Achievers Go (A-GO): Plataforma Web per gestionar l'avenç dels alumnes en una assignatura utilitzant tècniques de gamificació

Aran Franco Carreras

9 de febrer de 2026

Resum– L'objectiu del projecte és desenvolupar una plataforma web adreçada tant als docents com als estudiants per facilitar la gestió i seguiment de l'activitat acadèmica a l'aula. Integrant tècniques de gamificació, es permeten crear assignatures amb activitats i tasques associades, assignar-hi insígnies i punts, importar estudiants mitjançant fitxers CSV, organitzar-los en grups i subgrups de treball i fer-ne l'avaluació. Aquesta solució permet un seguiment continu del progrés acadèmic i l'augment de la motivació i participació de l'alumnat. Per a emmagatzemar les dades, s'ha creat una base de dades basada en un model relacional utilitzant *MariaDB*. El *backend* s'ha implementat amb *Node.js* i *Express.js*, mitjançant una API REST que gestiona les peticions del sistema. El *frontend* s'ha desenvolupat amb *Vue.js* seguint una arquitectura SPA, complementada amb l'ús de *Vue Router*. Finalment, el sistema s'ha encapsulat en un entorn dockeritzat per a un futur desplegament.

Paraules clau– Gamificació, seguiment acadèmic, SPA, Vue.js, Node.js, Express.js, MariaDB, API REST, Docker, CSV, UCD, Figma, CSS, HTML, JavaScript, Scrum, Frontend, Backend.

Abstract– This project aims to develop a web platform designed for both teachers and students to allow the management and monitoring of academic activities in the classroom. By integrating gamification techniques, the platform enables the creation of courses with associated activities and tasks, the assignment of badges and points, the import of students via CSV files, the organization of students into work groups and subgroups, and performance evaluation. This solution allows continuous tracking of academic progress while enhancing student motivation and engagement. Data is stored in a relational database implemented with *MariaDB*. The *backend* is developed using *Node.js* and *Express.js* through a REST API that handles system requests. The *frontend* is built with *Vue.js* following a SPA (Single Page Application) architecture, complemented by *Vue Router* for navigation. Finally, the system is encapsulated in a dockerized environment for a future deployment.

Keywords– Gamification, Academic tracking, SPA, Vue.js, Node.js, Express.js, MariaDB, REST API, Docker, CSV, UCD, Figma, CSS, HTML, JavaScript, Scrum, Frontend, Backend.

1 Introducció - Context de treball

En un món on l'estímul constant és un imperatiu en el nostre dia a dia, l'àmbit de l'educació no se n'ha quedat exempt. Sovint l'alumnat necessita rebre resultats immediats per a sentir-se motivat. En aquest context, neix A-GO, una plataforma web que integra mecanismes de gamificació amb l'objectiu de fomentar un aprenentatge més actiu i incrementar la participació i implicació de l'alumnat a l'aula.

El projecte A-GO pretén potenciar la motivació de l'alumnat incorporant dinàmiques pròpies dels jocs com, ara bé, insígnies i recompenses, creant així un entorn en què l'estudiant esdevingui el protagonista del seu propi aprenentatge. A més, la plataforma busca proporcionar al professorat eines efectives per a l'avaluació dels estudiants.

Per tal d'entendre el funcionament de la plataforma, aquest document s'endinsa en l'anàlisi de plataformes de gamificació ja existents en l'àmbit educatiu, com a part de l'estat de l'art. A continuació, es descriuen els objectius

• E-mail de contacte: 1607152@uab.cat
• Menció: Tecnologies de la Informació
• Treball tutoritzat per: Maria Carmen de Toro (Departament d'Enginyeria de la Informació i de les Comunicacions)
• Curs 2025/26

principals del projecte, derivats de la seva motivació inicial, així com la metodologia que es seguirà per assolir-los i la planificació prevista. També s'explica el procés de desenvolupament d'A-GO, que inclou el disseny de la plataforma i l'arquitectura. Finalment, el document es completa amb la presentació dels resultats obtinguts i les conclusions finals.

2 ESTAT DE L'ART

A-GO no presenta una experiència completament innovadora, plataformes com, ara bé, Kahoot [3] o Socrative [2] ja han demostrat poder dinamitzar l'aula i estimular l'aprenentatge de l'alumnat.

Kahoot per exemple, aposta per una mecànica de joc basada en qüestionaris. Aquests es caracteritzen per preguntes de tipus vertader/fals o de resposta múltiple, dissenyades per fomentar respostes ràpides. Un cop creats i realitzats els qüestionaris, la plataforma mostra les puntuacions i els *leaderboards* en temps real. Així doncs, es promou una dinàmica competitiva que motiva a l'alumnat.

Socrative, tot i que també pot incorporar elements de competició mitjançant funcionalitats com Space Race, prioritza una avaluació de caràcter més formatiu. La plataforma permet a l'estudiant respondre les preguntes desenvolupant i argumentant les seves respostes, de manera que el professorat pot proporcionar *feedback* i realitzar una avaluació més profunda. A més, ofereix la possibilitat de generar i descarregar informes detallats dels resultats, que permet al docent fer un seguiment del rendiment de l'alumnat.

Tanmateix, tal com s'ha observat, la majoria d'aquestes eines es limiten a l'avaluació immediata sense oferir una visió global del desenvolupament acadèmic de l'estudiant. Tot i que Socrative permet obtenir informes de resultats, la plataforma no està dissenyada per realitzar un seguiment continu de tota una assignatura. Per aquest motiu, A-GO pretén esdevenir una plataforma web que integri mecàniques de seguiment, que permeten al professorat obtenir informació detallada sobre el progrés global de cada estudiant. No només amb finalitats avaluatives, sinó també per adaptar l'ensenyament a les necessitats individuals. A més, proporciona a l'alumnat una visió del seu propi progrés dins de l'assignatura, complementada amb elements de gamificació que incrementen la motivació i el compromís amb l'aprenentatge.

3 OBJECTIUS

A continuació es descriuen els objectius del projecte, diferenciant entre objectius generals, que defineixen la finalitat principal del sistema, i objectius específics, que concreten les funcionalitats necessàries per assolir-la.

3.1 Objectius Generals:

Crear una plataforma que permeti als usuaris donar-se d'alta i accedir-hi mitjançant l'autenticació OAuth de la Universitat Autònoma, garantint la distinció de rols entre estudiants i docents. Els docents disposaran d'eines per gestionar assignatures, assignar-hi grups i subgrups de treball i crear activitats amb tasques associades, per fer un seguiment complet del progrés de l'alumnat. Els estudiants podran seguir el seu rendiment, veure les tasques pendents i rebre *feedback* del docent. A més, la incorporació de mecàniques de gamificació, com insígnies i punts, farà que la plataforma es diferenciï d'altres entorns educatius, com Moodle, fomentant la participació i la motivació dels alumnes.

