WEB DE GENERACIÓ D'EXAMENS DE PROVA

Gabriel Franco Morales

Resum—Resum del projecte, màxim 10 línies

En el context educatiu actual, la col·laboració entre professorat i alumnat és fonamental per millorar els processos d'ensenyament i aprenentatge. A la fase d'avaluació, però, les eines existents per a la creació de proves presenten limitacions importants, com la creació unidireccional de tests, on només el professorat és responsable, fet que limita la participació activa de l'alumnat i la diversitat de continguts. Aquest projecte proposa una plataforma web col·laborativa on professorat i alumnat poden crear preguntes d'examen i adaptar tests a les necessitats individuals. Això millorarà la qualitat del contingut i fomentarà la implicació activa de tots els participants en el procés educatiu, ja sigui mitjançant la creació de preguntes o la consolidació de coneixements amb la pràctica.

Paraules clau—Paraules clau del projecte, màxim 2 línies- Context educatiu, Plataforma web col·laborativa, HTTP, MERN, Frontend, Backend, Node JS, Express JS, React JS, Component, Sprint i Backlog, Bases de Dades, MySQL,Tests

Abstract—Versió en anglès del resum

In the current educational context, collaboration between teachers and students is essential to improve teaching and learning processes. In the evaluation phase, however, the existing tools for creating tests present important limitations, such as the unidirectional creation of tests, where only the teaching staff is responsible, which limits the active participation of students and the diversity of contents This project proposes a collaborative web platform where teachers and students can create exam questions and adapt tests to individual needs. This will improve the quality of the content and encourage the active involvement of all participants in the educational process, either by creating questions or consolidating knowledge with practice.

Index Terms—Versió en anglès de les paraules clau- - Educational context, Collaborative web platform, HTTP, MERN, Frontend, Backend, Node JS, Express JS, React JS, Component, Sprint and Backlog, Databases, MySQL, Tests

1 Introducció - Context del treball

En el context educatiu actual, la col·laboració entre professorat i alumnat és essencial per millorar tant els processos d'ensenyament com els d'aprenentatge. Tot i això, les eines existents per la creació de proves que permetin l'avaluació de l'alumnat i el seu progrés presenten una sèrie de limitacions fonamentals, tals com la creació de proves i tests de forma unidireccional, on el professorat assumeix tota la responsabilitat, limitant la participació activa de l'alumnat i la diversitat de continguts.

Aquest projecte proposa superar aquestes barreres mitjançant una plataforma web col·laborativa on professorat i estudiant de qualsevol institució puguin crear preguntes d'examen i adaptar tests de pràctica a les necessitats individuals. Això no només enriquirà la qualitat del contingut, sinó que també fomentarà la implicació activa de tots els participants en el procés educatiu, ja sigui mitjançant la proposició de preguntes, o bé en la consolidació dels conceptes mitjançant la pràctica.

- *E-mail de contacte: gabrifrancoaranda@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Javier Sanchez Pujades(Ciències de la Computació)*
- *Curs 2024/25*

2 Objectius del projecte

Com s'ha esmentat a l'apartat anterior, l'objectiu principal del projecte consisteix en la creació d'una **plataforma web** que permeti la **col·laboració** entre professorat i alumnat d'una institució acadèmica alhora de generar proves pràctiques, fomentant la participació activa dels implicats i millorant la consolidació dels conceptes estudiats. A continuació, es desglossen els objectius específics segons la seva naturalesa:

2.1 Objectius Tècnics

- Desenvolupar una plataforma web funcional.

- Permetre el registre i gestió segura dels seus usuaris, diferenciant els rols i les seves funcions.
- Crear una interfície intuïtiva que permeti l'execució de funcions com la gestió de les preguntes proposades de manera senzilla i ràpida.
- Dissenyar una base de dades escalable i eficient.
- Incorporar un sistema de pràctica de conceptes segons les necessitats de l'alumne.

2.2 Objectius Personals

- Entendre i aprendre a crear una aplicació web en totes les seves fases.
- Aplicar els meus coneixements adquirits durant la carrera.
- Desenvolupar un coneixement i pràctica amb eines noves tals com **Node JS, React JS i MySQL**.
- Millorar en el procés de planificació de les tasques.

3 Estat de l'art

Abans d'iniciar qualsevol projecte, és necessari efectuar una recerca prèvia del mercat per obtenir idees. Així doncs, les principals fonts d'informació que es van prendre per referència han estat les següents:

3.1 Moodle [1]

Plataforma de gestió d'aprenentatge per exel·lència. Permet crear espais virtuals altament personalitzables amb diverses eines com compartir materials educatius, gestionar cursos... Si bé presenta eines per l'avaluació de l'alumnat, restringeix la seva col·laboració en un alt grau i la creació de tests o proves té un cert grau de dificultat.

Característiques interessants a tenir en compte: la seva gestió de cursos i participants.

3.2 Peer Wise [2][3]

Plataforma web creada per permetre als estudiants crear, compartir i avaluar preguntes d'examen de selecció múltiple. El seu objectiu es limita a l'intercanvi, resolució i avaluació de les preguntes proposades només per alumnes. Aquest fet suposa que no hi ha una font certament objectiva que permeti avaluar si les respostes i raonaments dels alumnes són del tot encertarts.

Característiques interessants a tenir en compte: mètode d'avaluació.

3.3 Kahoot! [4]

Plataforma web que permet al professorat crear qüestionaris de forma simple, permetent la seva participació a alumnes amb un dispositiu i, sovint, una clau específica. L'alumnat com a tal segueix en una posició on només ha de respondre les preguntes, no realitzar-les.

Característiques interessants: format dels qüestionaris, respostes en temps real.

3.4 ProProfs Quiz Maker

Plataforma en línia que permet al usuari crear qüestionaris personalitzats. El seu punt fort és la varietat de preguntes disponibles i la visualització. Segueix limitant els continguts al propi usuari, sense intervenció de fonts externes com podria ser l'alumnat.

Característiques interessants: retroalimentació d'alguns qüestionaris, assignació de punts i reotralimentació automàtica.

4 Metodologia [5]

La metodologia emprada per a la creació d'aquest projecte ha estat **Agile Scrum**, tot i que adaptada al context d'un sol desenvolupador. Cal destacar que **Scrum** està dissenyada per ser aplicada en equips amb diversos membres, i per aquest motiu, s'han seleccionat i ajustat únicament aquells conceptes que poden ser gestionats de manera individual. S'han omès els elements que requereixen necessàriament la col·laboració entre múltiples participants.

A nivell global, el projecte segueix un procés de desenvolupament basat en el model **Waterfall**, estructurat en fases seqüencials i preestablertes: anàlisi de requisits, disseny, implementació, testeig i desplegament. Aquestes fases segueixen un ordre inalterable, permetent un flux clar i organitzat del treball.

Dins de cada fase, el treball es divideix en períodes de temps delimitats anomenats **Sprints**, amb una durada d'una a dues setmanes. A l'inici de cada **Sprint** es defineixen les tasques a realitzar en un **Sprint Backlog**, que permet fer un seguiment detallat de l'estat de cada activitat, classificades en tres categories: Per Fer, En Progrés i Finalitzat.

En concloure cada **Sprint**, es realitza una revisió exhaustiva de les tasques completades i de les pendents. A partir dels resultats d'aquesta anàlisi, s'ajusta la planificació per minimitzar l'impacte de les tasques no finalitzades en el calendari global del projecte, garantint així una gestió eficient i flexible del desenvolupament.

5 Planificació

La planificació global de tot el projecte ha près de marc global de temps desde el 12 de Septembre de 2024 fins el 4 de Febrer de 2025.

