Back-office Servidor de Correcció automàtica pel professorat

Eric Llorente Garcia

Resum — Els estudis relacionats amb la programació requereixen que els seus estudiants puguin aprendre i plasmar les seves habilitats fent una gran quantitat d'exercicis pràctics de programació. Per tal que els alumnes tinguin una forma còmoda i automàtica de corregir els seus propis exercicis sense l'assistència del professorat, s'utilitzen eines de correcció automàtica. En aquest projecte, es busca millorar la interfície d'una eina de correcció automàtica ja existent, que s'usa en algunes assignatures de la menció de computació del grau d'enginyeria informàtica de la Universitat Auònoma de Barcelona. Més concretament, cal millorar la banda del servidor, la qual és feta servir pel professorat. L'objectiu és realitzar una pàgina web on es tinguin les mateixes funcionalitats que ja es troben en el corrector original, però amb una interfície més intuïtiva i amigable pel professorat.

Paraules clau—Corrector, autocorrector, aplicació web, Java, Spring Boot, exercicis de programació

Abstract — Studies related to coding require that their students can learn and apply their skills by doing a large number of coding exercises. In order for students to have a comfortable and automatic way to correct their own exercises without the assistance of teachers, automatic correction tools are commonly used. This project's aim is to improve the interface of and existing self-correction tool, which is used in some subjects of the computer science degree of the Universitat Autònoma de Barcelona. More specifically, it is necessary to improve the server-side, which is used by the teaching staff. The goal is to create a web page that has the same functionalities as those found in the original corrector, but with a more intuitive and user-friendly interface.

Index Terms — Corrector, self-corrector, web app, Java, Spring Boot, coding exercises

1 Introducció - Context del treball

En les carreres relacionades amb la programació és habitual que els alumnes realitzin una gran quantitat d'exercicis pràctics. Per aconseguir que aquests exercicis de programació puguin resultar de màxima utilitat als alumnes, és necessari que tinguin una forma de poder comprovar que s'han dut a terme correctament. Per aquesta raó, vàries universitats utilitzen eines de correcció automàtica que permeten als alumnes saber la nota del seu exercici i, en alguns casos, els errors presents.

En aquest projecte, es treballarà a partir d'una eina ja desenvolupada que s'utilitza actualment per diverses assignatures de la menció de computació. Aquest corrector permet als alumnes fer correccions en qualsevol moment, així com lliuraments. Les correccions s'encarreguen de donar *feedback* a l'alumne, mentre que el lliurament és l'entrega final de l'alumne i la nota del qual serà assignada a aquest. Si l'alumne entrega més d'un lliurament, la nota vàlida serà la de l'últim.

L'autocorrector també té una banda pels professors, on poden veure les diferents correccions i lliuraments fetes pels alumnes, les seves notes i els arxius de les pràctiques dels alumnes. També poden gestionar els correctors i lliuraments, creant-ne de nous o eliminant els ja existents. Per últim, tenen l'opció de consultar estadístiques com la quantitat d'hores aproximada dedicades pels alumnes a un corrector, de manera que poden saber si una pràctica està resultant complicada, per així actuar en conseqüència fent els aclariments necessaris.

- *E-mail de contacte: 1566166@uab.cat*

- *Menció realitzada: Enginyeria de Software*
- *Treball tutoritzat per: Javier Sánchez Pujadas, Departament de Computació*
- *Curs 2023/24*

L'objectiu d'aquest treball és millorar la interfície d'aquesta eina en la banda del professorat, creant una pàgina web des d'on es puguin fer les mateixes funcionalitats. La motivació d'aquest treball sorgeix perquè, actualment, aquesta eina s'utilitza per mitjà de comandes en un *cmd* de Windows. Aquesta implementació no és especialment intuïtiva i requereix saber com funcionen les diferents comandes existents. Per això es vol aconseguir tenir una eina més simple pel professorat.

2 Objectius

Per tal de realitzar un treball que comprengui totes les funcionalitats que es demanen, s'ha desglossat en diferents objectius:

- **Objectiu 1: Desenvolupar les funcionalitats principals de la pàgina web**. En aquest primer objectiu, es vol aconseguir tenir les funcionalitats principals de l'eina d'autocorrecció a la web. Aquestes funcionalitats suposen poder consultar les notes dels alumnes, tant dels correctors com dels lliuraments, poder donar d'alta o baixa els professors i alumnes i, finalment, crear o eliminar correctors i lliuraments. Per altra banda, els professors han de poder iniciar sessió amb les seves credencials, primerament assignades per un professor administrador.

- **Objectiu 2**: **Incorporar la pàgina web en un servidor de la UAB.** Per tal de proporcionar accessibilitat a la web als professors, el projecte es trobarà en una màquina proporcionada per la UAB des d'on servirà la web als usuaris.

- **Objectiu 3: Poder descarregar la pràctica de l'alumne.** Els professors han de poder descarregar els arxius de la pràctica de l'alumne, tant els arxius de les correccions com dels lliuraments. Això permet als professors consultar directament la feina feta per l'alumne en aquells casos que sigui necessària una revisió manual.

- **Objectiu 4: Informes dels correctors i lliuraments.** Quan un alumne entrega una correcció o lliurament, aquesta entrega genera uns informes que proporcionen més detalls a l'alumne sobre les comprovacions que fa el servidor. L'alumne pot veure aquelles parts que el seu codi ha fet bé i les que no, obtenint també la nota total. En aquest objectiu es vol permetre als professors descarregar aquests informes perquè els puguin consultar si s'escau.

- **Objectiu 5: Generar diverses estadístiques de l'alumnat, correctors i lliuraments.** En l'eina de correcció automàtica original es poden consultar certes estadístiques per mitjà de comandes. Aquestes estadístiques permeten, entre altres, veure resums de correccions d'un alumne concret, veure quantes correccions s'han demanat d'un corrector, quantes hores s'han dedicat a cada corrector i la càrrega de treball dels alumnes.