3.2 Objectius Específics:

- Desenvolupar una plataforma amb dos rols d'usuaris: estudiants i professors, cadascun amb el seu *backoffice* i *frontoffice* adequats.
- Permetre als docents crear assignatures, associar-hi estudiants dividits per grups a partir d'un fitxer CSV i assignar-los posteriorment a subgrups de treball.
- Crear activitats amb una data de venciment, i tasques associades que també tinguin dates de venciment individuals, sempre anteriors a la data final de l'activitat.
- Establir un sistema d'insígnies de diferents nivells segons el nombre de tasques completades dins del període establert de l'activitat (buida, bronze, platejada i daurada), de manera que fomenti el treball constant.
- Assignar punts a cada tasca segons si s'ha completat dins del període establert de cada tasca, de manera que els punts premiïn la participació en l'aula.
- Proporcionar al professorat tres vistes dins de cada assignatura: visualització de les activitats amb les seves tasques, assignació dels alumnes als subgrups de treball i una vista per a l'avaluació acompanyada d'un *overview* del progrés dels estudiants.
- Proporcionar als estudiants:
  - Un *backoffice* per a l'edició del seu avatar,
  - Una vista per seguir el seu progrés, l'estat de les tasques, les insígnies, els punts obtinguts i el *feedback* rebut.
  - Una versió anonimitzada del progrés dels altres subgrups en format de rànquing setmanal.

4 METODOLOGIA

Per al desenvolupament del projecte, s'ha emprat una metodologia àgil basada en Scrum [1], un marc de treball que permet organitzar el projecte en fases curtes de desenvolupament anomenades sprints. Aquests permeten una gestió eficient del treball i contribueixen a aconseguir els resultats proposats.

Cada sprint, amb una durada aproximada de dues setmanes, es divideix en objectius representats en tasques concretes. Aquestes tasques s'han gestionat mitjançant la plataforma Jira [4], que permet visualitzar-ne l'estat, establir prioritats segons les necessitats del projecte i generar gràfics i estadístiques de seguiment del progrés. A més, cada sprint ha anat acompanyat de reunions setmanals de tutorització per revisar l'avanç i resoldre els dubtes que han sorgit.

Addicionalment, la planificació temporal del projecte s'ha complementat amb un diagrama de Gantt creat amb ProjectLibre [5], que permet representar les dependències entre tasques, els terminis establerts i els recursos emprats.

Aquesta metodologia ha facilitat el seguiment del progrés, assegurant la documentació de totes les decisions preses durant l'execució del projecte, i simultàniament, ha permès detectar possibles desviacions temporals amb més agilitat.

Cal destacar l'ús de la intel·ligència artificial generativa en aquest projecte, que s'ha utilitzat per a millorar la redacció dels textos i per al disseny d'avatars personalitzats dels usuaris, però en cap cas com a font d'informació.

Finalment, el codi del projecte està disponible al repositori GitHub[1].

5 PLANIFICACIÓ

El diagrama de Gantt de la Figura 1 mostra la planificació del projecte. Aquesta es divideix en les següents fases:

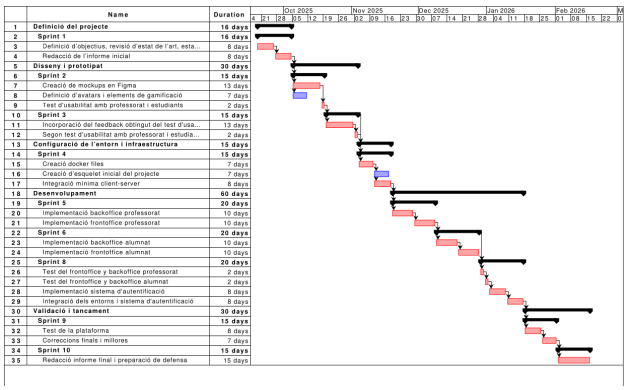
- En primer lloc, la fase de definició del projecte es realitza en un únic sprint amb una durada de dues setmanes. Aquesta etapa inclou la definició dels objectius, la revisió de l'estat de l'art, la selecció de la metodologia i de les eines, així com la redacció de l'informe inicial.
- La segona fase correspon al disseny i prototipat, composta per dos sprints que s'estenen al llarg de quatre setmanes. Durant aquesta etapa es desenvolupen els *mockups* amb Figma, es defineixen els avatars i els elements de gamificació, i es duen a terme proves amb professorat i alumnat per validar les primeres idees abans d'iniciar la implementació.
- A continuació, en la fase de configuració de l'entorn, amb una durada de dues setmanes, es creen els Dockerfiles, l'esquelet inicial del projecte i una integració mínima client-servidor, que servirà de base per al desenvolupament posterior.
- La quarta fase correspon al desenvolupament i és la més extensa del projecte, amb una durada total de vuit setmanes repartides en tres sprints. Durant aquesta fase s'implementa el *backoffice* i el *frontoffice* per al professorat i l'alumnat, així com el sistema d'autenticació i la integració entre els diferents entorns.
- Finalment, la fase de validació i tancament es divideix en dos sprints finals. Aquesta fase inclou proves de la plataforma, resolució d'errors i millores finals, així com la redacció de l'informe final i la preparació de la defensa del projecte.

5.1 Canvis respecte a la proposta inicial de planificació

Per causes alienes al projecte, la planificació va patir un lleu retard: l'endarreriment en el desenvolupament del *backoffice* i *frontoffice* del professorat durant l'Sprint 5.

Aquest va provocar que l'inici de l'Sprint 6 es desplacés al 22 de gener, és a dir, quatre setmanes més tard del previst. Malgrat això, la implementació del *frontend* i *backend* de l'alumnat es va aconseguir completar en una setmana, avançant el pronòstic inicialment establert.

[1] https://github.com/nara3205/Achievers-Go



	Name	Duration
1	Definició del projecte	16 days
2	Sprint 1	16 days
3	Definició d'objectius, revisió d'estat de l'art, esta...	8 days
4	Redacció de l'informe inicial	8 days
5	Disseny i prototipat	30 days
6	Sprint 2	15 days
7	Creació de mockups en Figma	13 days
8	Definició d'avatars i elements de gamificació	7 days
9	Test d'usabilitat amb professorat i estudiants	2 days
10	Sprint 3	15 days
11	Incorporació del feedback obtingut del test d'usa...	13 days
12	Segon test d'usabilitat amb professorat i estudia...	2 days
13	Configuració de l'entorn i infraestructura	15 days
14	Sprint 4	15 days
15	Creació docker files	7 days
16	Creació d'esquelet inicial del projecte	7 days
17	Integració mínima client-server	8 days
18	Desenvolupament	60 days
19	Sprint 5	20 days
20	Implementació backoffice professorat	10 days
21	Implementació frontoffice professorat	10 days
22	Sprint 6	20 days
23	Implementació backoffice alumnat	10 days
24	Implementació frontoffice alumnat	10 days
25	Sprint 8	20 days
26	Test del frontoffice y backoffice professorat	2 days
27	Test del frontoffice y backoffice alumnat	2 days
28	Implementació sistema d'autenticació	8 days
29	Integració dels entorns i sistema d'autenticació	8 days
30	Validació i tancament	30 days
31	Sprint 9	15 days
32	Test de la plataforma	8 days
33	Correccions finals i millores	7 days
34	Sprint 10	15 days
35	Redacció informe final i preparació de defensa	15 days

Fig. 1: Diagrama de Gantt del projecte.