Com s'ha comentat amb anterioritat, ha seguit una sèrie de fases predefinides per la metodologia **Waterfall**, formades per cicles de treball d'una o dues setmanes anomenades **Sprints**. Segons la fase i les modificacions que s'han dut a terme durant la realització de tot el treball, la planificació final del projecte ha estat la següent:

Fase Anàlisi de Requisits:

- **Sprint 1 (12/09 – 21/09)** – Anàlisi de requisits i

definició d'objectius.

Fase de Disseny:

- **Sprint 2 (24/09 – 05/10)** – Disseny de l'arquitectura SW, BD...
    - Entrega Informe Inicial (06/10).

Fase de Desenvolupament:

- **Sprint 3 – 7 (07/10 – 31/12)** – Desenvolupament de tots els requisits funcionals i no funcionals.
    - Entrega Informe de Progrés I (10/11).
    - Entrega Informe de Progrés I I(15/12).

Fase de Testeig:

- **Sprint 8 (31/12 – 10/01)** – Testeig, depuració de codi i detalls.

Fase Desplegament / Entrega:

- **Sprint 9 (10/01 – 19/01)** – Conclusions i entrega del codi.
    - Entrega Proposta Informe Final (19/01).
- **Sprint 10 (21/10 – 04/02)**
    - Entrega Proposta Presentació (02(04).
    - Entrega Final (04/02).

5.1Eines de planificació

L'eina emprada per la realització de la planificació:

JIRA Software [6]: És una eina que, tot i estar dissenyada principalment per a equips, és perfectament adaptable a un sol desenvolupador. Ha estat utilitzada per planificar, organitzar i fer el seguiment del treball mitjançant taulers **Scrum** i **Kanban**, que permeten dividir el projecte en tasques específiques i assignar-les a diferents **Sprints**. Així mateix, la funcionalitat del **Backlog** facilita la priorització de les activitats i la visibilitat del progrés del projecte. **JIRA**

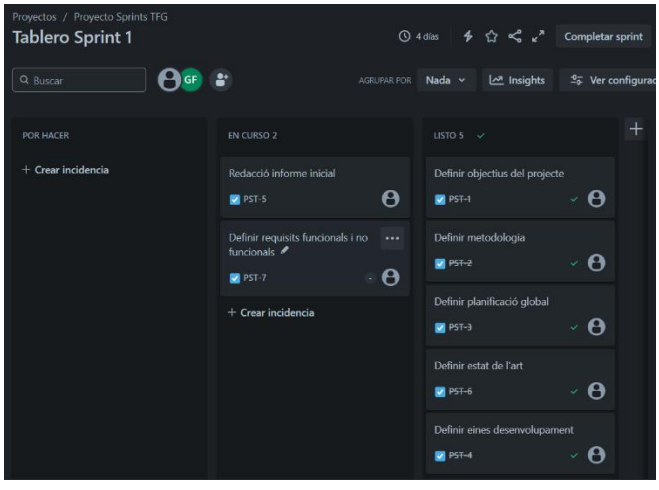


Fig.1: Backlog Sprint

també ofereix eines d'informes i mètriques que han estat útils per monitoritzar el ritme de treball i ajustar la planificació segons les necessitats.

6 ANÀLISIS I REQUISITS

La fase d'anàlisis de requisits es va dividir en una sèrie de processos que van ajudar a definir els objectius i requisits del projecte:

1. **Entrevista amb el tutor i persones de l'àrea del ensenyament:** Es van efectuar reunions amb el tutor i amb coneguts de l'àrea de l'enseyament per tractar de definir aquells requisits que hauria d'haver implementats al projecte.
2. **Revisió i el laboració de l'estat de l'art:** Amb una recerca prèvia al inici del projecte, es van poder observar altres aplicacions i plataformes web que presenten característiques similars, recollint aquelles features més importants.
3. **Brainstorming:** Com a alumne, jo mateix vaig fer nombroses pluges de idees per tal de definir les features més importants que caldria implementar per oferir un software de qualitat.

En conjunt, es van obtenir nombrosos requisits. Davant la gran quantitat d'elements a implementar, es va reduir el seu número i es van classificar segons la seva naturalesa per oferir un software de qualitat.

A continuació, es presenta una llista dels requisits més importants, identificats per un codi identificador, on s'entén **RF** com Requisit funcional i **RNF** com Requisit No Funcional i la seva prioritat:

RF-1-Gestió d'usuaris: La plataforma web ha de permetre registrarse i iniciar sessió tant als professors com els alumnes, diferenciant ambdós rols i permetent recuperar l'estat en el que es va fer *logout*. Ha de permetre realitzar canvis en les seves dades personals un cop tingui accés. Finalment, l'usuari ha de poder eliminar el seu compte.

RF-2-Gestió del temari: La web ha de permetre als professors poder crear i eliminar assignatures i temes i poder afegir i eliminar els participants que hi vulguin dins de cada assignatura.

RF-3-Gestió de les preguntes: La web ha de permetre als professors formular i gestionar les preguntes dels exàmens que es generen a la plataforma. Els alumnes també poden proposar preguntes que s'inclouen als exàmens, subjectes a una prèvia avaluació per part del professorat. Hi haurà un apartat dins la web on els professors podran veure les preguntes formulades pels alumnes (poden aparèixer als tests) i les preguntes ja acceptades. Ha de permetre als professors editar també les preguntes que són al banc de preguntes.

RF-4-Generació exàmens de prova alumnat: L'aplicació ha de poder generar exàmens de prova de forma automàtica a partir d'una base de dades. Podran escollir entre escollir els conceptes a avaluar o un test reactiu que genera preguntes en temps real segons les respostes de l'usuari.

RF-5-Generació exàmens prova professorat:
El professorat podrà generar exàmens de forma automàtica o manual per tal de que els alumnes practiquin. Podran realitzar-ho en un àmbit de pràctica o bé avaluatiu, enregistrant les notes en aquest últim.

RF-6-Base de dades: La base de dades ha de ser fàcil de gestionar i ha de permetre unes relacions entre taules sòlides i coherents, evitant qualsevol tipus de problema entre els registres. Ha de permetre simular un entorn acadèmic real.

7 Disseny

Un cop es va realitzar l'anàlisi de requisits, es va començar a dissenyar el software utlitzant diverses eines i artefactes. Entre els més importants, podem destacar:

Diagrama Casos d'Ús [7]: Diagrama centrat en identificar requisits funcionals i la seva relació, limitar l'abast del sistema i identificar i definir que fa cada actor implicat. Aquests diagrames van ajudar a definir de forma més consistent que s'havia de desenvolupar.

Diagrama Entitats-Relació: Diagrama que reflexa la base de dades emprada al projecte. Va ajudar a representar de forma simple i visual les entitats i relacions implicades en el desenvolupament del projecte. Durant el transcurs del desenvolupament del projecte, aquest diagrama va patir canvis.