Aquests objectius s'han reestructurat respecte els de l'informe inicial, amb la finalitat d'agrupar-los millor i fer-los més fàcils d'entendre.

3 Estat de l'art

Tot i que les eines de correcció automàtica d'exercicis de programació són habituals en graus relacionats amb la informàtica, en molts casos són desenvolupades pel mateix professorat per l'àmbit docent, motiu pel qual no s'acostuma a pujar documentació o informació de l'eina a Internet.

Aquí mateix, en el grau d'enginyeria informàtica de la UAB, varies assignatures utilitzen altres eines de correcció automàtica. Com a estudiant, les he utilitzat en moltes ocasions. Aquestes eines acostumen a trobar-se allotjades a Caronte. Permeten programar en la pròpia eina, tot i que per comoditat és habitual fer-ho en un entorn de programació i posteriorment pujar els arxius o el codi a l'eina. No he trobat documentació al respecte, però per la meva experiència, el que fan aquestes eines és comparar de forma literal la sortida generada per l'estudiant amb la sortida esperada pel professorat. Els dos resultats han de ser estrictament iguals per tal de passar el test.

Un tipus d'eina molt similar són els *Online Judge*[1], sistemes que corregeixen programes, principalment enfocats a fer concursos online de programació. Aquests sistemes compilen i executen el codi enviat i també comparen el resultat del codi amb el resultat correcte. En addició, es poden afegir diverses restriccions, com de temps, memòria i seguretat.

Algunes universitats han adaptat aquests programes per utilitzar-los en activitats docents. En són exemples la Universitat Politècnica de Catalunya (UPC) amb Jutge.org[2] i la Universitat de Valladolid (UVa) amb OnlineJudge[3].

Aquests programes suporten una gran quantitat de llenguatges de programació i s'empren en diferents assignatures. En el cas de Jutge.org, els professors poden accedir a estadístiques de l'estudiant. Aquestes estadístiques poden ser globals o d'un estudiant particular.

Com a conclusió, existeixen eines que ofereixen opcions molt similars a les que presenta aquest projecte, tot i que aquests correctors també estiguin pensats per altres activitats com els concursos de programació.

3 Metodologia

La metodologia escollida per aquest projecte és una metodologia Agile, basada en el desenvolupament iteratiu i incremental.

Les metodologies àgils consten de diferents *sprints*, que

tenen l'objectiu de dividir i modular l'abast del projecte. En cada sprint es defineixen les tasques que es realitzaran (product backlog[4]) amb l'objectiu de tenir un prototip funcional per poder mostrar al client.

El motiu de triar aquesta metodologia ve donat per la possibilitat de rebre *feedback* per part del client, que en aquest cas és el tutor, per saber si el projecte va ben encaminat i fer les modificacions necessàries progressivament. D'aquesta manera, si una funcionalitat no és com el client esperava o s'ha produït un malentès, es soluciona abans de continuar amb la resta de les tasques, el que pot suposar en molts casos arrossegar l'error en futures implementacions.

Per dur a terme aquesta metodologia, s'ha dividit l'abast total del desenvolupament del projecte en 7 fases. Generalment, cada fase era suficient gran per a poder encabir més d'un sprint, tot i que no ha sigut així en tots els casos. L'objectiu era realitzar sprints setmanals que comprenguessin una funcionalitat de la fase pertinent que posteriorment es pogués mostrar al client. Malgrat això, no sempre ha sigut possible dur a terme tot el desenvolupament dels sprints en només una setmana.

3.1 Fases del desenvolupament

A continuació es llisten i expliquen les fases del desenvolupament:

- **Fase 1. Estructura inicial de la pàgina amb vistes navegables.** Aquesta primera fase consisteix a dissenyar i crear l'estructura bàsica que tindrà la web. Un cop acabada, ha de ser possible navegar per les diferents vistes, tot i que les funcionalitats encara no són implementades. L'objectiu principal és crear la primera versió dels controladors i les vistes.

- **Fase 2. Creació dels usuaris i assignatures.** Hi ha dos tipus d'usuari, els professors i els alumnes. Tots dos tenen els mateixos atributs, però només els professors podran iniciar sessió amb les seves credencials. En aquesta fase es vol dissenyar el model de l'usuari, el qual pot ser un professor o un alumne. Per altra banda, es vol implementar la creació d'assignatures, on es poden indicar els alumnes i professors que hi participaran.

- **Fase 3. Inici de sessió i donar d'alta o baixa els usuaris.** En aquesta fase es vol ser capaç d'iniciar sessió com a professor, així com poder afegir o eliminar alumnes i professors. La informació dels usuaris es guarda en arxius .csv[5] a petició del tutor. Per tant, per aquesta fase és necessari llegir i escriure en arxius .csv que es troben dins del projecte.

- **Fase 4. Afegir i eliminar correctors i lliuraments.** Per poder afegir correctors i lliuraments, aquests han de ser inclosos en el fitxer .ini[6] d'una assignatura determinada. D'aquesta forma, el servidor de l'eina de correcció serà capaç de llegir els nous correctors i els seus atributs, com quins arxius ha d'entregar l'alumne o quins correctors ha de passar l'entrega. Per aquesta fase és necessari poder llegir i escriure informació en els fitxers .ini.

- **Fase 5. Mostrar notes dels alumnes.** El propòsit d'aquesta fase és poder consultar les notes dels alumnes, tant d'una correcció com d'un lliurament. Quan un alumne fa una entrega qualsevol, es genera un arxiu .csv. En el cas de les correccions, es genera aquest arxiu per cada una de les entregues que faci l'alumne, i totes es poden consultar. Si és un lliurament, sempre es sobreescriu l'última nota, per tant, només es pot accedir a la nota de l'últim lliurament.

- **Fase 6. Descarregar les correccions i lliuraments.** De forma similar a la fase anterior, quan un alumne fa una entrega també es genera una carpeta amb els seus arxius principals. En aquesta fase es vol poder descarregar l'últim lliurament o correcció d'un alumne.

