Didascalias per a poder realitzar la integració correctament. Fan servir les ulleres de Realitat Virtual Oculus[1], amb el motor gràfic Unity[2], que es pot fer servir gratuïtament, i programat amb un llenguatge declaratiu C# [4].

D'altra banda, es planteja que el Mission Control pugui ser o una aplicació d'escriptori [5] o una aplicació web [6]. L'aplicació d'escriptori es pot executar només als ordinadors i són bastant ràpides, ja que es comuniquen directament amb el Sistema Operatiu. En canvi, l'aplicació web funciona al navegador, la qual cosa ens permet fer-la servir a diferents dispositius com PC, tauletes, mòbils, etc.

Si ens fixem en les aplicacions web, hi ha la opció de fer servir VueJS [7] com a framework de desenvolupament del *frontend*, atès que és ràpid, lleuger, responsive i en tinc coneixements com a programador. També existeix la opció de fer servir Angular i React [8], que també tenen característiques similars i són molt populars.

Pel que fa al *backend* trobem NodeJS [9] que ofereix una extensa quantitat de llibreries i permet gestió asíncrona. També podem trobar ASP.NET [10], basada principalment en C# i amb facilitats per gestionar la seguretat i la escalabilitat; i Lavarel [11], basada en PHP i considerada fàcil d'usar i amb una sintaxi elegant.

Per a poder emmagatzemar dades o credencials es pot fer servir una base de dades. Una de les opcions tradicionals i més popular és MySQL, a més de ser de les més ràpides. Una segona opció és PostgreSQL, sent la més utilitzada a dia d'avui [12].

També és important comentar les dues tecnologies de comunicació que es podrien fer servir: WebSocket [13], que ens permet una comunicació orientada a connexió, de tal manera que podem tenir un flux de dades constant, segur, ràpid i fluït entre dos dispositius; i Rest API [14], que es basa en serveis web mitjançant el protocol HTTP i que es pot escalar amb facilitat. El WebSocket serà útil per a la comunicació entre ulleres i *backend*, i Rest API per a les crides que facin els diferents navegadors que estiguin executant l'aplicació web.

La comunicació entre els diferents actors de l'estructura pot ser en format JSON [15]. Es tracta d'un estàndard bastant comú, la qual cosa a tots els llenguatges de programació existeixen llibreries que permeten fer-ho servir amb molta facilitat.

Totes les tecnologies que s'han mencionat són molt populars i es troben entre les més utilitzades dins del món del desenvolupament de software segons l'enquesta anual de StackOverflow[3].

4 Desenvolupament

A continuació s'explica com s'ha desenvolupat cada *sprint* amb els seus punts més rellevants.

4.1 Primer *sprint*: disseny inicial

Per començar, el primer *sprint* és el més important atès que és quan es posen sobre la taula els objectius, tasques i requisits a realitzar durant tot el projecte. També es prenen decisions importants de disseny, es crea una primera idea d'estructura del sistema i es desenvolupa un primer prototip molt primitiu. S'iniciarà a la setmana 1 i s'acabarà a la setmana 4.

He contemplat dur a terme les tasques T1, d'anàlisi de requisits; i T2, de creació de prototipus. D'aquesta manera s'ha dissenyat un prototip que ha servit com a punt de referència i que ha estat acordat amb els companys del projecte de Didascalias.

Alhora, es va començar a dissenyar l'arquitectura (T3), les estructures de comunicació (T4), els endpoints del *backend* (T7); l'obertura a la Internet tant del servidor web (T5), com del Servidor de Joc (T8). En aquestes tasques de disseny s'han pres decisions importants, com ara quines tecnologies s'escolliran.

4.1.1 Decisions de disseny

Es va decidir que la comunicació entre el client que corre l'aplicació al navegador i el servidor es realitzi mitjançant Rest API. Això ens ha permès poder establir *cookies* de sessió [16] per a la gestió de credencials. A més, existeixen nombroses llibreries que faciliten la gestió de *cookies* i peticions.

Pel que fa al *backend*, que li anomenarem "Sevidor de Joc" d'aquí en endavant, es va decidir fer servir NodeJS i una base de dades MySQL. NodeJS té moltes llibreries útils i populars que agilitzen el desenvolupament, a més de basar-se en JavaScript que ofereix molta flexibilitat i facilitats a l'hora de programar. MySQL es va triar per ser l'opció més popular i la utilitzada com a tria per defecte de molts desenvolupadors.

D'altra banda, per al web, es va decidir fer servir VueJS. Ofereix també molta flexibilitat en basar-se en JavaScript, i a més com a desenvolupador en tinc bastant coneixement.

4.1.2 Estructura del sistema

Durant l'anàlisi de requisits va sorgir la idea de definir els diferents casos d'ús en forma d'escenaris on es mostra com l'usuari final podria interactuar de diferents maneres amb el sistema a desenvolupar.