Diagrama UI: Diagrama emprat per establir un disseny visual i una relació entre les pantalles que la web hauria de mostrar. Un exemple a continuació:

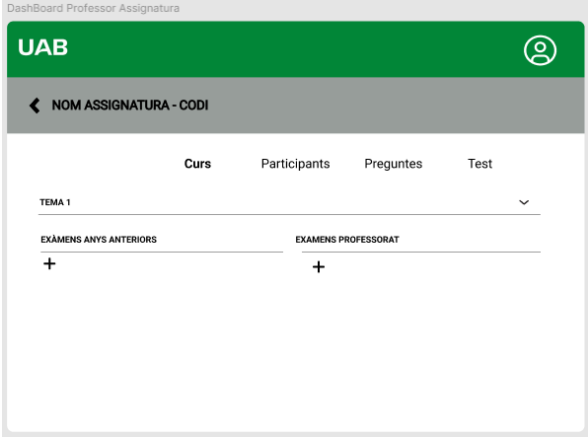


Fig. 2: Exemple pantalla assignatura

8 Desenvolupament

8.1 Eines de Desenvolupament

Les eines emprades en el desenvolupament de l'aplicació són les següents:

Github [8]: Eina de control de versions per excel·lència. Es va decidir utilitzar en format extensió de **Visual Studio Code** degut a que ofereix un control de versions simplificada, accessible des de l'editor de codi i és ràpida i segura. Segons les features que es desenvolupaven i que tenien un alt risc, es considerava crear una branca auxiliar on dur a terme els canvis. En cas contrari, tot es desenvolupava a la branca **master.**

Visual Studio Code [9]:
Editor de codi font lleuger desenvolupat per **Microsoft**. És una eina multiplataforma que suporta una gran varietat de llenguatges i permet una alta personalització. Es va escollir per aquest projecte degut a les següents raons:

1. **Suport a JS:** Degut a que es tenia planejat emprar **JavaScript** com a llenguatge de desenvolupament, es va optar per utilitzar aquest editor de codi.
2. **Familiaritat:** Degut a que es un entorn fàcil d'utilitzar i ja es tenia experiència prèvia.
3. **Alta personalització i extensions:** Té una alta capacitat de personalització i extensions que ajuden a realitzar el codi de forma més simple i llegible.

MySQL Workbench [10]:
Eina d'administració, modelat i consulta de la base de dades **MySQL.** Va ser escollida degut a les següents raons:

1. **Modelat de base de dades:** Permet crear models d'entitat i relació de forma simple, permetent la seva planificació.
2. **Consulta de dades:** Té un editor SQL integrat que permet executar **consultes** per modificar, afegir o eliminar dades.
3. **Exportació i importació:** Permet importar informació de forma senzilla en diversos formats.

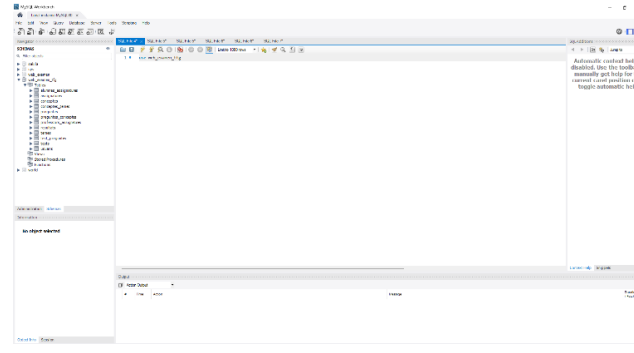


Fig. 3: Entorn MySQL Workbench

També permet exportat esquemes de bases de dades senceres per facilitar el seu desplegament en altres dispositius.

8.2 Entorns, Framewoks, Llenguatges Biblioteques de Desenvolupament

Com s'ha comentat a l'apartat anterior, el codi desenvolupat ha estat realitzat mitjançant **JavaScript** tant en l'apartat de **Frontend** com en el de **Backend** i en **SQL** en l'apartat de la base de dades. Així doncs, l'entorn emprat pel desenvolupament de l'aplicació en el **Backend** ha estat **Node JS**

juntament amb **Express JS.**

NODE JS [11]:
Node JS es considera un dels entorns d'execució per excel·lència degut a que permet executar codi fora del navegador. Va ser dissenyat per oferir una plataforma potent, escalable i asíncrona per desenvolupar aplicacions de xarxa i altres tipus d'aplicacions **Backend**. Les seves principals característiques son les següents:

Basat en JavaScript: Al utilitzar el mateix llenguatge que s'empra per desenvolupar aplicacions al navegador, permet que els desenvolupadors puguin treballar amb un únic llenguatge tant al **Frontend** com al **Backend**.

Asíncron no Bloquejant: Les operacions d'entrada i sortida (E/S), com accedir a fitxers, fer consultes a bases de dades o sol·licituds de xarxa, es processen de manera asíncrona. Això vol dir que el servidor no s'atura mentre espera una resposta, sinó que continua gestionant altres tasques. Això és ideal per manejar milers de connexions simultànies.

Gran Ecosistema: **Node JS** compta amb un gestor de paquets, el **Node Package Manager (NPM)**, que proporciona accés a milions de biblioteques de tercers que permeten afegir funcionalitats a les aplicacions de manera senzilla.

EXPRESS JS [12]:
Tot i que **Node JS** ofereix una base molt robusta, sovint pot ser complex treballar amb ell directament quan es tracta de desenvolupar aplicacions web. Aquí entra **Express JS**, un framework minimalista dissenyat per facilitar el desenvolupament de servidors i APIs:

Facilita la creació de rutes: Express simplifica la definició de rutes per gestionar sol·licituds **HTTP**. Amb **Express**, es pot crear una ruta en una sola línia de codi com es pot veure a continuació:

```
const http = require('http');

const server = http.createServer((req, res) => {
  if (req.url === '/' && req.method === 'GET') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello World');
  }
});

server.listen(3000, () => {
  console.log('Servidor escoltant al port 3000');
});
```

Fig. 4: Node JS Server

```
const app = express();
app.use(express.json());
app.use(cors({
    origin: 'http://localhost:5173', //origen específic
    methods: ['GET', 'POST', 'PUT', 'DELETE'],// Metodes permesos
    credentials: true // Credencials necessaris
}));


app.use(cookieParser());  //Cookies

app.get('/', (req, res) => { //Funció Hello World
    res.send('Hello World');
  });


//Funció d'escolta del servidor
app.listen(8081, () => {
    console.log("Running Server...");
});
```

Fig. 5: Node JS + Express JS

Com es pot comprovar, la creació d'un servidor mitjançant **Express JS** es més fàcil d'entendre i ofereix eines addicionals, tals com **CORS [13]. CORS** és un middleware que s'utilitza en aplicacions de **NodeJS** (normalment amb **ExpressJS**) per habilitar i configurar fàcilment **CORS** (Intercanvi de Recursos entre Orígens Creuats). Aquesta llibreria permet especificar quins orígens estan autoritzats a accedir als recursos del servidor i quines sol·licituds són permeses.

Middlewares: Express JS també incorpora **middleware,** que són funcions que s'executen abans que una sol·licitud arribi a la seva ruta final. Això permet afegir funcionalitats com autenticació, registre de sol·licituds, validació o manipulació de dades de manera modular i fàcil.

REACT JS [14][15]:
Alhora de la creació del **Frontend,** es van barallar les opcions més populars del moment: **Angular, React JS, Vue JS...**

Finalment es va escollir **React JS. React JS** és una biblioteca **JavaScript** desenvolupada per **Facebook**, dissenyada per construir interfícies d'usuari (UI) d'alt rendiment i interactives. S'utilitza principalment per al desenvolupament d'aplicacions web de pàgina única (**SPA, Single Page Applications**) i s'ha convertit en una de les tecnologies més populars del desenvolupament. Els principals avantatges són els següents:

Basats en Components: React divideix la interfície d'usuari en petits blocs reutilitzables anomenats **components**. Cada **component** encapsula la seva pròpia lògica i estil, cosa que facilita la seva reutilització i manteniment.

Flexibilitat i ecosistema obert: al ser una biblioteca centrada en el desenvolupament d'interfícies d'usuari, permet triar exactament quines biblioteques o eines complementàries vols utilitzar.

Corva d'aprenentatge senzilla: Al emprar **JSX,** una combinación entre **HTML** i **JavaScript,** si es dominen les bases d'ambdós llenguatges es fàcil d'aprendre.