- **Fase 7. Informes de les entregues i estadístiques.** L'objectiu és poder veure els informes d'una correcció o lliurament i a més, poder executar les comandes del corrector per obtenir estadístiques dels estudiants i les entregues. Els informes de les entregues també es generen automàticament. En aquesta fase es vol poder obrir en el mateix navegador els informes dels lliuraments i correccions.

- **Fase 8. Documentació i possibles correccions.** Finalment, en l'última fase, es vol dur a terme tota la documentació del projecte, així com les correccions o addicions de funcionalitats necessàries.

4 Eines utilitzades

Durant el desenvolupament del projecte s'han utilitzat diverses eines i llenguatges. El criteri a l'hora d'escollir-les va ser prioritzar aquelles eines que ja fossin familiars i que s'adaptessin als requisits del projecte.

4.1 IDE

Com a IDE s'ha utilitzat IntelliJ[7] de JetBrains. IntelliJ és un IDE que té suport per diversos llenguatges, incloent-hi tots els necessaris per a aquest projecte. Els motius principals per triar-lo són que és un programa molt còmode d'utilitzar amb tota mena d'eines i que compto amb experiència prèvia utilitzant-lo. També, cal destacar que té un assistent de codi molt avançat que dona suggeriments útils així com possibles solucions pels errors.

4.2 Gestió

Per gestionar el projecte he fet servir l'eina de desenvolupament software Jira[8]. Encara que hi ha altres eines molt similars, aquest cop també m'he decantat per Jira per ser l'eina amb la que tinc més experiència. A Jira, he anat creant les diferents fases del projecte i el backlog. Un cop s'han decidit les tasques a fer (backlog), Jira presenta un

taulell similar a qualsevol taulell Kanban[9], on es poden anar col·locant les tasques. Aquest taulell té principalment 3 columnes:

- **Tasques per fer:** es col·loquen les tasques que encara no s'han començat.
- **En curs:** s'hi troben les tasques que estan en desenvolupament.
- **Llest:** on hi ha les tasques finalitzades.

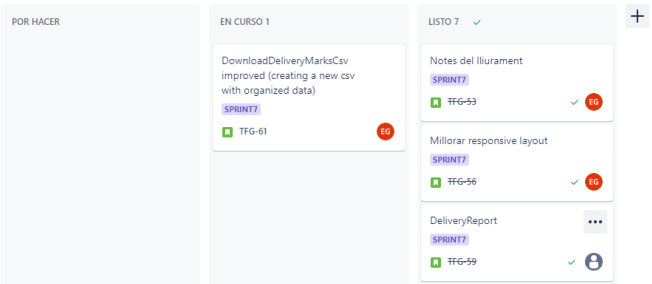


Fig.1 Taulell de Jira

A l'apèndix es pot consultar el diagrama de Gantt del projecte, el qual també s'ha anat gestionant amb Jira.

4.3 Backend

En el backend s'ha utilitzat Java[10] amb Spring Boot[11]. Spring Boot és una eina que permet treballar amb el framework de Java anomenat Spring[12] d'una forma molt còmode. Concretament, és el framework Spring el que dona suport a Java per desenvolupar pàgines web. La decisió per decantar-se per un framework de Java va ser presa per la recomanació del tutor

4.4 Frontend

Els llenguatges de programació fets servir pel frontend han sigut HTML[13] i CSS[14]. Addicionalment, s'ha utilitzat Thymeleaf[15], un motor de plantilles a la banda del servidor que permet enviar informació des del servidor al client d'una forma molt simple. A més, Thymeleaf és un motor que funciona amb Java, el que ha fet que fos especialment adient.

4.5 Control de versions

El control de versions s'ha portat amb Github[16]. Github és un servei basat en *cloud* que permet que els desenvolupadors puguin allotjar-hi el seu codi i col·laborar entre ells. En aquest cas, Github s'ha utilitzat principalment per mantenir el control de versions, creant una nova branca per cada una de les fases i pujant el codi amb freqüència. L'avantatge de treballar d'aquesta forma és que, si per un casual, les edicions de codi o noves implementacions fan que el programa deixi de funcionar o generen comportaments no esperats, sempre es pot tornar a una versió anterior o consultar el seu contingut.

5 Requisits

Aquesta secció recull els requisits a implementar durant el projecte. Aquests requisits tenen com a finalitat poder complir tots els objectius proposats. Estan agrupats en funció de si són funcionals, no funcionals i tècnics.

5.1 Requisits funcionals

Número	Requisit
1	Donar d'alta i baixa els alumnes
2	Donar d'alta i baixa els professors
3	Afegir i eliminar correctors
4	Afegir i eliminar lliuraments
5	Afegir i eliminar assignatures
6	Inici de sessió del professorat
7	Llistar els alumnes
8	Llistar els professors
9	Llistar les assignatures
10	Llistar els correctors
11	Llistar els lliuraments
12	Donar d'alta alumnes a partir d'un CSV
13	Canviar la contrasenya dels alumnes
14	Llistar els alumnes que han fet una correcció determinada
15	Llistar els alumnes que han fet un lliurament determinat
16	Consultar les notes de totes les correccions d'un alumne
17	Descarregar el CSV de notes de l'última correcció d'un alumne
18	Descarregar els arxius de l'entrega de l'última correcció d'un alumne.
19	Obrir l'informe de l'útlima correcció d'un alumne
20	Descarregar el CSV de notes d'un lliurament amb un format més comprensible
21	Descarregar els arxius de l'entrega del lliurament d'un alumne
22	Obrir l'informe del lliurament d'un alumne
23	Tractar individualment els casos en que una entrega és realitzada per dos alumnes
24	Executar diferents estadístiques relacionades amb els correctors, lliuraments i alumnes