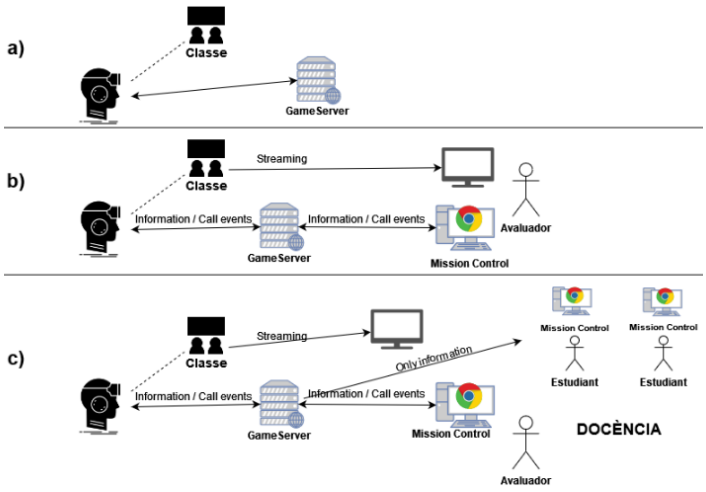


Figura 1. Escenaris de casos d'ús.

L'escenari a) de la figura 1 mostra com l'usuari fa servir les ulleres VR per a practicar amb la simulació o preparar-se per a les proves o exàmens del màster.

Seguidament, l'escenari b) mostra com l'usuari amb les ulleres VR retransmet en temps real dades i estadístiques cap un servidor que, alhora, es comunica amb la pàgina web, el Mission Control, on l'avaluador monitora l'estat del joc i crea situacions; tot això mentre visualitza per *streaming* el punt de vista de l'usuari, que és quelcom extern al Mission Control. D'aquesta manera l'avaluador pot fer un seguiment més acurat del que passa a la simulació i pot posar a prova a l'estudiant amb les ulleres VR.

Finalment, l'escenari c) mostra el mateix cas anterior però afegint més espectadors que poden veure el *streaming* de les ulleres VR i accedir al Mission Control per a monitorar les dades, però sense poder crear situacions ni interactuar amb el joc, de tal manera que l'avaluador pot aprofitar per a impartir docència als espectadors basant-se en una simulació en curs.

4.1.3 Primer prototip base

A continuació, es va desenvolupar una versió inicial del lloc web (tasca T6) amb un sol botó, que envia una petició cap al Servidor de Joc i aquest respondrà (T9) amb les dades conseqüents. El navegador del client, en rebre resposta, calcula la latència i la mostra per pantalla.

Com es pot veure a la figura 2, el client mitjançant connexió a la Internet, es connecta al servidor web basat en VueJS i rep els continguts de la pàgina web. Seguidament, les peticions que realitzi l'aplicació des del navegador del client s'envien directament al Servidor de Joc, que corre una aplicació de NodeJS, i aquest respon depenent del tipus de sol·licitud.

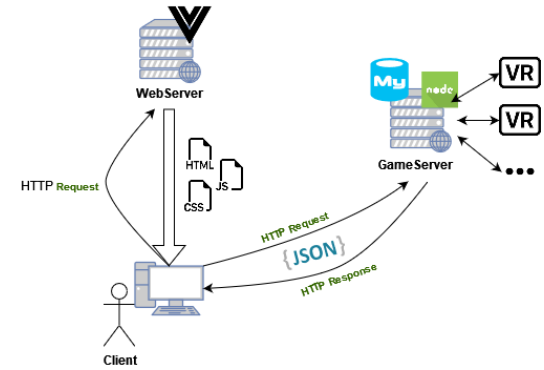


Figura 2. Estructura de servidors i comunicacions.

A més, el Servidor de Joc es connectaria amb les diferents Oculus VR; però això es dissenyarà en detall i s'implementarà als següents *sprints*.

4.2 Segon *sprint*: web inicial

A aquest segon *sprint* es va definir completament l'arquitectura (tasca T3), es va fixar un prototip final (T2) i es va començar a desenvolupar els primers apartats del lloc web (T6) i del *backend* (T7 i T9) basant-se en el prototip.

4.2.1 Definició del prototip final

El prototip final es crea amb Microsoft Whiteboard[1] de tal manera que, sense haver de programar les funcionalitats, es pot apreciar com hauria de quedar la interfície d'usuari del lloc web i quines utilitats i accions estaran disponibles. Tot plegat, permet acordar i especificar amb facilitat i claredat el sistema a desenvolupar amb els programadors del projecte Didascalias.

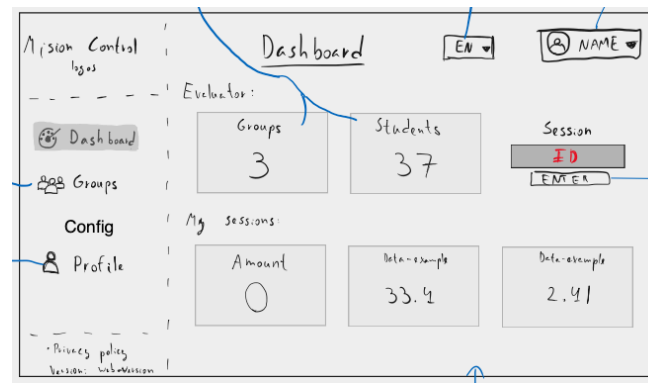


Figura 3. *Dashboard* segons el prototipus.