Rendiment DOM: Al estar realitzant un projecte on els canvis són freqüents, l'ús d'un virtual **DOM** fa que les actualitzacions al **DOM real** siguin eficients.

Modularitat i manteniment: Les aplicacions de **React** són més fàcils de mantenir perquè es pot actualitzar només les parts necessàries, gràcies al seu model basat en components.

Adaptabilitat i integració: Al no tenir una arquitectura estricta, **React** permet integrar-se amb moltes altres eines i tecnologies.

Altres llibreries/Hooks Importants:

mysql: Llibreria per connectar-se a una base de dades **MySQL** i realitzar operacions com consultes i insercions**.**

jsonwebtoken: S'utilitza per crear i verificar tokens JWT, que serveixen per autenticar usuaris de manera segura.

bcrypt: Llibreria per encriptar contrasenyes abans de guardar-les a la base de dades, millorant la seguretat. En el cas del projecte s'empra per **hash**.

cookie-parser: Facilita la gestió de cookies **HTTP**, útil per guardar i llegir tokens d'autenticació com el JWT.

axios: Llibreria per fer peticions **HTTP**, sovint utilitzada per interactuar amb APIs des de **React**. És el que permet el pas d'informacioó entre **Frontend i Backend.**

react-icons: Llibreria de **React** que ofereix accés a icones vectorials. Exemples com les icones d'usuari o la fletxa de retorn.

8.3 Procés de Desenvolupament

Al tractar-se d'un projecte amb no massa complexitat, es va decidir aplicar el patró d'arquitectura **MERN [16].**

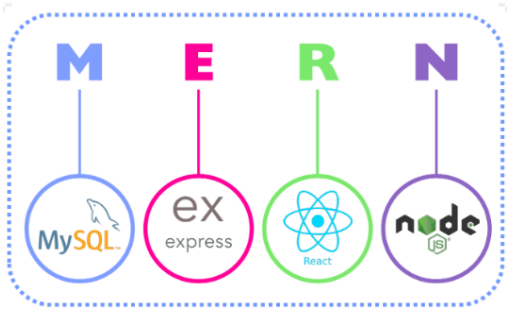


Fig. 6: Arquitectura MERN

Aquest patró simplifica molt l'estrucutura de l'aplicació. Hi ha un **Frontend** basat en **React** el qual s'encarrega de rebre tota la informació provinent de l'usuari, tals com els inputs, recuperació de informació... A continuació, aquesta informació es tractada i adaptada a les necessitats del servidor a través de funcions i variables, evitant així conflictes entre el tipus de dades, atacs per **Cross Scripting,** inserció de variables erronies**...** Mitjançant la llibreria **Axios,** es permet enviar la data al **Backend** basat en **Node JS + Express JS.**

Un cop al servidor i gràcies a les funcions definides, la data torna a ser tractada per evitar qualsevol tipus de problemàtica. A continuació, mitjançant la mateixa llibreria **Axios** i **mysql,** es realitzen peticions parametritzades a la base de dades, evitant així atacs per **SQL Injection** i facilitant el treball als desenvolupadors.

Un cop recuperada la data, aquesta s'envia de tornada al **Frontend,** on es tornada a tractar i es adaptada segons el que es requereixi.

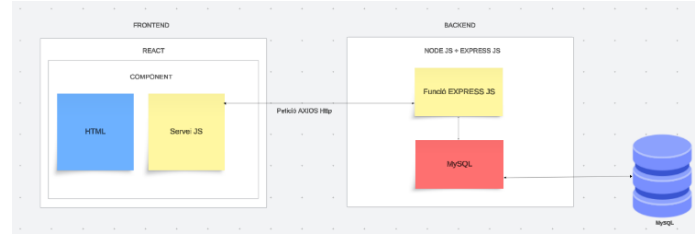


Fig. 7: Arquitectura SW

El procés d'implementació de les funcionalitats es va dividir en tres fases principals:

Anàlisi de relacions dels requisits:
Es va definir un esquema inicial per identificar les dependències entre els requisits i la base de dades, així com les funcions necessàries per implementar-los. **Exemple**: El requisit **RF-1 (Gestió d'usuaris)** requeria accions com registre, login, edició de dades i eliminació de comptes, amb dependències clares entre aquestes funcionalitats.

Definició del desenvolupament:
Es va establir un pla d'acció basat en la jerarquia de dependències. Per exemple, l'edició del perfil només era possible després d'un registre i un login, iniciant el desenvolupament pel registre.

Definició dels components i necessitats:
Es va prioritzar el desenvolupar el **Frontend** sobre el **Backend**. **Exemple:** Per al registre, es desenvolupava primer una interfície d'entrada de dades, seguida d'una funció al **Backend** per processar-les.

Creació Frontend:
La creació del **Frontend** constava de dues parts. La primera, la part visual. Al poder utilitzar **components**, molts cops es definia un **component** principal (sovint constituïa tota la pàgina i els seus elements específics), i després, segons les necessitats, es definien **components auxiliars.** Aquests sovint eren elements que podien ser reutilitzats en diverses pàgines, tals com una **capçalera.**

La segona part era la creació de funcions que permetien la connexió amb el servidor i el posterior tractament de

dades. Normalment per cada **component,** hi ha una funció **useEffect,** la qual s'executa sempre que canvien els valors d'algunes variables definides. Així doncs, casi sempre la pàgina executa una funció alhora de la seva renderització que permet recuperar dades vitals pel seu mostreig, com podria ser per exemple **l'edició de dades d'usuari,** on es vital veure quins son els valors de l'usuari en el moment previ abans de la seva edició.

Altres funcions només executen accions de suport, com el filtratge de dades o bé, en el cas d'aquest projecte, accions més importants com la revisió i enviament d'un **feedback** immediat a l'usuari segons les seves decisions.

Creació Backend:
La creació del **Backend** es basava en les necessitats dels **components** del **Frontend.** Segons si aquestes eren de recuperació, actualització... de dades de la base de dades, s'establien **middleware** i **rutes.** El **middleware** són funcions que ajuden a verificar les dades arribades del **Frontend** i poden ser utilitzades en diversos casos per estalviar codi.

En el cas d'aquest projecte, degut a la simplicitat, es va optar per utilitzar pocs **middlewares.** Les **rutes,** per un altre costat, serveixen per quan un usuari realitza una petició **HTTP** (com obrir una pàgina o enviar un formulari), la sol·licitud arriba a una web, i aquesta ha de saber com gestionar aquesta sol·licitud. Això es fa mitjançant **rutes** que es defineixen al servidor.

```
//Funció de verificació existencia de user gràcies a les cookies
const verifyUser = (req, res, next) => {

    const token = req.cookies.token;

    if (!token) {
        return res.json({ Error: "No autenticació" });
    } else {
        jwt.verify(token, "jwt-secret-key", (err, decoded) => {
            if (err) {
                return res.json({ Error: "Token incorrecte" });
            } else {
                req.name = decoded.name;
                req.niu = decoded.niu;
                req.role = decoded.role;
                next();
            }
        });
    }
};

//Funció per verificar al usuari, el seu rol i id
app.get('/', verifyUser, (req, res) => {
    return res.json({ Status: "Success", name: req.name, niu:req.niu, role:req.role});
});
```

Fig. 8: Exemple Middleware + Ruta

Dins les mateixes **rutes**, les consultes **SQL** son generades i executades. Un cop rebuda la resposta, segons la naturalesa de la consulta, les dades es retornen al **Frontend.**

8.4 Funcionament de l'aplicació

L'aplicació consta d'un total de 22 **arxius JSX** (amb 10 arxius d'estil CSS addicionals) per la part del **Frontend** i un sol arxiu **JS** a la part del **Backend** anomenat **server.js,** subdividits en dues carpetes, **Frontend/frontend i Backend** respectivament.