Taula 1. Requisits funcionals

5.2 Requisits no funcionals

Número	Requisit
1	La interfície ha de ser clara i intuïtiva
2	La navegació i interacció amb la web ha de ser ràpida, sense temps d'espera
3	La web ha de ser compatible amb diferents navegadors
4	La pàgina s'ha d'allotjar en un servidor de la UAB
5	La llista d'alumnes també ha d'incloure el

	professorat, per tal que el servidor els pugui detectar

Taula 2. Requisits no funcionals

5.3 Requisits tècnics

Número	Requisit
1	El llenguatge de programació pel backend és Java
2	S'utilitza el framework d'Spring
3	Els llenguatges del frontend han de ser HTML i CSS
4	S'utilitza Thymeleaf per passar la informació del backend al frontend
5	La informació dels usuaris s'ha de mantenir en CSV's
6	L'idioma de la web serà el català

Taula 3. Requisits tècnics

6 DESENVOLUPAMENT

6.1 Eina de correcció automàtica actual

Per entendre bé el desenvolupament d'aquest projecte, és necessari conèixer com és i quina estructura té l'eina que s'utilitza actualment. L'eina consta de tres carpetes principals, a continuació es detalla la seva informació més rellevant de cara a aquest desenvolupament:

- **Client.** Conté l'arxiu CorrectionClient.ini on s'especifica la connexió al servidor, a quina assignatura es fa i per part de quin estudiant.

- **Correctors.** Conté el codi i els executables que permeten fer ús de les comandes.

- **Server.** Inclou el *Server.BAT*, el qual s'encarrega d'estar en constant execució escoltant en el port definit per rebre les peticions dels alumnes. Dins d'aquest directori, s'hi troba el de *Courses*, on hi ha les diferents assignatures amb la seva corresponent informació, com els seus estudiants, professors, correctors i lliuraments.

  Tota la informació de cada assignatura excepte els estudiants es troba dins del fitxer .ini. A l'interior d'aquest fitxer es defineixen els correctors i els lliuraments, a més dels professors. Per definir un corrector, cal indicar quins són els arxius d'entrada per part de l'estudiant, els arxius de sortida que rebrà, els fitxers que es guardaran en el registre de correccions, la comanda que activarà el corrector, els arxius per l'informe i el temps màxim de correcció. Respecte als lliuraments, s'ha d'indicar la data d'inici i finalització, el directori on es troben els lliuraments, els fitxers que s'han de guardar de l'alumne i les correccions que ha de superar.

Per tal de tenir accés al corrector actual, aquestes carpetes es troben dins del projecte, englobades dins de *CorrectorTFG*. A través de la web, ara és possible afegir, eliminar o actualitzar els directoris i fitxers que s'hi troben a dins.

Durant el desenvolupament s'ha intentat respectar l'estructura original sempre que ha sigut possible.

6.2 Estructura de fitxers

Com ja s'ha mencionat, el desenvolupament del projecte s'ha realitzat en l'entorn d'IntelliJ, utilitzant principalment Java. Per poder treballar amb Spring Boot és recomanable tenir una estructura de carpetes i fitxers específica. Aquesta estructura consta de:

- **Controladors:** tenen els *endpoints* de l'aplicació. S'encarreguen de rebre la crida del frontend, executar la lògica necessària, sovint cridant els mètodes dels serveis, i retornar una sèrie de valors.

- **Models:** contenen l'estructura dels objectes dels usuaris, correctors, lliuraments i assignatures.

- **Serveis:** en Spring, els serveis contenen els diferents mètodes relacionats amb el model. Es pot arribar a cridar una mateixa instància d'un servei des de qualsevol punt del programa gràcies a l'anotació *Autowired* de Spring. Això permet que si, per exemple, un model s'ha omplert amb certes dades, es pugui utilitzar el servei associat i accedir a aquestes dades des de tot arreu.

- **Vistes:** les vistes en Spring Boot sempre han d'anar dins de la carpeta *resources*, d'altra forma no seran detectades. És habitual fer servir thymeleaf dins de les vistes, ja que és una de les formes més còmodes que proporciona el *framework* de Spring per rebre informació dinàmica.

6.3 Explicació dels desenvolupaments principals

En aquest apartat s'explicarà, sense aprofundir en detalls tècnics, com s'han dut a terme les funcionalitats principals de la web. Posteriorment, a l'apèndix, es podran veure els resultats.

- **Inici de sessió:** amb l'objectiu de realitzar l'inici de sessió de la web i donat que els arxius .csv que contenen la informació dels professors es troben dins de cada assignatura, s'ha decidit tenir un nou arxiu a la carpeta de *Courses* anomenat *LoginUsers*, el qual conté tots els professors de les diverses assignatures, tot i que si un mateix professor es troba a vàries assignatures sortirà només una vegada.

  Cada vegada que el programa s'executa, carrega els usuaris d'aquest arxiu i els guarda en una llista d'usuaris. Quan s'afegeix un professor a qualsevol assignatura, també es comprova si es pot afegir a *LoginUsers* (és a dir, s'afegirà si encara no hi és). Si s'elimina un

professor, altre cop es fa una comprovació, aquest cop es tracta d'assegurar-se si aquest professor ja no forma part de cap assignatura, cas en el que s'esborrarà el professor de *LoginUsers*.

L'arxiu LoginUsers té un usuari anomenat admin que no és cap dels professors. Aquest usuari permet al professor responsable afegir i eliminar professors mantenint sempre accés a la web.

- **Llistar assignatures:** per mostrar la vista d'assignatures, ja que totes les assignatures es troben a *Courses*, el programa s'encarrega de llegir i mostrar els noms de les carpetes d'aquest directori.

  Un cop se selecciona una assignatura, el controlador pertinent assignarà aquest valor a *SubjectService*, la qual es tracta d'una classe de tipus servei. Com s'ha comentat anteriorment, a Spring aquest tipus de classes són accessibles des de qualsevol part del programa per mitjà de l'anotació *Autowired*.