A conseqüència del retard, es va reduir la realització de proves a la plataforma més enllà de les previstes a les sessions setmanals de tutoria. També es va veure afectada la implementació del sistema d'autenticació amb la Universitat Autònoma, la qual no es va poder dur a terme.

Altrament, els Sprints 8 i 9 es van agrupar amb l'Sprint 10. Així doncs, aquesta última fase es va centrar a dur a terme millores generals de la plataforma i es va dedicar temps a la redacció final de l'informe.

6 DISSENY DE LA PLATAFORMA

Amb l'objectiu d'establir uns fonaments sòlids per al desenvolupament de l'aplicació final, s'ha elaborat un prototip mitjançant Figma [6], una plataforma de disseny d'interfícies que permet simular la interacció entre pantalles i definir els fluxos principals de navegació.

L'elaboració del prototip s'ha basat en els principis del UCD (*User-Centered Design*), Figura 2, que situa les necessitats dels usuaris al centre del procés de disseny. Aquest model promou la participació activa dels usuaris, l'avaluació contínua de l'ús de la plataforma, la incorporació de millores i la iteració constant. Per tant, l'ús d'aquests principis de disseny, ha permès definir la plataforma a partir de les necessitats reals dels futurs usuaris: docents i estudiants.

Consegüentment, el procés de disseny no s'ha plantejat com una fase estàtica, sinó com un cicle iteratiu continu. Per validar el prototip, s'han dut a terme dos tests d'usabilitat amb usuaris del perfil docent i un amb un usuari estudiant. Durant aquestes sessions de test, s'han assumit simultàniament els rols de facilitador i d'escriba per presentar la plataforma a l'usuari: com a facilitador, s'ha guiat la interacció amb la plataforma; com a escriba, s'han registrat de manera objectiva les accions, reaccions i comentaris dels participants.

El *feedback* recollit dels usuaris s'ha analitzat posteriorment i s'ha incorporat en noves iteracions del prototip, ajustant el disseny a les necessitats identificades. Paral·lelament, s'han dut a terme reunions setmanals de tutoria per a revisar el disseny i proposar millores.

D'aquesta manera, s'ha aconseguit crear un primer prototip funcional de la plataforma A-GO (vegeu Apèndix 2, A.2), que ha servit com a base per al posterior desenvolupament de la plataforma.

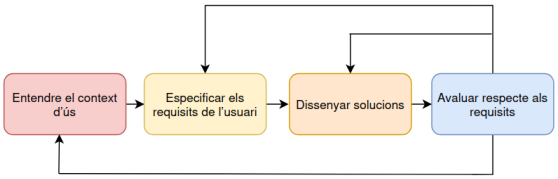


Fig. 2: Principis del disseny centrat en l'usuari.

7 ARQUITECTURA DE LA PLATAFORMA

Per tal de presentar l'arquitectura del sistema, que segueix l'esquema representat a la Figura 3, s'ha decidit dividir aquesta secció en dues parts principals: Disseny de la Base de Dades i Solució Tecnològica.

La primera part descriu l'estructura relacional de la base de dades i les seves taules principals, mentre que la segona exposa les decisions tecnològiques emprades per al desenvolupament del *backend*, del *frontend* i dels entorns dockeritzats.

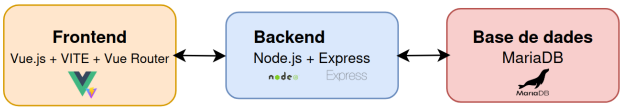


Fig. 3: Esquema de l'arquitectura de la plataforma.

7.1 Disseny de la Base de Dades:

S'ha dissenyat una base de dades basada en un model relacional. Aquesta elecció vol donar resposta a la naturalesa altament interconnectada de les dades que gestiona el sistema. Així doncs, aquest model permet la integritat de les dades actuals així com una estructura fàcilment escalable per a futures implementacions.

El disseny s'ha dut a terme mitjançant ERD Editor [7], una eina que ha permès definir l'esquema conceptual i les relacions entre entitats de manera visual. Posteriorment, el model s'ha exportat en un fitxer *.init*, utilitzat per a la inicialització amb el sistema gestor de bases de dades *MariaDB* [13].

La base de dades, presentada a la Figura 4, està formada per les següents taules:

- **Users**: conté la informació dels usuaris, tant d'alumnes com de professors, diferenciats pel camp *tipus*.
- **Groups** i **Subgroups**: permeten organitzar els estudiants en grups i subgrups de treball dins d'una assignatura.
- **Subject**: representa les assignatures, cadascuna associada a un docent responsable.
- **Assignment**: defineix les activitats principals d'una assignatura. Cada activitat tindrà vinculada una insígnia.
- **Task**: recull les tasques específiques dins de cada **Assignment**, amb la seva data de venciment i puntuació associada.
- **Subgroups_task**: registra les entregues de tasques dels subgrups i si s'han lliurat dins del termini establert.
- **Student_group**: permet establir la relació entre **Groups** i **Users** de tipus estudiant, indicant quins estudiants formen part de cada grup. Aquesta taula és necessària perquè la importació dels estudiants es farà a partir d'un fitxer CSV, el qual només contindrà els alumnes i els grups als quals corresponen. L'assignació de subgrups es farà posteriorment pel docent.
- **Subgroups_student**: estableix la relació entre **Subgroups** i **Users** del tipus estudiant, indicant quins estudiants formen part de cada subgrup de treball.

Aquesta arquitectura facilita la traçabilitat completa de la informació, des de la creació d'una assignatura, l'assignació d'estudiants a aquesta, fins a l'entrega de cada tasca per un subgrup. Es proporciona al docent funcionalitats d'avaluació, gamificació i seguiment del progrés dels alumnes, tal com s'ha establert en els objectius.