Aquest prototip contempla l'aspecte dels següents apartats: pàgina d'inici de sessió amb credencials, que permetrà fer inici de sessió als usuaris registrats o entrar a visualitzar una sessió amb el número de sessió; un *dashboard* o centre de control, on només es podrà accedir si s'inicia sessió correctament i mostrarà la quantitat de grups d'alumnes assignats, quantitat d'alumnes, les teves sessions, algunes dades per determinar sobre les sessions i un camp per veure una sessió en curs; una pantalla de visualització dels grups i alumnes, amb el seu tutor corresponent i els e-

[1]*El web de venda de les Oculus és https://www.oculus.com/rift-s/*
[2]*El web del motor gràfic d'Unity és https://unity.com*
[3]*Pots accedir a l'enquesta de StackOverflow a https://survey.stackoverflow.co/2023/#overview*

[1]*Per saber més detalls sobre què és es pot consultar el següent web: https://www.microsoft.com/es-es/microsoft-365/microsoft-whiteboard/*

mails; una finestra de perfil per veure i modificar les teves dades i, finalment i el més important, el Mission Control.

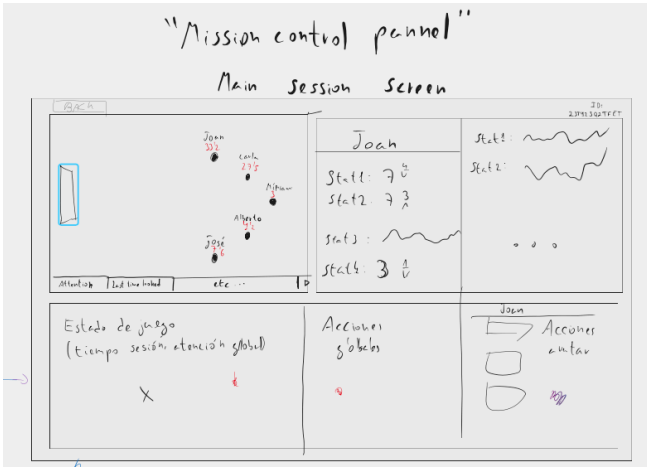


Figura 4. Mission Control segons el prototipus.

Com es pot veure a la figura 4., el Mission Control és on es visualitzen les sessions, i es divideix en quatre apartats. El primer permet veure la disposició dels alumnes virtuals de la simulació; al segon es visualitzen estadístiques d'un alumne virtual que s'hagi seleccionat al mapa; el tercer mostra dades globals, com ara el moviment del jugador, el temps de la sessió, etcètera; i un apartat on es poden cridar accions globals o d'un alumne virtual, com ara que passi alguna cosa a la classe o que l'alumne virtual seleccionat insulti.

Un cop definida tota la interfície, van quedar també marcades les expectatives del sistema a desenvolupar i es va poder continuar amb el desenvolupament basant-se en aquest prototip.

4.2.2 Desenvolupament del portal de benvinguda

A partir d'aquí, es va començar a desenvolupar la primera interfície: la pantalla d'inici de sessió. Aquesta, segons el prototip, hauria de contenir els logotips de les universitats que participen del projecte, un apartat per introduir la sessió a la que es vol entrar i un altre apartat on introduir les credencials de l'usuari. Els botons d'inici de sessió i d'entrar a la sessió haurien d'anar connectats a una funció que s'encarregui de comunicar-se mitjançant l'API amb el servidor de joc.

El servidor de joc, per la seva part, ha de comprovar i retornar una resposta. Es comprova o bé que la sessió existeix, o bé que les credencials siguin correctes. Com s'ha comentat anteriorment, hi ha una base de dades MySQL on es guarden les credencials dels usuaris en forma de *hash* [17]. Més endavant es va dissenyar i programar la gestió de *cookies* i com establir un sistema d'autorització per a usuaris-

En definitiva, l'*sprint* va finalitzar dins del termini indicat, amb un prototip definitiu clar i amb un sistema d'inici de sessió simple i segur.

4.3 Tercer *sprint*: implementació del *backend*

Al tercer *sprint* he contemplat acabar de dissenyar els endpoints i les estructures de comunicació (tasca T4), així com haver començat la integració amb Didascalias (T10), continuat el disseny i la programació del *backend* (T7 i T9) i seguir desenvolupant el *frontend* (T6).

Fou molt important començar a introduir les ulleres VR de Didascalias com a actor funcional del sistema, doncs són imprescindibles pel projecte i calia anar preparant i programant-les.

4.3.1 Introducció de les ulleres VR

Per a poder monitorar correctament la simulació fou important establir una comunicació ràpida i constant entre les ulleres VR i Servidor de Joc per recollir i gravar dades. Per això, es va determinar que aquesta comunicació funcionarà mitjançant WebSocket.

Per tal d'unificar criteris, es va acordar amb Didascalias un format de missatges amb JSON amb una estructura determinada i diferents tipus de missatges. Aquest format s'ha decidit repetir a tots els missatges que s'intercanvien les ulleres VR i el servidor de joc:

```
{
   type: 0,
   session: 0,
   data: null,
}
```

Es va triar aquest format perquè és fàcil de manipular i llegir: al *type* s'especifica el tipus de missatge; al camp *session* sempre s'inclou el numero de sessió i; el camp *data* inclou, lògicament i si escau, les dades que s'intercanviïn.