- **Donar d'alta assignatura:** aquesta vista permet al professor afegir noves assignatures, indicant el nom, l'identificador, un arxiu .csv d'alumnes i un altre de professors. El programa llegirà els arxius d'alumnes i professors creant llistes amb els seus valors i tot seguit executarà un mètode que crearà el nou directori de l'assignatura. Dins d'aquest, s'afegiran els arxius d'AlumnesSIGMA.csv i professorsSIGMA.csv, que guardaran les llistes de professors i alumnes llegides prèviament. Addicionalment, s'afegirà l'arxiu .ini inicialitzat amb la informació bàsica de l'assignatura. Encara que s'hagi creat l'arxiu ProfessorsSIGMA, realment el servidor només detectarà els professors si es defineixen en l'arxiu .ini. Per tant, també cal escriure el contingut de ProfessorsSIGMA dins l'arxiu .ini amb un format concret. El motiu de tenir l'arxiu .csv de professors és simplement per facilitar altres mètodes.

- **Llistar professors i alumnes:** aquestes dues vistes mostren els professors i alumnes de l'assignatura seleccionada, per mitjà de l'ús de *SubjectService*, que permet saber quina és l'assignatura. El programa s'encarregarà de llegir la informació dels arxius .csv corresponents i mostrar-la a la vista.

  Per com està dissenyat, el *server.BAT* sempre comprova que els professors també formin part de l'arxiu .csv dels estudiants. Per aquest motiu, quan es mostra la vista d'alumnes, cal filtrar d'entre tots els usuaris de l'arxiu AlumnesSIGMA quins són únicament estudiants.

  A l'hora d'eliminar els usuaris, siguin professors o alumnes, es rep l'identificador de l'usuari que es vol eliminar i es crida el mètode *DeleteUsersFromCSV*. Aquest mètode guarda en una llista temporal tots els usuaris de l'arxiu .csv excepte per l'usuari que es vol eliminar. Tot seguit, reescriu l'arxiu amb la llista temporal d'usuaris. En el cas dels professors, com també es troben a l'arxiu d'alumnes, això es farà un cop per l'arxiu de professors i un altre pel d'alumnes.

- **Donar d'alta usuaris:** per poder afegir professors o alumnes, primer es comprova que la informació del nou usuari compleix el format requerit i seguidament que aquest usuari no es trobi ja repetit. Si es compleixen aquestes condicions, s'afegirà l'usuari al final de l'arxiu .csv.

  Quan es dona d'alta un professor, també s'afegirà a la llista d'estudiants i, en el cas que encara no sigui a la de *LoginUsers*, també s'hi afegirà.

  Si s'utilitza l'opció d'afegir els estudiants des d'un arxiu .csv, es faran les dues comprovacions esmentades abans per cada un dels estudiants. S'afegiran tots els estudiants que compleixin les comprovacions i es mostrarà una llista amb aquells que no, de manera que el professor pugui saber quins estudiants ha d'ajustar.

- **Llistar correctors:** els correctors d'una assignatura es troben definits en el seu arxiu .ini. El mètode *ReadCorrectorFromINI* llegeix la informació d'aquest arxiu, tot buscant el patró [CORRECTOR nom_del_corrector] i guarda tots els noms en una llista que més endavant es mostrarà a la vista.

  Si el professor selecciona un dels correctors, podrà veure la llista d'alumnes que ja han fet una correcció. Per saber quins són aquests alumnes es busca en el directori *CorrectorRegistry*, dins del directori de la correcció, quins són els noms de les carpetes que conté, donat que els noms d'aquestes carpetes fan referència als identificadors dels alumnes. És important tenir en compte que una correcció pot tenir dos autors, el que dona lloc que el nom de la carpeta corresponent contingui els identificadors dels dos estudiants. Aquests casos s'han de tractar de forma diferent, separant primer els dos identificadors i tractant-los de forma individual després. Si es dona el cas que un estudiant fa una correcció individual i també una de doble, aquest estudiant només ha de sortir un cop a la llista.

  Quan s'elimina un corrector, novament cal accedir a l'arxiu .ini de l'assignatura i esborrar tota la informació relacionada. Per fer-ho, es crea un arxiu .ini temporal on s'hi copia tot el contingut, ometent el corrector que es vol eliminar. En acabat, es torna a escriure el contingut a l'arxiu original.

- **Llistar lliuraments:** aquesta funcionalitat és molt similar a la de llistar correctors, doncs els lliuraments també es troben definits a l'arxiu .ini de l'assignatura. Altra vegada cal llegir aquest arxiu, però aquest cop

buscant el patró [DELIVERY nom_del_lliurament].

Anàlogament, si se selecciona un lliurament es mostraran els estudiants que ja l'han entregat, però aquest cop el directori on es generen les carpetes amb els identificadors dels estudiants és *Lliuraments,* dins el directori del lliurament. Aquesta vegada, també és possible descarregar un arxiu .csv de notes del lliurament, que té les notes de tots els estudiants. L'arxiu s'actualitza cada vegada que un estudiant fa una entrega d'aquest lliurament. Com l'arxiu generat de forma automàtica és una mica complicat de llegir, es genera un nou arxiu amb la informació més organitzada i detallada, que és el que finalment descarrega el professor.

Per eliminar un lliurament es segueix la mateixa lògica que en els correctors.

- **Donar d'alta correctors i lliuraments:** el funcionament per afegir correctors i lliuraments és idèntic, només canvia la informació introduïda pel professor. En tots dos casos, cal definir la nova entrega dins del fitxer .ini, seguint un format concret per tal de ser detectat pel *Server.BAT*. En aquelles entrades del formulari on el professor ha d'indicar més d'un element, per exemple, els fitxers de sortida, s'hauran de separar per una coma per poder ser correctament detectats i així mantenir el format correcte.

- **Correcció d'un estudiant:** si es selecciona un dels estudiants que ha realitzat la correcció, entrem a una vista on tenim accés a més informació.

  Primerament, trobem la llista de correccions que ha fet l'estudiant, indicades en funció del dia i hora que les va realitzar. Es pot seleccionar qualsevol d'aquestes correccions per veure la seva nota pertinent. Tan si les correccions són únicament fetes per l'estudiant o tenen dos autors, s'han de mostrar.