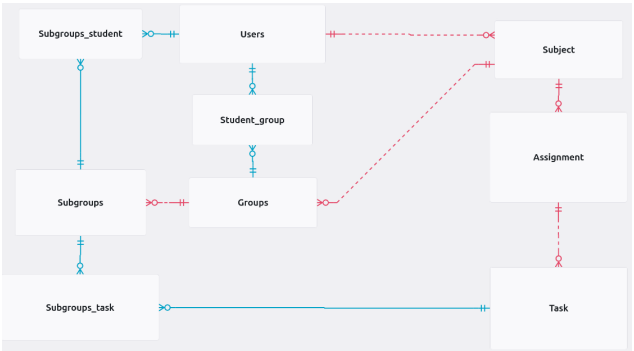


Fig. 4: Esquema relacional de la base de dades de la plataforma.

7.2 Solució tecnològica:

Aquesta secció es divideix en diverses subseccions amb l'objectiu de descriure les diferents eines emprades en el desenvolupament del projecte.

7.2.1 Entorn Dockeritzat:

Tal com mostra la Figura 5, s'han creat fitxers Docker específics que permeten definir l'entorn d'execució de la plataforma.

Aquesta arquitectura es basa en l'ús de dos components principals de Docker:

- **Dockerfiles**: s'han definit dos fitxers Docker separats, un per al *frontend* i un altre per al *backend*, amb l'objectiu de crear entorns d'execució independents per a cada servei. Aquests fitxers automatitzen la instal·lació de les dependències amb *npm* (gestor de paquets de NodeJS), configuren el directori de treball corresponent i exposen els ports necessaris per la comunicació entre serveis.

- **Docker Compose**: actua com a gestor de l'orquestració dels tres serveis (*frontend*, *backend* i base de dades *MariaDB*). Aquest fitxer coordina la comunicació interna entre contenidors a través d'una xarxa virtual i gestiona la persistència de dades mitjançant volums, assegurant que la informació de la base de dades no es perdi en reiniciar els contenidors.

L'ús d'aquest entorn garanteix que qualsevol canvi en el codi font es reflecteixi immediatament en el contenidor corresponent. Addicionalment, el desplegament del projecte en producció es farà en un entorn completament dockertitzat. Per aquest motiu, s'ha optat per iniciar el desenvolupament directament en aquests entorns, evitant incompatibilitats futures i facilitant el desplegament del sistema.

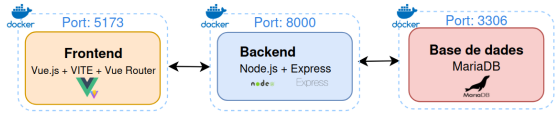


Fig. 5: Esquema de l'entorn dockertizat.

7.2.2 Backend:

La part del servidor s'ha implementat utilitzant *Node.js* [8] que ha estat escollit principalment per la seva arquitectura basada en esdeveniments i el seu model d'entrada/sortida no bloquejant, que permet gestionar de manera eficient un gran nombre de peticions simultànies. És per aquest motiu que el seu ús és especialment rellevant en aplicacions web amb múltiples usuaris connectats alhora, com és el cas d'aquest projecte.

El servidor s'ha implementat seguint una arquitectura REST, on les funcionalitats del sistema s'ofereixen mitjançant serveis web accessibles a través de peticions HTTP. Aquesta arquitectura facilita la separació entre client i servidor i contribueix a fer el sistema més escalable i fàcil de mantenir.

A partir d'aquesta arquitectura, s'ha creat una API REST utilitzant el framework *Express* [9], que permet definir els diferents *endpoints* de manera senzilla.

Paral·lelament, s'ha incorporat el paquet *CORS* [14] per controlar la comunicació entre el *frontend* i el *backend*. En aplicacions web és habitual que aquests s'executin en ports diferents. Així doncs, la política CORS (*Cross-Origin Resource Sharing*) permet gestionar les peticions HTTP entre diferents dominis o orígens.

Durant el desenvolupament, es va produir el següent error en intentar comunicar el *frontend* amb l'API:

```
Access to fetch at 'http://localhost:8000/api/login'
from origin 'http://localhost:5173' has been blocked
by CORS policy: Response to preflight request doesn't
pass access control check: No 'Access-Control-Allow-
Origin' header is present on the requested resource.
```

Aquest error evidencia la importància de configurar correctament les capçaleres d'accés. Per solucionar-ho, s'ha definit la següent directiva, que especifica l'origen autoritzat i els mètodes HTTP permesos:

```
app.use(cors({
    origin: 'http://localhost:PORT',
    methods: ['GET', 'POST', 'PUT', 'DELETE'],
}));
```

Finalment, per millorar el desenvolupament, s'ha incorporat l'eina *Nodemon* [15], donat que incorpora HMR (*hot module reload*). Aquest consisteix a reiniciar automàticament el servidor cada vegada que es detecta un canvi en els fitxers del projecte. Això evita haver de parar i tornar a iniciar manualment el servidor després de cada modificació.

7.2.3 Frontend:

Pel que fa al *frontend*, s'ha escollit el framework *Vue.js* [10], seguint una arquitectura SPA (*Single Page Application*). Aquesta arquitectura millora l'experiència d'usuari, ja que redueix el nombre de recàrregues completes de pàgina i permet una navegació més fluida. L'aplicació s'organitza mitjançant SFC (*Single File Components*): components reutilitzables encapsulats en fitxers independents que integren la lògica (JavaScript), el contingut (HTML) i l'estil (CSS), facilitant la reutilització del codi.

Com que la navegació entre vistes no és nativa en l'arquitectura SPA, s'ha utilitzat *Vue Router* [16] per gestionar el sistema d'encaminament. L'us de *Vue Router* permet carregar els components en funció de la navegació en l'aplicació i, alhora, fa possible que l'usuari pugui usar la funcionalitat nativa de *history back* (navegar a la pàgina anterior) del navegador.

A més, el fet de disposar de diferents rutes, cadascuna associada a un URL específica, permet que les diferents parts de l'aplicació puguin ser indexades pels motors de cerca. Això contribueix a millorar el posicionament web (SEO) del projecte.

Finalment, s'han utilitzat variables de sessió per controlar l'accés a les diferents rutes i garantir que cada usuari només pugui accedir a les funcionalitats que li corresponen segons el seu rol i identificador. Abans de cada navegació, es valida que el rol de l'usuari coincideixi amb el requerit per la ruta i que l'identificador present a l'URL correspongui al de l'usuari autenticat. D'aquesta manera, s'evita, per exemple, que un professor amb *id* 1 pugui modificar manualment l'URL a */teacher/2* per accedir al contingut d'un altre professor. El fragment de codi corresponent es pot consultar a l'Apèndix A.5.

8 Presentació de resultats

En aquest apartat es presenten els resultats obtinguts durant el desenvolupament del projecte.