També es definiren quatre tipus de missatges principals. Primer, el "*session start data*", on les ulleres VR envien aquelles dades que siguin necessàries per a preparar una sessió, com ara la posició dels alumnes virtuals o les accions que es permetran fer des de la pàgina web. Segonament, el "*session ongoing data*", on les ulleres VR van enviant les dades de la sessió en temps real, com ara el moviment de les mans o l'estat dels alumnes virtuals. Tercerament, "*session request*", que serveix per a que les ulleres VR demanin al Servidor de Joc que se li assigni un numero de sessió lliure. Finalment, "*call event*", que com el seu nom indica serveix per a que el Servidor de Joc demani a les ulleres VR que es dugui a terme una acció determinada.

4.3.2 Connexió entre API Rest i WebSocket

En aquell moment teníem un procés oferint un servei de tipus API Rest per comunicar-se amb els clients web i, d'altra banda, un servei WebSocket per connectar el Servidor de Joc a les diferents ulleres VR per obtenir la informació. No obstant això, encara ens calia la part més important: la comunicació en temps real entre la simulació i el lloc web.

Aquesta comunicació és important a l'hora d'invocar esdeveniments des de la pàgina web del Mission Control i que arribin a la simulació en curs. Es tracta de que fent una crida a la Rest API, aquesta pugui demanar a una connexió de WebSocket del Servidor de Joc que enviï certa informació a una simulació.

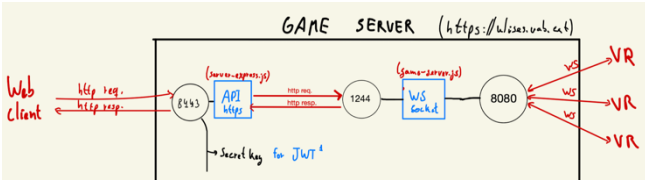


Figura 5. Funcionament intern del Servidor de Joc.

La solució més pràctica trobada, com es pot veure a la figura 5, és fer que el servei WebSocket disposi d'una sèrie d'*endpoints* de Rest API (al port 1244). Per tant, quan el servei de Rest API que ofereix *endpoints* al lloc web requereixi informació provinent d'una sessió en curs o necessiti enviar un esdeveniment, aquest farà una petició Rest al Servidor de Joc al client WebSocket.

Posem un cas pràctic: un usuari vol enviar un esdeveniment "A" a la simulació amb sessió 1234. Des de la web es fa clic al botó corresponent i envia una petició HTTP al servidor Rest demanant amb la ID de sessió, la ID de l'esdeveniment i la informació que calgui que s'executi un esdeveniment. A continuació, la Rest API rebrà la petició, revisarà les credencials de l'usuari i farà una petició Rest als endpoints que ofereix el procés de WebSocket que s'encarrega de les connexions amb les diferents ulleres VR. Finalment, el procés de WebSockets en rebre la petició i, si tot és correcte, cercarà el *socket* amb el número de sessió demanat i enviarà l'esdeveniment.

4.3.3 Gestió del *token* de sessió

Per a poder realitzar l'autenticació i gestió de credencials cal una eina que ens permeti crear i gestionar aquesta informació sensible de manera segura. JWT [18], conegut com a "JSON Web Tokens", és un estàndard que permet generar *tokens* que contenen l'algorisme de xifrat, un *payload* amb informació i una signatura d'autenticació.

Aquesta eina té com a objectiu facilitar al programador la gestió de credencials. En aquest cas, el *token* conté l'ID de l'usuari, el nom i el correu electrònic; i se li assigna una validesa de dues hores.

Aquest *token* es genera en iniciar sessió quan les credencials enviades són correctes, i s'envia amb totes les peticions que s'intercanviïn el servidor Rest i el client web mitjançant *cookies*. Aquesta *cookie* és *HTTP-Only* [19], és a dir, que el client no la podrà modificar ni utilitzar mitjançant JavaScript.

4.4 Quart *sprint*: programació i integració

A aquest quart *sprint* es va acabar de dissenyar el *backend* (tasca T7), i es va continuar programant tant el *frontend* com el *backend* (T6 i T9) alhora que es van realitzant tasques d'integració amb Didascalias (T10).

4.4.1 Últims retocs visuals

Atès que totes les finestres estaven dissenyades al prototip i de moment el *backend* ofereix *dummy responses*, quedava programar les últimes finestres. També, es volia que el lloc web es pogués fer servir des de dispositius com tauletes o telèfon; la qual cosa vol dir que calia assegurar-se de que sigui *responsive*.

Per fer-ho, la tasca de programació es redueix a ajustar i crear regles CSS per tal d'assegurar la capacitat d'adaptació del lloc web. Alhora, però, es van acabar de desenvolupar les finestres restants.