  Com ja s'ha explicat, en el directori *CorrectorRegistry* hi ha diversos subdirectoris amb els identificadors dels estudiants. Dins d'aquests subdirectoris, cada vegada que l'estudiant corresponent faci una correcció, es genera una nova carpeta que te com a nom el dia i hora de la correcció. Aquesta carpeta conté els arxius principals de la correcció, un arxiu .csv amb la nota i els resultats, i un informe en format .html.

  Per tant, per la funcionalitat de mostrar les diferents correccions a la vista que comentava, cal accedir al directori de l'estudiant dins de *CorrectorRegistry* i llegir els noms de les carpetes que conté, doncs cada carpeta és una nova correcció. És important tenir en compte que un estudiant pot tenir més d'un directori associat, depenent de si ha fet correccions individuals i dobles.

  En aquesta vista, també es pot descarregar l'arxiu .csv de l'última correcció, el qual, com es comentava abans, es genera de forma automàtica per part del servidor en el moment de la correcció. Per tenir en compte el cas que l'estudiant pugui haver realitzat correccions individuals i dobles, es comprova quina és la última carpeta associada a l'estudiant que ha sigut modificada i s'agafa l'arxiu .csv de la seva última correcció.

  A més, en aquesta vista es pot obrir l'informe de l'última correcció. Donat que únicament es genera un informe en format html, però l'arxiu css associat es trobava originalment a la banda del client, s'ha copiat l'arxiu css a la carpeta de *Courses* del servidor. Quan s'obre l'informe, s'edita el seu contingut per incloure la informació del fitxer .css de *Courses* i d'aquesta forma millorar la seva visualització.

  Per últim, es poden descarregar en format .zip els arxius principals de la correcció. Aquests arxius són els que s'ha mencionat que es troben dins de les carpetes de correccions de l'estudiant.

  Tant per obrir l'informe com per descarregar la correcció, de nou s'ha tingut en compte que l'entrega pot haver sigut individual o doble, i per obtenir l'última correcció s'ha comprovat quin és l'últim directori modificat.

- **Lliurament d'un estudiant:** la vista del lliurament de l'estudiant te forces similituds amb la vista de la correcció. Aquest cop, però, només es mostra la nota de la última entrega, doncs la resta no queden guardades perquè es sobreescriuen.

  Es pot descarregar l'útima entrega també, amb la diferència que, aquest cop, el directori, a més d'arxius, conté subdirectoris. Per gestionar-ho, es recorre recursivament l'arbre de subdirectoris per obtenir tots els fitxers i afegir-los al .zip.

  En els lliuraments, els informes de tots els alumnes es troben a la mateixa carpeta, *DeliveryReports*, per tant, es comprovarà quin d'aquests arxius que conté en el nom l'identificador de l'alumne que s'està buscant és l'últim que ha estat modificat. Torna a ser necessari editar l'arxiu .html de l'informe per afegir-hi el contingut de l'arxiu .css.

- **Sincronització amb el Server.BAT:** el servidor que inclou l'eina d'autocorrecció no s'actualitza automàticament, pel que fa falta afegir una funcionalitat que s'encarregui de fer-ho.

  Tot i no ser la funcionalitat ideal per actualitzar el servidor, aquest es reinicia des del programa cada vegada que s'actualitza informació que el servidor necessita saber. Aquesta implementació no és idònia perquè hi poden haver diversos usuaris concurrents fent

operacions amb el servidor, per tant, reiniciar-lo causaria problemes.

El meu tutor ha preparat unes noves comandes que haurien de poder actualitzar el servidor sense apagar-lo. Per desgràcia, per ara aquestes comandes no detecten bé les assignatures existents. Per aquest motiu s'ha mantingut, de moment, la funcionalitat de reiniciar el servidor.

6.4 Gestió de dependències

Per gestionar la gran quantitat de llibreries que es necessiten al llarg del desenvolupament, s'utilitza l'eina Maven[17]. Maven és una eina de gestió de projectes que s'utilitza per gestionar dependències, com eina de compilació i, en alguns casos, també com a eina de documentació. A més, Spring Boot utilitza Maven de forma predeterminada, per tant, el seu ús resulta imperatiu. Al llarg del projecte, Maven ha simplificat la importació de llibreries i dependències, de les quals pràcticament no m'he hagut de preocupar.

6.5 Testing

Durant el desenvolupament, s'ha utilitzat JUnit[18] i la llibreria de Spring Boot *SpringBootTest* per realitzar les proves del codi.

Abans d'acabar cada sprint, s'automatitzaven proves de les noves funcionalitats. En alguns casos, aquestes proves són unitàries, centrant-se únicament en una funcionalitat, i en altres les proves són d'integració, comprovant que el flux de treball i les crides entre funcions tenen el comportament esperat.

En el cas de les proves unitàries, s'han fet cridant a la funció que es vol comprovar amb els paràmetres o valors necessaris, per després comprovar que retorna el valor correcte.

Respecte les proves d'integració, aquestes simulen una crida HTTP per part del client, com si s'estigués utilitzant la pàgina. Altra vegada, es torna a comprovar que els resultats que retorna, en aquest cas al frontend, són els que pertoquen.

7 Resultats

Un cop finalitzat el període de desenvolupament, cal mirar enrere i veure quins dels objectius inicials s'han complert. Del total de 5 objectius, s'han pogut complir tots excepte els objectius 2 i 5.

Per una banda, de moment no tinc accés a un servidor de la UAB per poder-hi allotjar la web. Si durant el període entre l'entrega del dossier i la presentació del projecte rebo accés a un servidor, hi pujaré la web perquè es pugui servir de forma pública.

Per altre banda, l'objectiu 5, que consistia a afegir diferents estadístiques que puguessin consultar els professors, no l'he realitzat per falta de temps. És un dels objectius més complexos del projecte i es va considerar que no era crític, motiu pel qual es va deixar pel final.