L'aplicació s'inicia amb el llançament del **component Homepage**, encarregat de mostrar a l'usuari les opcions de registrar-se o bé iniciar sessió. Els **components Register i Login** són els encarregats de dur a terme dites funcions. Destacar l'ús de **jwt tokens i cookies** per evitar el **Cross-Site Scripting,** facilitar el treball al servidor a l'hora de la validació i també diferenciar que pot realitzar cada usuari segons el seu rol.

```
if (err) {
    return res.json({ Error: "Error intern" });
}
if (response) {
    const role = data[0].role;
    const name = data[0].name;
    const niu = data[0].niu;
    const token = jwt.sign({ name, niu, role }, "jwt-secret-key", { expiresIn: '1d' });

    res.cookie('token', token, { httpOnly: true, secure: true, sameSite: 'None' });

    return res.json({ Status: "Success" });

} else {
    // Retornant error en cas de contrasenya incorrecta
    return res.json({ Error: "Contrasenya incorrecta" });
}
```

Fig. 9: Creació Token i Cookies

Seguidament, mitjançant el **token** i un **component auxiliar** anomenat **Modulespage**, es verifica que tots els credencials de l'usuari estan en ordre i són correctes per tal d'accedir a l'aplicació i visualitzar el que li correspon segons el valor del rol.

El **component Dashboard** és el **component** que es pot considerar l'inici de l'aplicació. És on es fa un mostreig de totes les assignatures on l'usuari està matriculat i pot accedir-hi.

Fig. 10: Component Dashboard

Només el professorat pot afegir o eliminar assignatures, l'alumnat només està restringit a accedir a elles si un professor els afegeix o bé, posseeix el id i la contrasenya de l'assignatura.

Un cop dins d'una assignatura, un menu amb 4 opcions apareix a la part superior. Aquest són: **Curs (component ElementsCurs), Participants (component ElementsParticipants), Preguntes (component ElementsQuestions) i**

Tests (component ElementsTest). Destacar que tots ells tenen en comú la part superior de la pantalla, fins al mateix menu, ja que tant el **header** com el propi **menu** de l'aplicació són dos **components (component Headercap per la part superior verda i component AssignaturaLayout per la part gris i el menu).**

Al **component ElementsCurs** és on els diversos **temes** de l'assignatura es despleguen com una llista. Dins de cada tema es pot dividir en tres subapartats: **Continguts, Proves Pràctiques i Proves Avaluatives.** Els apartats de les proves recuperen cada cop que es renderitza el component aquells **tests** associats al **tema** i que tenen el tipus definit com **avaluatiu o practica.** El professorat disposa d'un botó amb el símbol **+ a cada apartat** per tal de generar noves proves.

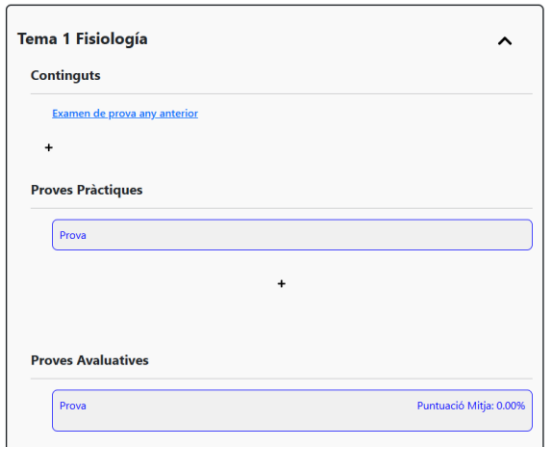


Fig. 11: Apartats Tema

Al seleccionar la icona **+**, es carrega un **component** anomenat **CreateQuizz,** encarregat de facilitar la creació del test al professorat. Aqui, el professor pot escollir els temes, el número de preguntes i la dificultat de les preguntes que vol incorporar. També es disposa d'un botó que permet la creació de forma **manual** del test carregant el **component CreateManualQuizz,** on el professor ha d'escollir a mà quines preguntes vol realitzar.

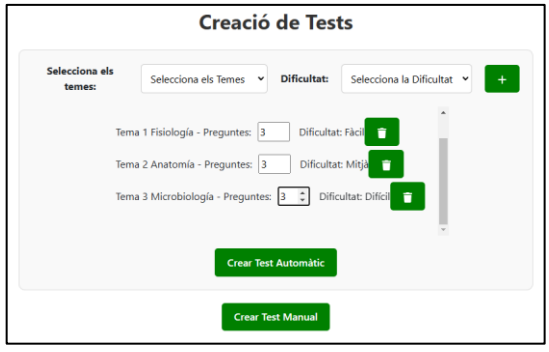


Fig. 12: Creació de Tests

Fig. 13: Apartats Participants

Posteriorment, el professor també pot personalitzar l'ordre de les preguntes del test i altres aspectes més concrets a través del **component CustomTest.**

A l'apartat de **Participants**, el **component ElementsParticipants** mostra tots els participants de l'assignatura, tant professorat com alumnat, i la informació bàsica de cadascun. En cas del professorat, pot visualitzar també les notes de l'alumne i un registre dels últims tests avaluatius que ha realitzat. També té la possibilitat d'eliminar o afegir participants mitjançant el id de cada alumne (**NIU per facilitar la semblança amb la realitat**) o bé mitjançant un fitxer **csv.**

La secció de **Preguntes** està composada per dos **components: ElementsQuestions i AddQuestion.** El primer és l'encarregat de mostrar totes aquelles preguntes que tenen un estat ja acceptat (poden aparèixer als tests) o avaluatiu (han de ser avaluats pel professorat abans de ser acceptades). També permet al professorat editar la pregunta en tots els seus paràmetres o eliminar-la del banc de preguntes global. L'alumne només podrà visualitzar aquelles preguntes que ell mateix ha afegit. El segon **component, AddQuestions,** és el que permet afegir preguntes tant a l'alumnat com al professorat perquè puguin ser avaluades.

En cas de l'alumne, només podrà realitzar fins a 3 preguntes simultànies.

Fig. 14: Apartat Afegir Pregunta

Finalment, a l'apartat de **Tests**, el **component Elements-Test és** l'encarregat de mostrar les opcions de les que disposa l'alumne per practicar pel seu compte. Té l'opció de generar tests segons els conceptes escollits o bé, pot generar un **tests reactiu** que, segons les respostes escollides en el moment per part de l'alumne, les probabilitats d'aparició d'un tema concret varien, generant preguntes variades que intenten definir les debilitats de l'alumne.

Fig. 15: Apartat Tests

El *layout* del test, ja sigui de caire **avaluatiu o pràctic,** segueix el mateix model. Com es pot veure a la *Il·lustració 17,* consta d'una preguna i fins a 4 opcions variables, amb una resposta correcta. Depenent de si forma part de l'eina de **pràctica** per l'alumnat o no, un temporitzador i eines de feedback apareixeran.

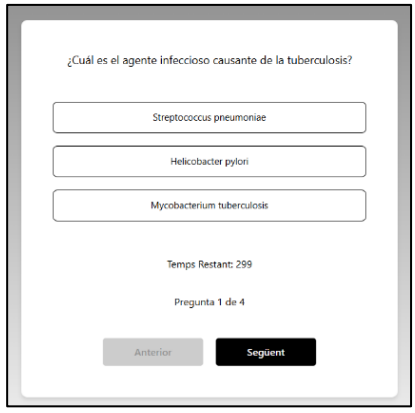


Fig. 16: Qüestionari

L'apartat del **Backend** simplement respon a les peticions de càrrega de certs **components** al ser renderitzats o bé, si l'usuari envia elements a la base de dades, com per exemple al afegir una pregunta o bé crear un test.