4.4.2 Integració i primeres proves

El correcte funcionament de les ulleres VR comunicant-se amb la estructura de servidors i el lloc web és fonamental i troncal. És per això que durant aquest sprint es va començar a integrar i provar el sistema amb els companys de Didascalias que disposen d'ulleres VR. Aquestes tasques es realitzaren en conjunt amb els programadors de Didascalias per testar les diferents funcionalitats, trobar errors i arreglar-los.

Fent aquestes proves es van trobar dos errors importants: un primer relacionat amb els esdeveniments que s'invocaven des de la web, on el primer arribava bé però a partir del segon el servidor es saturava; i un segon, relacionat amb la visualització del Mission Control quan s'introduïa una sessió inexistent o sense dades, que es quedava la pàgina web en blanc i saltaven errors.

El primer error fou causat per l'oblit de la instrucció de tancament "res.end();" a l'*endpoint* que s'encarrega dels esdeveniments a la Rest API, la qual cosa mai s'enviava resposta a la petició i es quedava el procés obert i bloquejat, consumint recursos. En afegir la instrucció oblidada, el problema va desaparèixer.

El segon error era més complex: no es contemplava la possibilitat de què la informació de la sessió fos "*null*", la qual cosa el component de la web es tancava o causava un error. S'ha solucionat tenint en compte que algunes variables poden ser *null* i, per tant, evitant errors.

4.5 Cinquè *sprint*: integració final i proves

A l'últim *sprint* del projecte, el cinquè, s'acabà de programar tot (tasques T6 i T9), es van fer les últimes tasques d'integració (T10), es van prendre les mesures de rendiment (T11), es creà la documentació (T12) i es va realitzar la una formació per passar el relleu als programadors del projecte Didascalias (T13).

4.5.1 Reconnexió de les ulleres VR

Un fet que es produïa de manera recurrent és que les ulleres VR es desconnectessin de la xarxa o perdessin la

connexió, fent que el *socket* es tanqués i la sessió es quedés oberta. En intentar connectar de nou, el Servidor de Joc retornava automàticament un nou número de sessió, entenent que tota connexió nova significava el començament d'una nova sessió.

Per arreglar-ho, es va plantejar que en fer la sol·licitud de sessió des de les ulleres VR en connectar-s'hi al Servidor de Joc, aquestes enviessin un identificador únic del dispositiu gràcies a la variable "*SystemInfo.deviceUniqueIdentifier*" [20]. Aquesta variable garanteix un *hash* únic respecte a tots els dispositius que existeixen.

Si guardem aquest identificador quan les ulleres VR demanen un número de sessió al Servidor de Joc, en desconnectar-se, podem fer que les ulleres VR demanin obrir una connexió facilitant de nou el seu identificador i la sessió que hi havia en curs. D'aquesta manera es pot garantir un sistema de reconnexió evitant problemes d'autenticació.

4.5.2 Documentació dins del codi

La documentació és essencial per entendre el sistema desenvolupat i el seu codi, permetent que altres programadors puguin prendre el relleu de les tasques de desenvolupament, reduint la quantitat de temps i esforç dedicat a la comprensió d'aquest.

En aquest cas, el funcionament del sistema a desenvolupat és fàcil d'entendre i amb la formació realitzada (tasca T12) és més que suficient. A partir d'aquí, s'ha considerat que és més important centrar els esforços en elaborar documentació del codi per permetre aquest relleu en cas de voler expandir el projecte.

Atès que el projecte està desenvolupat majoritàriament en JavaScript, la documentació creada s'anomena *JSDoc* [21]. Es una convenció de JavaScript i NodeJS que consisteix en comentar el codi per explicar i descriure funcions, mètodes, variables i altres elements destacats, de tal manera que es genera una documentació llegible i formatada.

4.5.3 Qualitat de software i proves

Per poder assegurar el bon funcionament del sistema es va decidir crear un *script* que imita el comportament d'unes ulleres VR. D'aquesta manera podem realitzar proves de caixa blanca per comprovar que les funcionalitats desenvolupades estan correctament programades. Amb aquest *script*, programat amb *JavaScript*, permet provar la connexió amb WebSocket cap al servidor, visualitzar l'intercanvi d'informació i testar la funcionalitat de reconnexió.

6 RESULTATS

Amb el sistema a desenvolupar gairebé acabat, es valorarà el resultat obtingut i es mostraran dades objectives que demostrin el bon funcionament. Es crea, per tant, un set de proves que permetrà valorar el rendiment dels diferents endpoints.

Es fa servir *autocannon* [22] per testar la Rest API: una eina de comparativa de rendiment basada en HTTPS que calcula la latències, connexions simultànies i diferents característiques, i amb una àmplia configuració de paràmetres.

Configurarem les proves de tal manera que es realitzin 50 i 100 connexions concurrents, amb 50 i 100 processos concurrents i amb una duració de la prova de 10 segons. D'aquesta manera, cada procés tindrà una connexió concurrent. Les 50 connexions representarien un ús intensiu del sistema perquè la quantitat d'alumnes a una aula d'un màster rarament arriba a aquesta quantitat. D'altra banda, les 100 connexions representen un ús extrem rarament assolible.