8.1 Creació base de dades

S'ha aconseguit crear i popular correctament la base de dades del sistema (Figura 6), emprant un fitxer *init.sql* per a la seva configuració inicial (vegeu Apèndix 3, A.3). La base de dades inclou totes les taules necessàries per a la gestió d'usuaris, assignatures, entregues, tasques i la relació entre estudiants, grups i subgrups.

```
Database changed
MariaDB [user]> show tables
    -> ;
+------------------+
| Tables_in_user   |
+------------------+
| Assignment       |
| Groups           |
| Student_group    |
| Subgroups        |
| Subgroups_student |
| Subgroups_task   |
| Subject          |
| Task             |
| Users            |
+------------------+
9 rows in set (0.001 sec)
```

Fig. 6: Taules de la base de dades executant `docker exec -it mariadb mariadb -u root -p`.

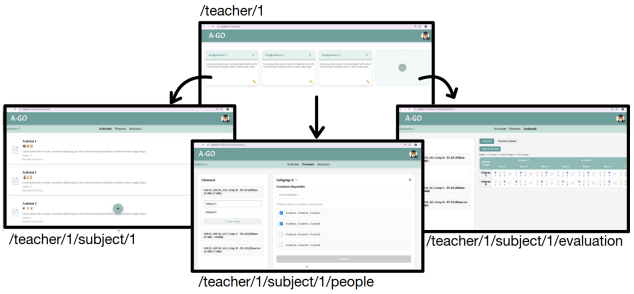


Fig. 7: Visualització de les rutes principals que defineixen la navegació del professorat dins la plataforma.

8.2 Frontend de la plataforma

El *frontend* de la plataforma està format per la interfície accessible tant per al docent com per a l'estudiant. Aquesta, com s'ha mencionat a la secció 7.2, s'ha desenvolupat amb *Vue.js* i s'ha estructurat mitjançant *Vue Router*, que ha permès organitzar la navegació en diferents rutes de l'aplicació web.

Per tal de mantenir la persistència de la sessió i el context de navegació, s'ha utilitzat l'API nativa *localStorage* [17] del navegador per emmagatzemar informació rellevant com l'identificador de l'usuari autenticat, l'assignatura activa o l'activitat seleccionada. Aquestes variables han permès també validar els paràmetres de les rutes, evitant accessos no autoritzats.

A continuació, es presenta el *routing* utilitzat per a la navegació específica de cada rol: docent i estudiant.

8.2.1 Routing de la navegació del professorat

S'ha aconseguit crear una interfície fidel al prototip inicial, definint diverses rutes que estructuren la navegació del professorat dins la plataforma, tal com mostra la Figura 7:

- **/login**: pantalla d'autenticació comuna per a tots els usuaris.
- **/teacher/:id**: panell principal del docent, des d'on es visualitzen les assignatures disponibles i se'n permet la creació de noves.
- **/teacher/:id/subject/:subject**: espai de gestió d'una assignatura concreta, que permet crear i organitzar les activitats.
- **/teacher/:id/subject/:subject/edit-assignment/:assignment**: entorn d'edició de tasques dins d'una activitat.
- **/teacher/:id/subject/:subject/evaluation**: espai destinat a l'avaluació de l'estudiant.
- **/teacher/:id/subject/:subject/people**: gestió dels participants de l'assignatura i assignació a subgrups.

8.2.2 Routing de la navegació de l'alumnat

La interfície destinada a l'alumnat segueix la mateixa estructura de navegació, però adaptada a les necessitats específiques d'aquest perfil (Figura 8). Les rutes principals definides són:

- **/login**: pantalla d'inici de sessió compartida amb el professorat.
- **/student/:id**: panell principal de l'estudiant, on es mostren les assignatures en què està matriculat i el subgrup al qual pertany.
- **/student/:id/subject/:subject**: visualització del progrés dins d'una assignatura, incloses les tasques i els elements de gamificació associats.
- **/student/:id/subject/:subject/edit**: espai de configuració que permet modificar l'avatar del subgrup.

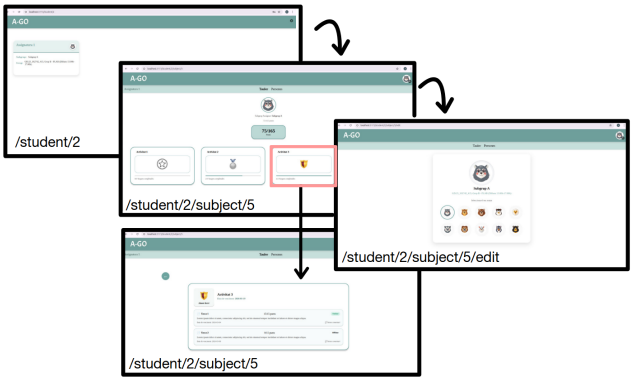


Fig. 8: Visualització de les rutes principals que defineixen la navegació de l'alumnat dins la plataforma.

8.3 Backend de la plataforma

El *backend* de la plataforma s'ha desenvolupat amb *Node.js* utilitzant el framework *Express*, que ha permès construir una API REST encarregada de gestionar la lògica per processar les peticions del *frontend* i garantir la comunicació amb la base de dades.

L'API s'ha organitzat segons el rol de l'usuari, diferenciant clarament les operacions destinades al professorat i a l'alumnat.

8.3.1 Endpoints de l'API

- **Peticions generals**
  - **/api/login (POST)**: Endpoint d'autenticació d'usuaris. Rep el correu o NIU i la contrasenya, i retorna les dades de l'usuari si la combinació és

correcta. El *frontend* emmagatzema l'usuari autenticat a *localStorage* per mantenir la sessió activa i controlar l'accés segons el rol.

- **Peticions destinades al docent**
  - **/api/teacher-dashboard (GET)**: Retorna totes les assignatures que gestiona un professor.
  - **/api/teacher-delete-subject (DELETE)**: Elimina una assignatura concreta a partir del seu identificador.
  - **/api/teacher-subject (GET)**: Retorna les activitats d'una assignatura determinada.
  - **/api/add-subject (POST)**: Permet afegir una nova assignatura i pujar un fitxer CSV amb el llistat d'estudiants.
  - **/api/add-assignment (POST)**: Crea una nova activitat.
  - **/api/teacher-edit-assignment (GET)**: Retorna les tasques d'una activitat concreta per a la seva edició.
  - **/api/add-task (POST)**: Crea una tasca dins d'una activitat i actualitza el recompte total de tasques.
  - **/api/update-task (PUT)**: Actualitza les dades d'una tasca existent.
  - **/api/delete-task (DELETE)**: Elimina una tasca concreta i actualitza el nombre total de tasques.
  - **/api/teacher-people (GET)**: Obté els grups, subgrups i estudiants associats a una assignatura.
  - **/api/create-subgroup (POST)**: Crea un nou subgrup dins d'un grup.
  - **/api/update-subgroup-name (PUT)**: Permet modificar el nom d'un subgrup existent.
  - **/api/delete-subgroup (DELETE)**: Elimina un subgrup.
  - **/api/teacher-evaluation (GET)**: Retorna les activitats amb les tasques i el seu estat per subgrup.
  - **/api/evaluate-task (PUT)**: Actualitza l'estat d'una tasca per a un subgrup concret.
  - **/api/save-task-comment (POST)**: Permet afegir un comentari a una tasca d'un subgrup.
- **Peticions destinades a l'alumnat**
  - **/api/student-dashboard (GET)**: Retorna totes les assignatures en què està matriculat l'estudiant.
  - **/api/student-subjects (GET)**: Retorna la informació general d'una assignatura per a un subgrup concret.
  - **/api/student-edit-avatar (PUT)**: Permet modificar l'avatar del subgrup dins d'una assignatura.