9 Testeig

Dins la fase de testeig es van dur a terme diverses proves de totes les funcionalitats implementades de forma que qualsevol possible error o anomalia del projecte fos tractada de forma consistent.

En primer lloc es van realitzar **proves de funcionalitat.** Aquestes proves bàsicament tenien com objectiu principal que els artefactes desenvolupats funcionessin i executessin de forma correcta i consistent els objectius marcats segons els requisits de l'anàlisi. Per exemple, que el **component Register** registrés les dades proporcionades a l'usuari. Seguidament, es van fer proves de la interfície d'usuari**,** sobretot alhora d'enviar formularis al **Backend** i mostrar les dades obtingudes a partir de les funcions de renderitzat de cada **component**. Aprofitant aquestes proves, es van realitzar **proves d'integració** amb l'objectiu de verificar que la navegació entre els **components** era correcta i les dades s'enregistraven correctament a la base de dades.

També abans d'iniciar la connexió amb la base de dades **MySQL** es van realitzar testeigs de les APIs mitjançant **Postman.** Finalment, es van dur a terme proves de interfície**.** Aquestes proves tenien com objectiu observar el comportament del programa davant la modificació de la pantalla de l'usuari.

10 Desplegament

Com ve s'ha descrit als apartats anteriors, l'aplicació consta de tres parts: la **Frontend**, basada **en React JS**, encarregada de mostrar la interficie a l'usuari, i la **Backend**, composada per un servidor basat en **Node JS** i una base de dades relacional basada en **MySQL**.

S'han escollit una serie de serveis de hosting al núvol per agilitzar el desplegament. Per la part del **Frontend**, s'ha emprat **Netlify**, un servei de hosting de pàgines web amb contingut dinàmic de forma ràpida i gratuïta, la qual es desplega de forma automàtica mitjançant els commits de Git i ofereix protocols SSL (Secure Sockets Layer) gratis per millorar la seguretat.

A la part del **Backend** s'ha emprat **Railway**. És una plataforma d'allotjament de Backend, base de dades… Permet igual que **Netlify** conectarse amb el projecte a **Git** i realitzar desplegaments segons l'últim commit. A més, permet connectar els diversos elements entre si, facilitant molt la gestió de variables d'entorn i altres elements necessaris per establir enllaços entre les diverses parts del projecte.

11 Assoliments

Durant el desenvolupament del projecte s'han implementat diverses funcionalitats que aporten gran valor a la plataforma.

En primer lloc, la gestió de comptes d'usuari inclou opcions com el registre i l'inici de sessió per a alumnes i professors, garantint rols diferenciats amb permisos específics per a cada tipus d'usuari. A més, cada perfil és personalitzable, permetent incloure informació detallada sobre l'usuari.

Pel que fa a la creació i gestió de tests, la plataforma permet la generació automàtica de preguntes basades en bancs de dades. El professorat pot dissenyar tests de pràctica personalitzats segons el tema, el nivell de dificultat i els conceptes, així com crear tests avaluatius protegits amb una clau d'accés. A més, els alumnes poden proposar preguntes, que són revisades i aprovades pel professorat abans

d'incloure-les al banc de preguntes. Aquest banc és totalment gestionable pel professorat, que pot afegir, editar o eliminar preguntes segons les necessitats. També es disposa de funcionalitats avançades de personalització dels tests avaluatius, com la modificació de l'ordre de les preguntes, la durada dels tests i la data de finalització.

En l'àmbit de la gestió de participants i assignatures, els professors poden gestionar els alumnes associats a cada assignatura, afegint o eliminant participants mitjançant arxius **CSV** o a través del **NIU**. Així mateix, poden ampliar el contingut temàtic afegint nous temes o categories dins d'una assignatura, enriquint el banc de preguntes i adaptant-lo als objectius del curs.

Durant l'execució dels tests, els alumnes poden accedir als tests avaluatius a través de codis proporcionats pel profes sor, gaudint d'una interfície intuïtiva. També compten amb una eina de pràctica que permet crear tests segons paràmetres definits (conceptes), així com un test intel·ligent que genera preguntes aleatòries i s'adapta a les respostes en temps real.

Finalment, el sistema de correcció i avaluació dels tests genera resultats complets, incloent la puntuació total, les respostes correctes i incorrectes, i la nota final, oferint així una experiència completa i útil per a l'entorn educatiu.

12 Conclusió

La web generadora d'exàmens ha aconseguit complir amb l'objectiu principal d'oferir una eina fàcil d'utilitzar, dissenyada per facilitar la pràctica d'exàmens i fomentar la col·laboració entre el professorat i l'alumnat a l'hora de generar continguts educatius.

Durant la fase de desenvolupament, la definició de requisits generals i l'aparició de nous requisits van dificultar el progrés del projecte, la qual cosa va provocar un endarreriment en la planificació global. Aquestes complicacions van ocasionar que alguns requisits no es poguessin implementar o que no es fes d'una manera òptima. Malgrat això, el producte final ofereix funcionalitats que el converteixen en una eina útil i adaptada a les necessitats del context educatiu actual.

A més, personalment, aquest projecte m'ha permès establir una base sòlida en noves tecnologies com **React JS** i **Node JS**, a la vegada que he pogut desenvolupar i consolidar la meva experiència prèvia en la planificació, disseny i proves de programari.

13 Agraïments

En primer lloc, vull expressar el meu agraïment al meu tutor, Javier Sánchez Pujades, per la seva proposta incial, així com les valuoses idees i consells que m'ha proporcionat al llarg de tot el desenvolupament del projecte.

També vull donar les gràcies als meus companys de Catalana Occident pel feedback constructiu i les crítiques, que van ser crucials durant la fase de desenvolupament.

I finalment, agraeixo sincerament a la meva família i amics pel seu suport incondicional, la motivació i calma durant moments d'estrés.

Bibliografia

[1] *Acerca de Moodle - MoodleDocs*. (n.d.). Moodle Docs. Retrieved October 5, 2024, from https://docs.moodle.org/all/es/Acerca_de_Moodle#Contru%C3%ADdo_para_el_aprendizaje,_globalmente

[2] Denny P. (2008). *PeerWise - Documentation*. PeerWise. Retrieved October 5, 2024, from https://peerwise.cs.auckland.ac.nz/docs/

[3] Dr Chris Adams. (n.d.). *Case study: Student-authored quizzes for revision with Peerwise*. University of Bristol. Retrieved September 30, 2024, from https://www.bristol.ac.uk/digital-education/case-studies/pre-2018/student-authored-quizzes-revision-with-peerwise/

[4] Ramírez, I. (2024, July 18). *Kahoot!: qué es, para qué sirve y cómo funciona*. Xataka. Retrieved October 5, 2024, from https://www.xataka.com/basics/kahoot-que-es-para-que-sirve-y-como-funciona

[5] Aguirre, M. F., & Gil, E. (2024, July 8). *Agile y Scrum: ¿qué son y cómo elegir la mejor metodología?* Appvizer. Retrieved September 23, 2024, from https://www.appvizer.es/revista/organizacion-planificacion/gestion-proyectos/agile-vs-scrum

[6] *Jira | Issue & Project Tracking Software*. (n.d.). Atlassian. Retrieved October 5, 2024, from https://www.atlassian.com/software/jira

[7] Perforce. (2023, May 5). *How to Draw a UML Use Case Diagram (With Examples)*. Gliffy. Retrieved November 2, 2024, fro https://www.gliffy.com/blog/use-case-diagrams

[8] Git. (n.d.). Git SCM. Retrieved October 5, 2024, from https://git-scm.com

[9] Microsoft. (n.d.). Visual Studio Code - Code Editing. Redefined. Retrieved October 5, 2024, from https://code.visualstudio.com