Es provaran els següents endpoints: el que ofereix les dades d'una sessió en temps real, perquè és l'*endpoint* que més es farà servir; el que tracta l'inici de sessió amb contingut d'usuari i contrasenya aleatoris, per provar situacions de possibles atacs; i el que ofereix les dades bàsiques d'un usuari, que es crida quan s'entra al portal principal d'un usuari que ja ha iniciat sessió.

1. Dades de sessió en marxa. Es crea una simulació amb un ID que va generant informació i enviant-la per WebSocket al Servidor de Joc en temps real. Es realitzen les proves a l'*endpoint* especificant la ID de la sessió creada.

```
Running 10s test @ https://cyclops.uab.cat/api/session/385/ongoingWS
50 connections
50 workers

\
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	60 ms	124 ms	268 ms	1261 ms	152.14 ms	158.2 ms	1925 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	12	12	356	458	326,2	123,07	12
Bytes/Sec	7.46 kB	7.46 kB	221 kB	285 kB	203 kB	76.4 kB	7.46 kB

```
Req/Bytes counts sampled once per second.
# of samples: 500

3k requests in 10.21s, 2.03 MB read
```

Figura 6. Prova de sessió en marxa amb 50 clients.

```
Running 10s test @ https://cyclops.uab.cat/api/session/385/ongoingWS
100 connections
100 workers

\
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	125 ms	507 ms	4770 ms	6663 ms	745.88 ms	1050.75 ms	8082 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	74	74	247	368	252,3	82,93	74
Bytes/Sec	46 kB	46 kB	153 kB	228 kB	157 kB	51.4 kB	46 kB

```
Req/Bytes counts sampled once per second.
# of samples: 1000

3k requests in 10.81s, 1.57 MB read
```

Figura 7. Prova de sessió en marxa amb 100 clients.

Com es pot apreciar a les figures 6 i 7, els resultats són bastant positius tenint en compte la quantitat de peticions

[1]*Per saber més detalls sobre què és es pot consultar el següent web: https://www.microsoft.com/es-es/microsoft-365/microsoft-whiteboard/*

que s'estan realitzant. Amb la càrrega de 50 clients, se suporten una mitja de 320 peticions per segons; i amb una càrrega de 100 clients, se suporten una mitja de 250 peticions. Aquestes quantitats són molt bones, doncs mostren una alta capacitat. A més, les latències amb 50 clients es mantenen molt baixes (<300ms pel 97.5% dels casos); no obstant amb 100 clients aquestes augmenten dràsticament fins a extrems inacceptables (8 segons) en algunes peticions molt puntuals, però la mitja ens indica que ens movem entre valors acceptables.

2. Inici de sessió. En aquest cas la simulació serà una petició amb mètode *POST* [23] afegint l'adreça electrònica i la contrasenya d'un usuari vàlid. D'aquesta manera avaluarem l'inici de sessió quan s'introdueixin credencials vàlides i, per tant, es realitzen consultes SQL i es genera el *token* de sessió.

```
Starting...
Running 10s test @ https://cyclops.uab.cat/api/login
50 connections
50 workers

/
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	918 ms	2754 ms	6720 ms	7020 ms	2803.64 ms	1475.82 ms	7020 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	6	9	5,8	2,45	3
Bytes/Sec	0 B	0 B	7.08 kB	10.6 kB	6.84 kB	2.88 kB	3.54 kB

```
Req/Bytes counts sampled once per second.
# of samples: 500

138 requests in 10.04s, 68.4 kB read
30 errors (30 timeouts)
```

Figura 8. Prova de *login* amb 50 clients.

```
Starting...
Running 10s test @ https://cyclops.uab.cat/api/login
100 connections
100 workers

/
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	664 ms	2692 ms	6883 ms	7328 ms	2923.21 ms	1446.67 ms	7328 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	2	2	7	8	6,2	1,99	2
Bytes/Sec	2.36 kB	2.36 kB	8.26 kB	9.44 kB	7.31 kB	2.35 kB	2.36 kB

```
Req/Bytes counts sampled once per second.
# of samples: 1000

241 requests in 10.39s, 73.1 kB read
78 errors (78 timeouts)
```

Figura 9. Prova de *login* amb 100 clients.

A les figures 8 i 9 podem observar que la quantitat de peticions per segon es redueix extremadament fins a una mitja de 6. A més, la latència creix de manera clara i contundent.

Aquests resultats son esperables, doncs estem realitzant consultes SQL i generant *tokens*, i és lògic que aquest *endpoint* no suporti bé grans càrregues puix que l'inici de sessió és una funcionalitat que no s'executarà molt seguidament. No obstant això, un atacant podria aprofitar aquesta debilitat per atacar el web.

Les latències de mitja són bastant elevades, però són perfectament acceptables, doncs com hem dit és un *endpoints* que no s'executarà repetidament i que realitza una operació complexa que no requereix d'immediatesa.

El que sí que pot suposar un problema és la quantitat d'errors i *timeouts* que es produeixen quan la càrrega és elevada. Un 21% (30 errors/138 req.) quan fem 50 connexions i un 32% (78 errors / 241 req.) quan en fem 100.