Respecte al fet de necessitar reiniciar el Server.BAT per actualitzar la seva informació, no ho considero una funcionalitat correcta i per això m'agradaria implementar-ho de forma diferent, permetent la concurrència d'usuaris, tan aviat com tingui disponibles les noves comandes.

Els altres objectius, sent el primer el més crític, s'han realitzat satisfactòriament, complint amb les funcionalitats definides des d'un principi.

8 Conclusions i treball futur

Finalment, només queda extreure conclusions de com ha anat aquest projecte.

L'objectiu principal del projecte era aconseguir dotar d'una interfície més visual i senzilla al professorat a l'hora d'utilitzar l'eina de correcció, de manera que no fos necessari conèixer les comandes que presenta l'eina ni fer operacions manuals, com editar els arxius .ini de les assignatures manualment o gestionar els arxius .csv dels estudiants. Tot això s'havia de fer mantenint les funcionalitats de l'eina d'autocorrecció i, en tot cas, afegint-ne de noves.

La meva reflexió un cop acabat el projecte és positiva tot i no haver assolit tots els objectius. El projecte tenia inicialment un gran abast i només no ha sigut possible arribar a complir l'objectiu 5. En el cas de l'objectiu 2, no crec que sigui un error de planificació o temps, simplement no tinc accés en aquest moment a un ordinador de la UAB. De totes formes, un cop el tingui, pujar el projecte és un procés que hauria de ser ràpid.

Tenint present que és el primer projecte d'abast gran al que m'enfronto pel meu compte, em sento content de la forma que ho he gestionat, especialment, sobre tot el que he après al llarg del projecte. Penso que la corva d'aprenentatge ha sigut molt significativa en tots els nivells, tan en el coneixement per gestionar i organitzar un projecte com en el coneixement tècnic adquirit respecte les tecnologies utilitzades.

Durant el projecte, en especial al principi, vaig cometre errors o prendre decisions que més endavant vaig haver d'anar canviant, però això m'ha permès aprendre de les equivocacions i no tornar a ensopegar amb la mateixa pedra. Posant-ho en context, crec que és complicat que tot surti a la perfecció en un primer projecte que ja comença a tenir certa magnitud. Per tant, és important saber sobreposar-se a les complicacions que vagin sorgint, i em sento satisfet de com ho he fet durant aquests mesos.

Sobre el treball futur, és evident que les dues primeres coses a fer serien pujar-ho a un servidor i afegir les

funcionalitats de les estadístiques. A partir d'aquí, la web ja tindria totes les funcionalitats especificades des d'un principi, i seria qüestió de saber què més pot ser d'interès pel professorat de les assignatures, així com obtenir el seu *feedback* de les funcionalitats ja implementades.

AGRAÏMENTS

En primer lloc, vull expressar el meu agraïment al meu tutor pels seus consells i disposició.

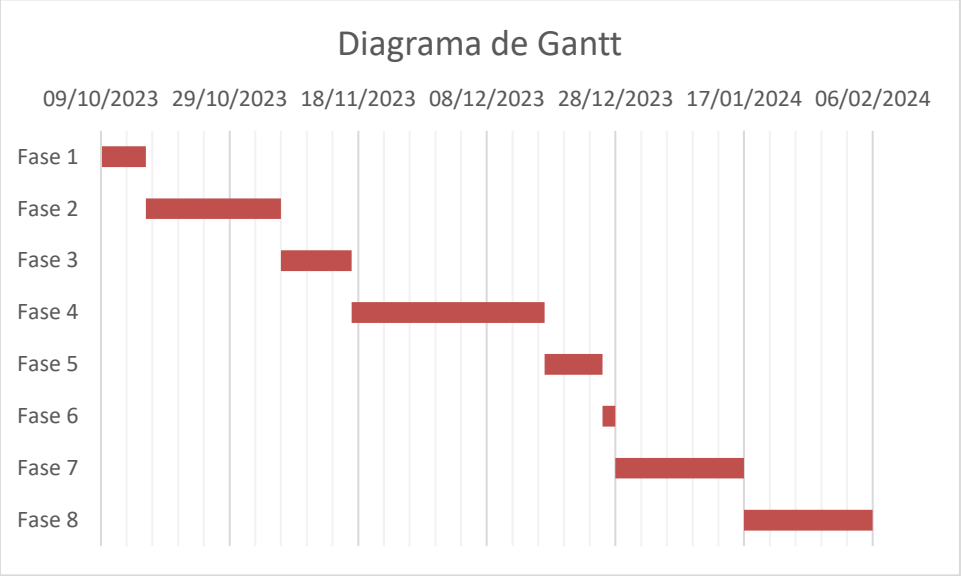
Finalment, agrair a la meva família pel seu suport i motivació al llarg de tots aquests anys.