[10] *MySQL Workbench*. (n.d.). MySQL. Retrieved October 5, 2024, from https://www.mysql.com/products/workbench

[11] Node.js — Run JavaScript Everywhere. Retrieved October 5, 2024, from https://nodejs.org

[12] Express - Node.js web application framework. Retrieved October 5, 2024, from https://expressjs.com

[13] Goode, T. (2018, November 4). *cors - npm*. NPM. Retrieved November 9, 2024, from https://www.npmjs.com/package/cors

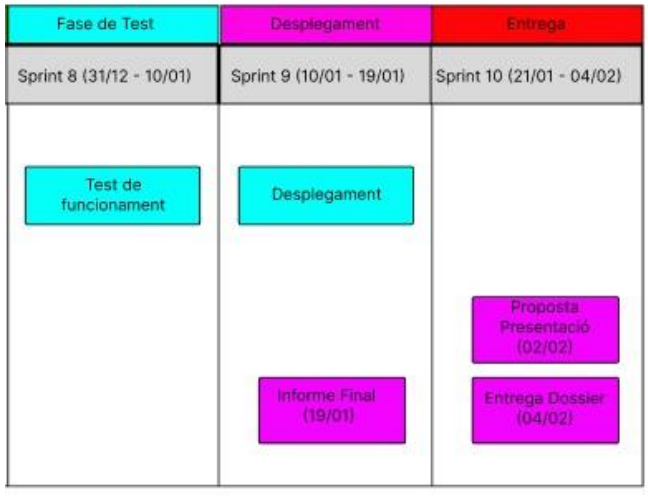
[14] Deyimar A. (2023, June 29). *Qué es React: definición, características y funcionamiento*. Hostinger. Retrieved September 23, 2024, from https://www.hostinger.es/tutoriales/que-es-react

[15] React. Retrieved October 5, 2024, from https://react.dev/

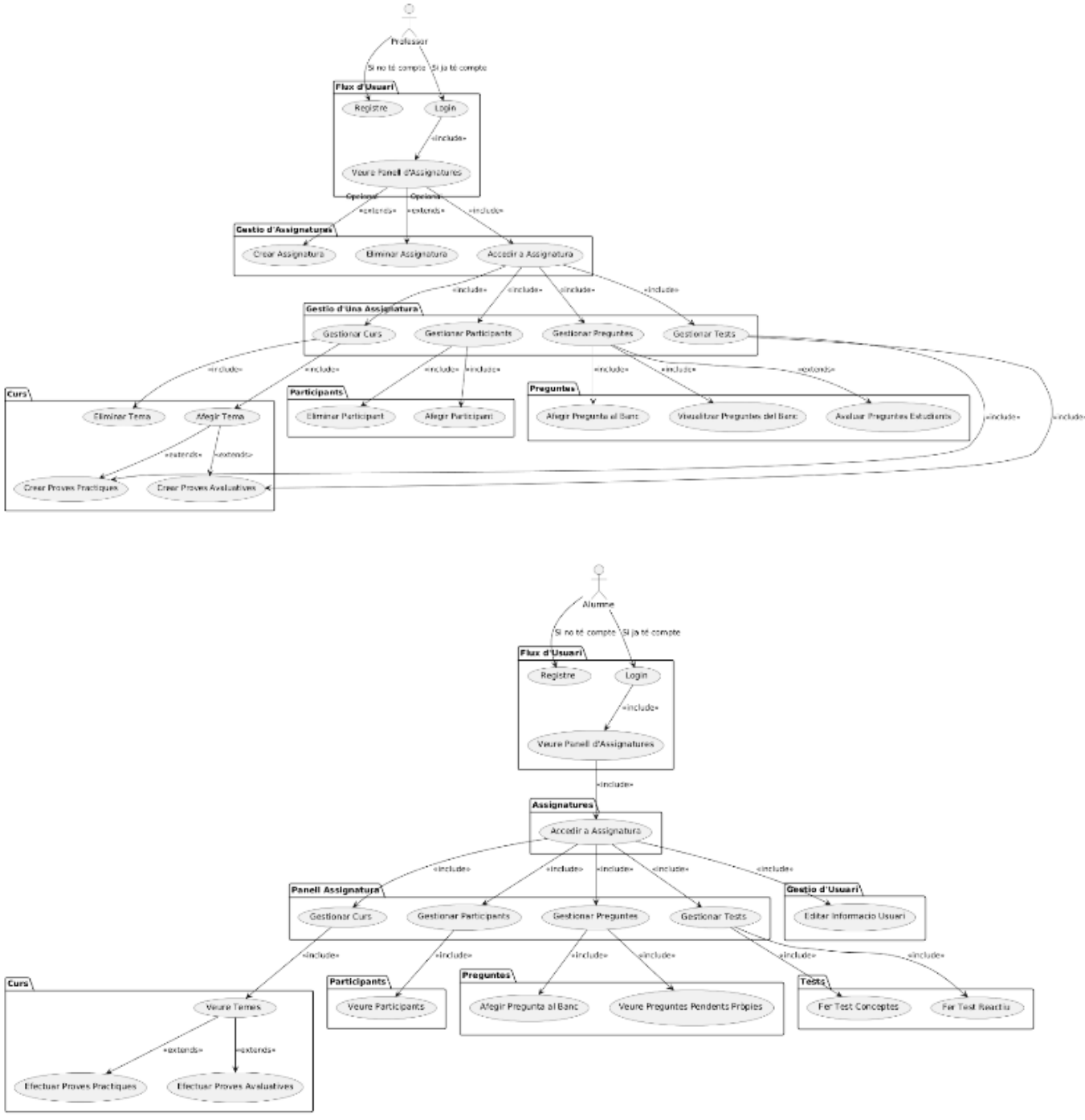
[16] *MERN Stack: Qué es y qué ventajas ofrece*. (2020, October 21). OpenWebinars. Retrieved October 7, 2024, from https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/