3. Dades de la *dashboard*. Aquesta simulació, simplement, es limita a fer una petició HTTP amb un *token* vàlid i rebre resposta. Recordar, a més, que la resposta rebuda és una *dummy response*, per la qual cosa no s'inclouen consultes SQL dins d'aquest anàlisi de rendiment. El que sí que es requereix a aquest *endpoint* és un *token* vàlid, per tant sí que entra dins d'aquest anàlisi l'execució de les funcions de comprovació de *tokens*.

```
Running 10s test @ https://cyclops.uab.cat/api/getDashboard
50 connections
50 workers

|
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	64 ms	194 ms	804 ms	3153 ms	260.34 ms	502.05 ms	5886 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	44	44	199	238	189,8	54,05	44
Bytes/Sec	24.8 kB	24.8 kB	112 kB	134 kB	107 kB	30.4 kB	24.8 kB

```
Req/Bytes counts sampled once per second.
# of samples: 500

2k requests in 10.5s, 1.07 MB read
```

Figura 10. Prova de *dashboard* amb 50 clients.

```
Running 10s test @ https://cyclops.uab.cat/api/getDashboard
100 connections
100 workers

\
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	111 ms	339 ms	1420 ms	5679 ms	485.44 ms	772.56 ms	8093 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	71	71	231	256	214,6	51,27	71
Bytes/Sec	40 kB	40 kB	130 kB	144 kB	121 kB	28.9 kB	40 kB

```
Req/Bytes counts sampled once per second.
# of samples: 1000

2k requests in 10.39s, 1.21 MB read
29 errors (26 timeouts)
```

Figura 11. Prova de *dashboard* amb 50 clients.

Com es pot apreciar a les figures 10 i 11, la quantitat de peticions per segon és un terç més baixa que a les proves de sessió. Els temps de resposta mostren que quan hi ha més peticions, respon millor que les proves de sessió; però quan hi ha menys aquest temps es inferior i per tant millor a les proves de sessió. Podem concloure que suporta bé davant gran quantitat de tràfic, però hi poden haver situacions de *timeout* [24] que al test de sessió no s'han donat.

En resum, les proves indiquen que la Rest API pot suportar grans càrregues de peticions simultànies amb alguns *timeouts* en certs casos. Els *endpoints* que no es preveuen que pateixin un ús repetit mostren pics inacceptables quan es realitza un ús intensiu, mentre que aquells que sí que s'utilitzen seguidament es mantenen en bones condicions sota aquestes condicions. D'altra banda, queda constància que els endpoints més lents podrien ser aprofitats per atacs de denegació de servei [25].

7 Conclusions

A l'inici es va proposar crear una extensió del projecte Didascalias basada en tecnologia web que permetés una millor i fàcil interacció i monitorització de les simulacions. Era una idea abstracta que havia de prendre forma. Es van estudiar diferents casos d'ús, es van proposar uns requisits base i es van explicar les possibilitats que es podrien obrir. Seguidament, es va acordar un sistema amb una interfície amb els desenvolupadors de Didascalias, representada mitjançant un prototip digital.

El *frontend* s'ha desenvolupat correctament segons el prototip mencionat, encara que algunes funcionalitats han quedat pendents d'acabar: el canvi d'idioma, la funció d'un super-usuari que pugui gestionar diferents universitats o instàncies, o una interfície que permeti als administradors crear usuaris sense necessitat de tocar directament scripts o bases de dades.

Pel que fa al *backend*, el seu disseny ha anat evolucionant davant diferents casuístiques o complicacions trobades. La escalabilitat del codi ha estat un factor important a tenir en compte. Es pot afirmar que totes les funcionalitats bàsiques i el funcionament del Mission Control han estat desenvolupades correctament. Queden pendents, però, la creació de taules de base de dades del MySQL per guardar assignacions de sessions i informació variada, així com canviar algunes *dummy responses* per aquestes dades reals.

De cara al futur, s'obren moltes funcionalitats a partir de l'arquitectura creada, i que pot anar escalant segons les necessitats de l'aplicació. Es poden posar sobre la taula altres projectes com ara: millorar el servidor *backend* de tal manera que s'emmagatzemin dades de sessions i comportament de l'usuari i poder fer mineria de dades per trobar patrons; afegir eines de gestió d'usuaris i administrador dins del web; comprovar la seguretat de l'aplicació, cercant vulnerabilitats i millorant la integritat i confidencialitat de les comunicacions; estudiar l'UX i l'UI per tal de millorar la usabilitat; reduir les latències i atacs de denegació de servei; etcètera.

8 Bibliografia

[1] Espanya, Ministeri d'Educació, "Reial decret 1834/2008, de 28 de novembre", *Boletín Oficial del Estado*, novembre 2008. https://www.boe.es/boe_catalan/dias/2008/11/28/pdfs/BOE-A-2008-19174-C.pdf (Accés: 14 oct. 2023).

[2] Álvarez, I.M., Manero, B., Morodo, A., Suñé-Soler, N., Henao, C., et al. Realidad Virtual Inmersiva para mejorar la competència de gestión del clima del aula en secundaria [Inmersive Virtual Reality to improve competence to manage classroom climate in secondary schools]. *Educación XX1*, 26(1), 249-272, gener 2023. https://doi.org/10.5944/educxx1.33418

[3] B. W. Boehm, "A spiral model of software development and enhancement", *Computer*, vol. 21, n. 5, p. 61-72, maig 1988, doi: 10.1109/2.59.