8.4 Comunicació client-servidor

A continuació es presenten un seguit d'exemples que mostren la correcta interacció entre el client i el servidor.

8.4.1 Importació d'estudiants a través d'un fitxer CSV

Des de l'inici del projecte es va considerar essencial simplificar el procés d'alta d'estudiants en una assignatura. Habitualment, el professorat ja disposa de llistats d'estudiants matriculats a una assignatura des del Campus Virtual. Per aquest motiu, s'ha implementat un sistema que permet importar tots els estudiants d'una assignatura mitjançant un fitxer CSV (vegeu Apèndix 4, A4, per al format del fitxer).

La Figura 9 ofereix una visualització de la dinàmica d'importació dels estudiants, així com la seva assignació als subgrups de treball.

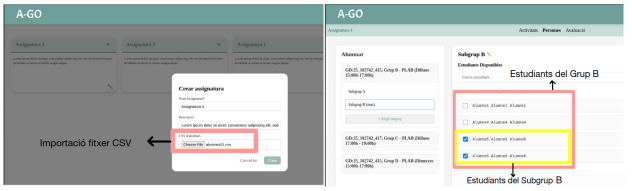


Fig. 9: Formulari de creació d'assignatures amb importació d'estudiants a través de CSV (esquerra) i visualització dels grups i subgrups de treball un cop importats (dreta).

Funcionament al frontend:

El professor pot adjuntar el fitxer CSV des del formulari de creació d'assignatures. Quan selecciona el fitxer local, aquest es captura a través de l'objecte *event* del navegador i s'emmagatzema temporalment.

Posteriorment, es construeix un objecte FormData, una interfície pròpia de JavaScript que facilita l'enviament de dades de formulari mitjançant HTTP. L'ús de JSON es va descartar perquè obligaria a codificar el fitxer i incrementaria la mida de la petició.

Un cop creat l'objecte FormData, s'envia al servidor a través d'una petició POST a l'endpoint */api/add-subject*. Aquesta comunicació es realitza de manera asíncrona, fet que permet a la interfície no quedar bloquejada mentre es processa la sol·licitud.

Funcionament al backend:

Per gestionar la pujada del fitxer CSV s'ha utilitzat el middleware Multer [12], una llibreria d'*Express* dissenyada per al tractament de les peticions multipart/form-data.

Quan el servidor rep la petició, Multer intercepta la sol·licitud, desa temporalment el fitxer al sistema i n'afegeix la informació a l'objecte *req.file*.

Un cop rebut el fitxer CSV, la funció el processa utilitzant el mòdul nadiu fs (File System [11]) de *Node.js*, que permet llegir arxius del sistema. En comptes de carregar tot el fitxer a la memòria, s'empra la funció asíncrona *createReadStream*, que el llegeix de manera progressiva. Aquesta estratègia és preferible especialment amb fitxers de gran mida. El flux de lectura es connecta a la llibreria *csv-parse* [18], encarregada d'interpretar el contingut del CSV.

8.4.2 Avaluació del professorat

S'ha aconseguit una correcta implementació del sistema d'avaluació. S'ha creat una vista estructurada en format de

taula, amb l'objectiu de facilitar el procés d'avaluació al docent així com la visualització del progrés dels estudiants tant pel professorat com alumnat (vegeu Figura 10).

Avaluació:

Per a cada tasca, el docent pot assignar un dels tres estats disponibles: *no entregat*, *entregat a temps* o *entregat fora de termini*. Quan es selecciona una d'aquestes opcions, el *frontend* envia una petició a l'endpoint */api/evaluate-task*, responsable d'actualitzar la base de dades.

En funció de l'existència prèvia del registre, el sistema crea una nova entrada a la taula *Subgroups_task* o bé n'actualitza el contingut. Això, permet evitar duplicats i garantir la persistència de la informació, assegurant que qualsevol modificació realitzada pel docent quedi registrada de manera immediata.

Visualització de resultats:

Més enllà de l'avaluació individual, la plataforma incorpora una vista d'*overview* que sintetitza el progrés global dels subgrups. Aquesta funcionalitat permet al docent identificar ràpidament el nivell d'activitat de cada equip, mentre que ofereix als estudiants una representació del seu rendiment (10).

Per construir aquesta vista, el *frontend* realitza una petició a l'endpoint */api/teacher-evaluation*, del qual obté dues estructures principals: *assignmentsTasks* i *taskStatusDict*. El primer diccionari descriu l'estructura acadèmica de l'assignatura:

```
assignmentsTasks = {
  id_assignment,
  name,
  data_venciment,
  insignia,
  descripcio,
  tasques_totals,
  tasks: [
    {
      id_task,
      name,
      punts,
      data_venciment,
      descripcio
    }
  ]
}
```

El segon diccionari emmagatzema l'estat real de cada tasca per subgrup i activitat:

```
taskStatusDict = {
  id_subgroup: {
    id_assignment: {
      estat,
      comentari
    }
  }
};
```

Un cop rebudes aquestes dades, el *frontend* les processa per generar els indicadors agregats de progrés. El procediment recorre cada subgrup i inicialitza una estructura destinada a emmagatzemar els resultats tant per una activitat com a escala global.

```
overviewTotal[subgroupId] = {
  assignments: {
    assignment_id: {
      doneTasks,
      totalTasks,
      donePoints,
      totalPoints
    }
  },
  totals: {
    doneTasks,
    donePoints
  }
};
```

A partir d'aquesta estructura es calculen el nombre de tasques completades, la insígnia corresponent i la puntuació acumulada. Cal destacar que els punts d'una tasca només computen quan aquesta ha estat entregada dins del període establert.

Així doncs, s'ha aconseguit implementar correctament l'avaluació permetent que el sistema no només reflecteixi la participació dels subgrups, sinó que també permeti un bon seguiment dels estudiants per part del docent.