APÈNDIX

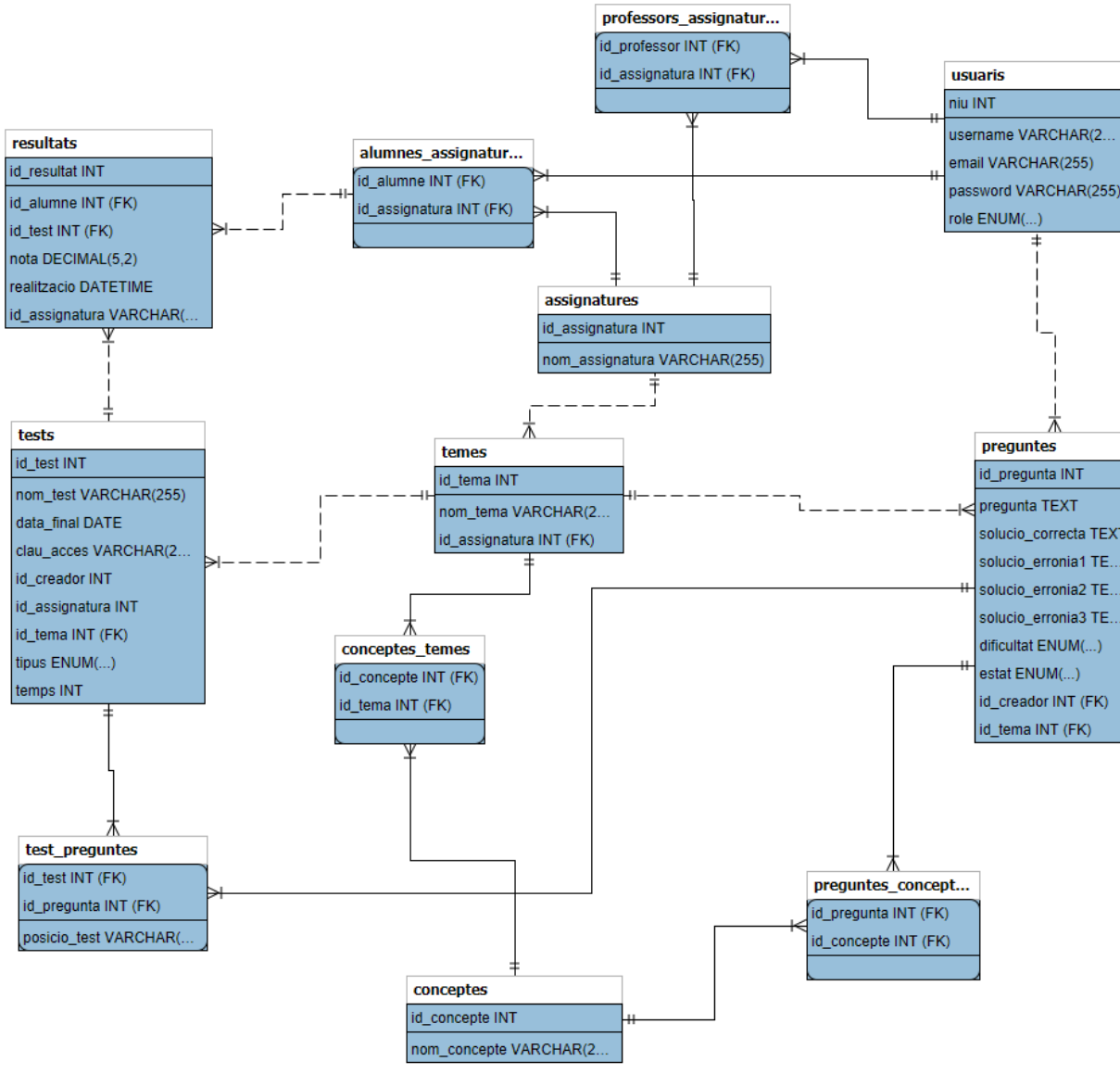
A1. DIAGRAMA DE GANTT



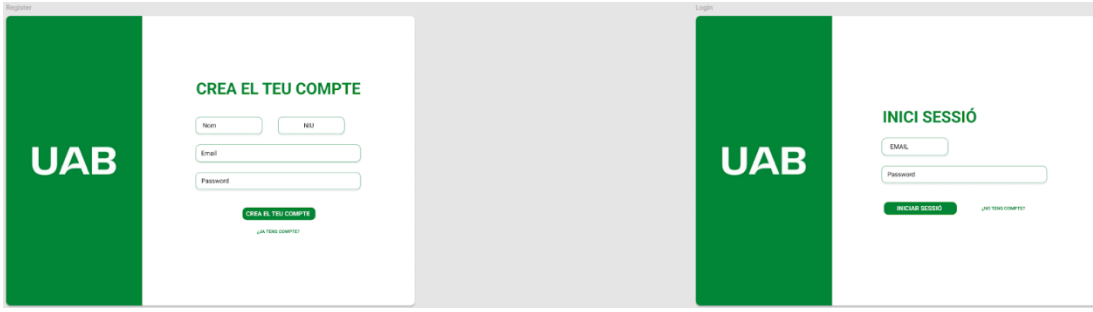
A2. DIAGRAMA CASOS D'ÚS

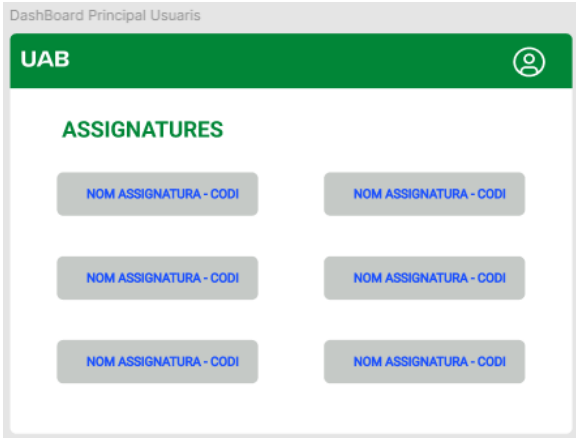


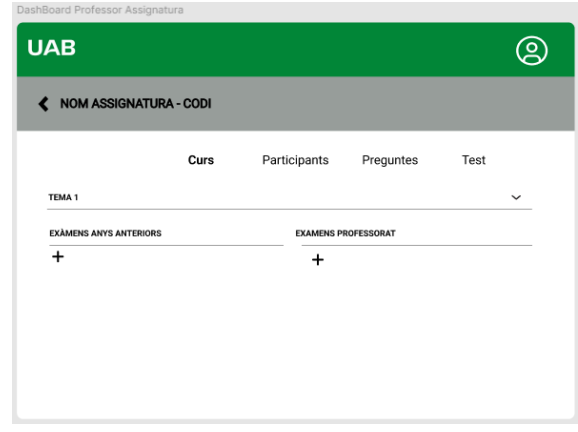
A3. DIAGRAMA E-R

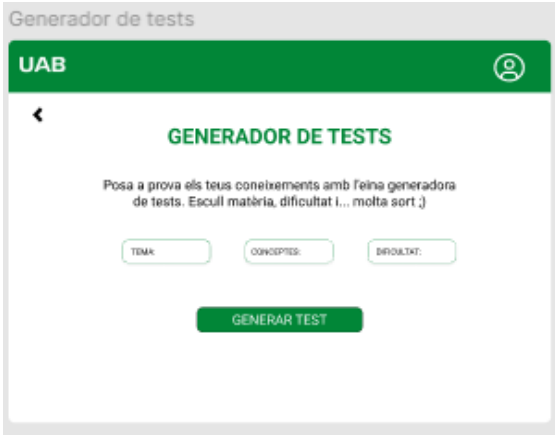


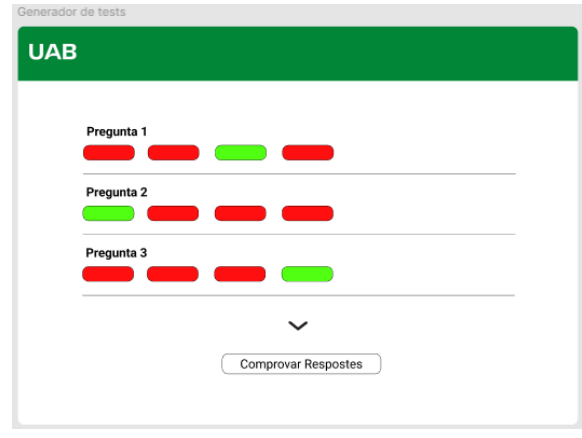
A4. DISSENY UI

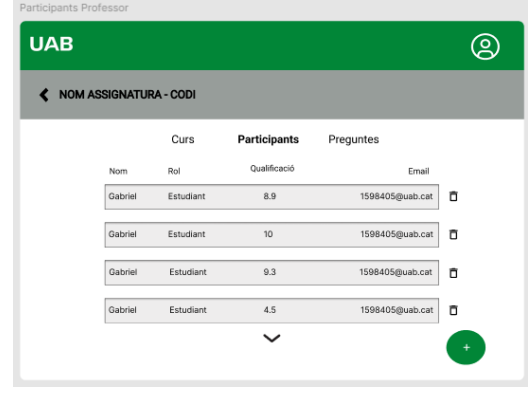












A1. DIAGRAMA DE COMPONENTS