[4] M. Johansen, M. Pichlmair and S. Risi, "Video Game Description Language Environment for Unity Machine Learning Agents", *2019 IEEE Conference on Games (CoG)*, London, UK, pp. 1-8, agost 2019, doi: 10.1109/CIG.2019.8848072.

[5] Pop, Paul. "Comparing web applications with desktop applications: An empirical study". *Linköping University*, Linköping, Sweden. 2002. http://www.imm.dtu.dk/~paupo/publications/Pop2002aa-Comparing%20Web%20Applications%20wit-.pdf (Accés: 11 nov. 2023).

[6] M. Callum, *VueJS: Up and Running: Building Acessible and Performant Web Apps*, Edición 1. Sebastopol: O'Reilly Books, 2018.

[7] Arriba Garcia, R. "Estudio de la popularidad del framework VueJS", *Radega.com*, maig 2019. https://www.radega.com/wp-content/uploads/2019/06/Estudio_de_la_popularidad_framework_VueJS.pdf (Accés: 15 oct. 2023).

[8] E. Saks, "JavaScript Frameworks: Angular vs React vs Vue". *Haaga-Helia University of Applied Sciences*. Helsinki, Finland, juliol 2019, https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf (Accés: 12 nov. 2023).

[9] E. Haro et al, "Desarrollo backend para aplicacions web, Servicios Web Restful: Node.js vs Spring Boot", *Revista Ibérica De Sistemas e Tecnologías De Informaçao*, pp. 309-321, 2019. https://www.proquest.com/scholarly-journals/desarrollo-backend-para-aplicaciones-web/docview/2195127847/se-2?accountid=15292 (Accés: 15 oct. 2023).

[10] J. Liberty, D. Hurwitz, "Programmiong ASP.NET 2.0", Edición 3. O'Reilly Books, october 2005.

[11] Lavarel, "Lavarel GitHub Repository", *Lavarel*. https://github.com/laravel/laravel. (Accés 11 nov. 2023).

[12] López Herrera, Patricia. "Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL", *Universidad Autónoma del Estado de México*. març 2016. https://core.ac.uk/download/pdf/80528621.pdf (Accés: 12 nov. 2023).

[13] V. Pimentel and B. G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket", *IEEE Internet Computing*, vol. 16, no. 4, pp. 45-53, agost 2021, doi: 10.1109/MIC.2012.64.

[14] Suzanti I. O., Fitriani N., Jauhari A., Khozaimi A., "REST API Implementation on Android Based Monitoring Application", *Internationa Conference on Science and Technology 2019 Indonesia*, vol 1569, octubre 2019, doi: 10.1088/1742-6596/1569/2/022088

[15] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format", *Internet Engineering Task Force*, RFC 8259, desembre 2017, https://datatracker.ietf.org/doc/html/rfc8259 (Accés: 15 oct. 2023).

[16] Johnston P., "Authentication and session management on the web", *Retrieved December*. 2004, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a39d484460d2adc3fe6bbd3a42fc916fbe125b80. (Accés: 23 des. 2023)

[17] Sriramya P., and R. A. Karthika. "Providing password security by salted password hashing using bcrypt algorithm." *ARPN journal of engineering and applied sciences*, 2015, 10.13 5551-5556. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9d82620f0866ec773b12c30f90d70b74661f972f. (Accés 23 des. 2023).

[18] M. Jones, Microsoft, J. Bradley, et al., "JSON Web Token (JWT)", *Internet Engineering Task Force*, RFC 7519, maig 2015, https://datatracker.ietf.org/doc/html/rfc7519. (Accés 28 des. 2023).

[19] Zhou Y. And Evans D., "Why aren't HTTP-Only cookies more widely deployed", *Proceedings of 4th Web*, 2010, https://yuchen-zhou.com/docs/http-only.pdf. (Accés 28 des. 2023).

[20] Unity, "Unity Documentation – SystemInfo.deviceUniqueIdentifier", *Unity version 2022.3*, https://docs.unity3d.com/ScriptReference/SystemInfo-deviceUniqueIdentifier.html (Accés 21 gen. 2024).

[21] Zakas Nicholas C., "Maintainable JavaScript", Taiwan: *O'Reilly Media*, 2012.

[22] Hunter I. T., Thomas, "Distributed Systems with Node.js", Taiwan: *O'Reilly Media*, 2020.

[23] MDN Web Docs, "POST Method", *MDN Web Docs*, https://developer.mozilla.org/es/docs/Web/HTTP/Methods/POST (Accés 1 feb. 2024).

[24] Wikipedia contributors, "Timeout (computing)", *Wikipedia The Free Encyclopedia*, gener 2023, https://en.wikipedia.org/wiki/Timeout_(computing) (Accés 2 feb. 2024).

[25] Jelena Mirkovic and Peter Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms", *SIGCOMM Comput*, Commun. Rev. 34, no. 2, pp. 39-53, abril 2024, doi: 10.1145/997150.997156.

APÈNDIX

A. DIAGRAMA DE GANTT