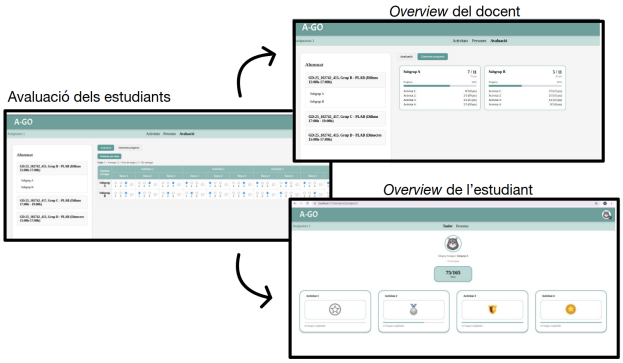


Fig. 10: Visualització de l'avaluació per part del docent, així com una visió global del progrés dels subgrups, tant per al docent com per a l'estudiant.

8.5 Creació d'entorns aïllats

Finalment, tal com mostra la Figura 11, s'ha aconseguit compilar i executar de manera aïllada el *backend* (*Node.js*), el *frontend* (*Vue.js*) i la base de dades (*MariaDB*) mitjançant Docker. S'han definit tres entorns independents que conformen l'arquitectura del sistema i permeten l'execució conjunta de tota la plataforma.

Fig. 11: Execució de la comanda `docker ps` per a la verificació de dockers aixecats.

9 Conclusions:

L'objectiu principal del projecte era doble: gamificar les entregues dels estudiants per augmentar la motivació i la participació, i alhora centralitzar la gestió d'informació acadèmica per facilitar al docent l'accés i seguiment del progrés de l'estudiant. Per aconseguir-ho, s'ha desenvolupat una plataforma web amb dos rols d'usuari, estudiants i professors, cadascun amb funcions específiques.

Enfocament en l'usuari: En primer lloc, s'ha creat un prototip funcional amb Figma, que ha estat avaluat per usuaris reals. Aquesta etapa ha permès definir un disseny adaptat a les necessitats tant dels estudiants com dels docents, i

ha servit de guia per al desenvolupament posterior del *frontend*.

Gestió d'usuaris i rols: Tot i que no s'ha integrat l'autenticació OAuth de la UAB, s'ha implementat un sistema d'inici de sessió intern amb control d'accés basat en rols. Aquest sistema, combinat amb la lògica de *routing* gestionada mitjançant *Vue Router*, ha permès que els estudiants accedeixin exclusivament a la seva vista personal, on poden editar el seu avatar i consultar el seu progrés. D'altra banda, els professors poden crear assignatures, activitats i tasques, associar-hi grups i subgrups de treball, avaluar-los, oferir *feedback* i fer un seguiment detallat del progrés dels alumnes.

Implementació de la dinàmica de gamificació: Un dels punts clau ha estat la materialització del sistema de puntuació. S'ha aconseguit diferenciar amb èxit dos tipus de participació:

- **Punts:** Premien els estudiants que completen les tasques a temps, fomentant el treball a l'aula.
- **Insígnies (Bronze, Plata i Or):** Motiven els alumnes a finalitzar les tasques d'una activitat encara que hagin superat la data de venciment per obtenir els punts, premiant així la constància i el progrés global.

Aquesta estructura permet que la plataforma es diferenciï d'altres entorns educatius més estàtics, oferint un incentiu continu per als alumnes.

Integració del sistema: Finalment, la plataforma s'ha desplegat en un entorn dockeritzat complet. Aquest entorn facilita l'execució de la plataforma i minimitza possibles problemes d'incompatibilitat en l'entorn de producció final.

En conclusió, la plataforma A-GO ha demostrat que és possible transformar la gestió acadèmica tradicional en una experiència més atractiva. A-GO no només simplifica les tasques administratives per al professorat, sinó que també utilitza la gamificació com a motor de compromís acadèmic per als estudiants.

10 Treball futur

Malgrat haver assolit la major part dels objectius plantejats, la plataforma presenta un ampli marge de creixement gràcies a la seva arquitectura escalable. Durant el desenvolupament s'han identificat diverses mancances així com millores que podrien optimitzar el rendiment del sistema i l'experiència d'usuari.

Actualment, la generació de l'*overview* o progrés dels subgrups requereix que el *frontend* processi les dades obtingudes del *backend*, provinents principalment de les taules *Task*, *Assignments* i *Subgroups_Task*. A partir d'aquestes dades es mostren les diverses insígnies i es computa la puntuació segons si les tasques s'han lliurat dins del termini establert. Aquest càlcul s'executa cada vegada que es refresca la pàgina. Per això, la creació d'una taula agregada seria una millora rellevant, ja que centralitzaria els càlculs i reduiria el processament al *frontend*.

Una altra millora prioritària és la integració amb el sistema d'autenticació OAuth de la Universitat Autònoma de Barcelona. A causa dels ajustos en la planificació, aquesta funcionalitat no s'ha pogut implementar. La seva incorporació permetria als estudiants iniciar sessió amb el seu NIU i la contrasenya institucional, evitant la creació de comptes específics per a la plataforma-

Durant l'ús de la plataforma també s'ha detectat una possible millora en el procés d'assignació d'estudiants als subgrups. Inicialment, no es contemplava aquesta necessitat, però l'experiència d'ús ha posat de manifest la conveniència d'implementar un filtre en la visualització dels estudiants. És a dir, si un estudiant pertanyent a un grup ja està assignat a un subgrup, no hauria d'aparèixer en el llistat corresponent, amb l'objectiu de reduir possibles errors d'assignació.

Paral·lelament, tal com es plantejava en el prototip inicial, es preveia incorporar un sistema de rànquings setmanals amb l'objectiu d'incrementar la motivació de l'alumnat. Aquesta funcionalitat havia d'estar ubicada a l'apartat de persones d'una assignatura i havia de mostrar tant un rànquing global de punts entre tots els grups, així com la posició específica en el propi grup de l'estudiant.

Finalment, cal remarcar que el projecte actualment es troba en un entorn de desenvolupament i que seria necessari realitzar el desplegament en un entorn de producció. Una de les modificacions imprescindibles seria la implementació del protocol HTTPS en substitució de l'HTTP utilitzat actualment, amb l'objectiu de garantir la confidencialitat de les dades transmeses. Així mateix, també seria recomanable revisar el mecanisme d'autenticació, ja que actualment la informació de sessió s'emmagatzema al *localStorage*. Tot i que aquesta és una solució habitual en fases de desenvolupament, en un entorn de producció podria exposar l'aplicació a vulnerabilitats com els atacs XSS (Cross-Site Scripting).

Referències

[1] K. Schwaber i J. Sutherland, *Scrum Guide*, 2025. Disponible: `https://www.scrumguides.org/scrum-guide.html`

[2] Socrative, "Socrative: Student Response System", 2025. Disponible: `https://www.socrative.com`

[3] Kahoot!, "Kahoot! Learning Platform", 2025. Disponible: `https://kahoot.com`

[4] Atlassian, "Jira Software", 2025. Disponible: `https://www.atlassian.com/software/jira`

[5] ProjectLibre, "ProjectLibre: Open Source Project Management", 2025. Disponible: `https://www.projectlibre.com/`

[6] Figma, "Figma: Collaborative Interface Design Tool", 2025. Disponible: `https://www.figma.com`

[7] Dineug, "ERD Editor VSCode Extension", 2025. Disponible: `https://marketplace.visualstudio.com/items?itemName=dineug.vuerd-vscode`

[8] Node.js Foundation, "Node.js JavaScript Runtime", 2025. Disponible: https://nodejs.org

[9] Express.js Foundation, "Express - Node.js Web Application Framework", 2025. Disponible: https://expressjs.com

[10] Vue.js, "Vue.js - The Progressive JavaScript Framework", 2025. Disponible: https://vuejs.org

[11] Node.js, "File system - Node.js Documentation", 2025. Disponible: https://nodejs.org/api/fs.html

[12] Express.js, "Multer middleware", 2026. Disponible: https://expressjs.com/en/resources/middleware/multer.html

[13] MariaDB Foundation, "MariaDB Server Documentation", 2025. Disponible: https://mariadb.org/documentation/

[14] Express.js, ÇORS middleware", 2026. Disponible: https://expressjs.com/en/resources/middleware/cors.html

[15] Remy Sharp, "Nodemon", 2025. Disponible: https://nodemon.io/

[16] Vue.js, "Vue Router - The official router for Vue.js", 2025. Disponible: https://router.vuejs.org/

[17] Mozilla Developer Network, "Web Storage API - localStorage", 2026. Disponible: https://developer.mozilla.org/docs/Web/API/Web_Storage_API#localstorage

[18] CSV Parse, çsv-parse: Node.js CSV parser", 2026. Disponible: https://csv.js.org/parse/

A APÈNDIX

A.1 Esquema complet de la base de dades

A continuació, es mostra la versió ampliada de l'esquema relacional de la base de dades desenvolupada per a la plataforma *A-GO*. Aquesta figura inclou totes les taules, relacions i claus externes amb més detall que la versió mostrada la Figura 4 .

Fig. 12: Esquema complet de la base de dades de *A-GO*.

A.2 Prototip interactiu a Figma

El prototip funcional de la plataforma es pot consultar a Figma[2]. Aquest prototip permet navegar entre les diferents pantalles i comprovar la interacció entre els rols de professorat i alumnat, així com els elements de gamificació implementats.

A.3 Fitxer de configuració `init.sql`

El fitxer *init.sql* s'ha utilitzat per a la creació i població inicial de la base de dades. A continuació es mostra un extracte representatiu del seu contingut, amb algunes de les taules principals i les relacions definides:

```
CREATE TABLE Users (
  id INT NOT NULL AUTO_INCREMENT,
  nom VARCHAR(100) NOT NULL,
  email_niu VARCHAR(150) NOT NULL,
  tipus ENUM('alumne','professor') NOT NULL,
  password VARCHAR(250) NOT NULL,
  PRIMARY KEY (id),
  UNIQUE (email_niu)
);

CREATE TABLE Subject (
  id INT NOT NULL AUTO_INCREMENT,
  nom VARCHAR(100) NOT NULL,
  descripcio VARCHAR(400) NULL,
  id_teacher INT NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_Users_TO_Subject
    FOREIGN KEY (id_teacher)
    REFERENCES Users(id)
    ON DELETE RESTRICT
);

CREATE TABLE Assignment (
  id INT NOT NULL AUTO_INCREMENT,
  nom VARCHAR(100) NOT NULL,
  data_venciment VARCHAR(25) NOT NULL,
  descripcio VARCHAR(400) NULL,
  id_subject INT NOT NULL,
  tasques_totals INT NOT NULL,
  insignia INT NOT NULL,
  sessions INT NOT NULL,
  PRIMARY KEY (id),
  CONSTRAINT FK_Subject_TO_Assignment
    FOREIGN KEY (id_subject)
    REFERENCES Subject(id)
    ON DELETE CASCADE
);
...
```

El fitxer complet conté totes les taules, claus primàries, claus externes i relacions necessàries per al correcte funcionament de la base de dades, així com les restriccions i índexs.

A.4 Fitxer CSV per a importació d'usuaris

Per a la creació automàtica d'usuaris estudiants, s'utilitzen fitxers CSV amb un format predefinit. A continuació es mostra un exemple representatiu:

```
Nom,Cognoms,"Número ID",Grups
Alumne1,"Alumne1 Alumne1",1111,"GD:25_102742_415, Grup B - PLAB (Dilluns 15:00h-17:00h)"
Alumne2,"Alumne2 Alumne2",2222,"GD:25_102742_417, Grup C - PLAB (Dilluns 17:00h - 19:00h)"
Alumne3,"Alumne2 Alumne2",3333,"GD:25_102742_417, Grup C - PLAB (Dilluns 17:00h - 19:00h)"
```

A.5 Protecció de rutes amb *Vue Router*

Per garantir la seguretat de l'aplicació i evitar accessos no autoritzats, s'ha implementat un mecanisme de protecció de rutes mitjançant *Vue Router* i l'ús de variables de sessió. Aquest sistema permet validar que l'usuari autenticat disposi del rol adequat per accedir a la vista sol·licitada i que l'identificador present a l'URL coincideixi amb el seu.

Aquest control s'aplica abans de cada navegació mitjançant la funció `beforeEach`. En cas que el rol de l'usuari no coincideixi amb el requerit per la ruta, el sistema el redirigeix a la pantalla de *login*. De la mateixa manera, si l'identificador de la ruta no correspon a l'usuari autenticat, es redirigeix automàticament al seu tauler principal.

A continuació es mostra el fragment de codi que implementa aquesta lògica:

```
router.beforeEach((to, from, next) => {
  const user = getLoggedUser()

  // Comprova si el rol coincideix amb el requerit per la ruta
  if (to.meta.role && user?.tipus !== to.meta.role) {
    return next('/login') // Pantalla Login
  }

  // Comprova si l'id de la ruta correspon a l'usuari autenticat
  if (to.params.id && String(user?.id) !== to.params.id) {
    return next(`/${user.tipus}/${user.id}`) // Pantalla Tauler
  }

  next()
})
```

[2] https://www.figma.com/site/McHiuHSu093InRS1QZmhwD/A-GO-TFG?node-id=0-1&t=W29Z4mROHLD8h31d-1