BIBLIOGRAFIA

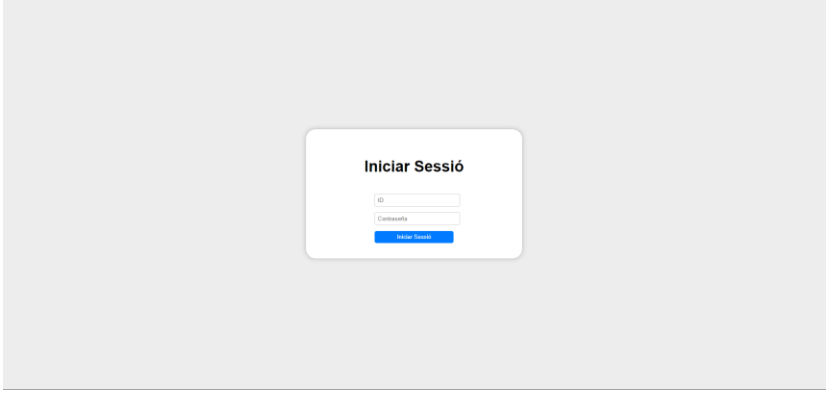
[1] *OnlineJudge. Github.* https://github.com/topics/online-judge.
[2] *Jutge.org. UPCommons.* http://upcommons.upc.edu/handle/2117/91202.
[3] *OnlineJudge. OnlineJudge* https://onlinejudge.org/index.php.
[4] *Scrum: ¿Qué es el product backlog? PYM.* https://programacionymas.com/blog/scrum-product-backlog
[5] *¿Qué es un archivo CSV y para qué sirve? OpenWebinars.* https://openwebinars.net/blog/que-es-thymeleaf/.
[6] *INI (extensión de archivo). Wikipedia.* https://es.wikipedia.org/wiki/INI_(extensi%C3%B3n_de_archivo).
[7] *IntelliJ IDEA – the leading Java and Kotlin IDE. JetBrains.* https://www.jetbrains.com/idea/promo/?source=google&medium=cpc&campaign=EMEA_en_ES_IDEA_Branded&term=intellij&content=602143185340&gad=1&gclid=Cj0KCQjwpompBhDZARIsAFD_Fp-ztepAZzkUvQmX3DmqQGg9bkju8GzhSPwJ35h4Nns-tHPyyh_WPzCMaArLMEALw_wcB.
[8] *¿Qué es Jira Software? Atlasssian.* https://www.atlassian.com/es/software/jira/guides/getting-started/introduction#what-is-jira-software.
[9] *¿Qué es un tablero Kanban y cómo utilizarlo? Explicación bàsica. Businessmap.* https://businessmap.io/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban.
[10] *Java (lenguaje de programación). Wikipedia.* https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n).
[11] *¿Qué es Spring Boot? ArquitecturaJava.* https://www.arquitecturajava.com/que-es-spring-boot/.
[12] *¿Qué es Spring Framework y por qué usarlo? OpenWebinars.* https://openwebinars.net/blog/conoce-que-es-spring-framework-y-porque-usarlo/.
[13] *HTML: Lenguaje de etiquetas de hipertexto. Mozilla.org.* https://developer.mozilla.org/es/docs/Web/HTML.
[14] *CSS. Mozzila.org.* https://developer.mozilla.org/es/docs/Web/CSS.
[15] *¿Qué es thymeleaf? OpenWebinars.* https://openwebinars.net/blog/que-es-thymeleaf/.
[16] *Github. Wikipedia.* https://es.wikipedia.org/wiki/GitHub.
[17] *Java: ¿Qué es Maven? ¿Qué es el archivo pom.xml? CampusMVP.* https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx.
[18] *JUnit. Wikipedia.* https://es.wikipedia.org/wiki/JUnit.

APÈNDIX

A1. DIAGRAMA DE GANTT



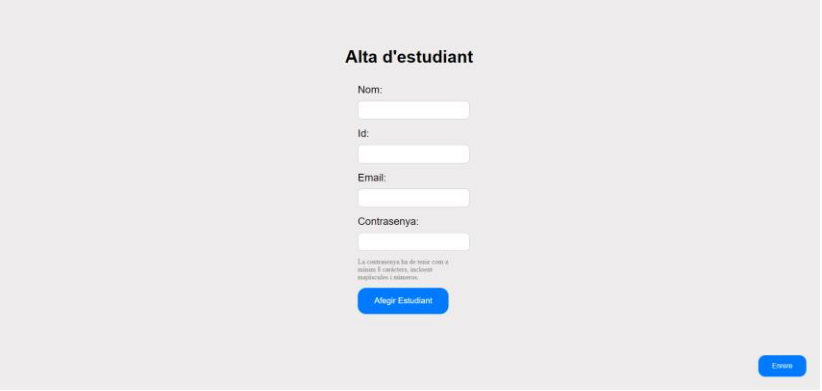
A2. VISTES PRINCIPALS DE LA WEB



Vista inici de sessió

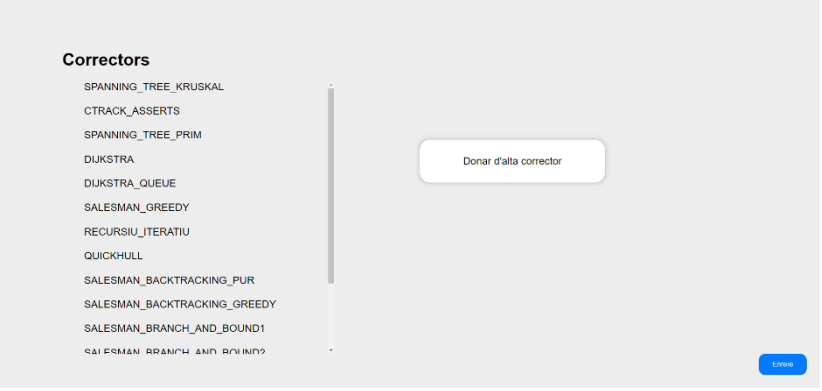
Vista d'una assignatura

Vista dels alumnes



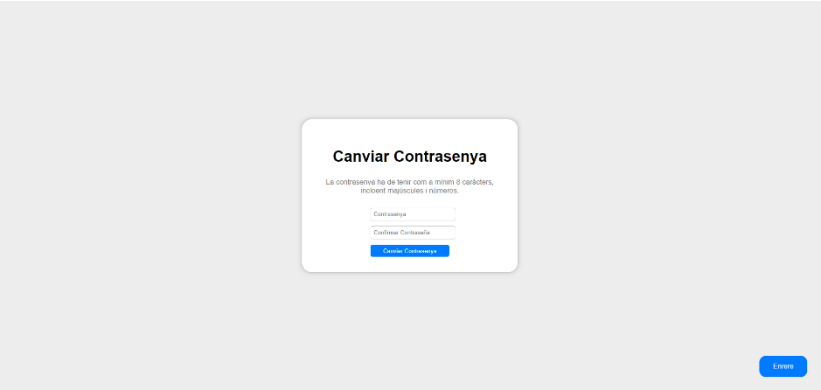
Vista per donar d'alta un alumne

Vista per donar d'alta estudiants des de CSV



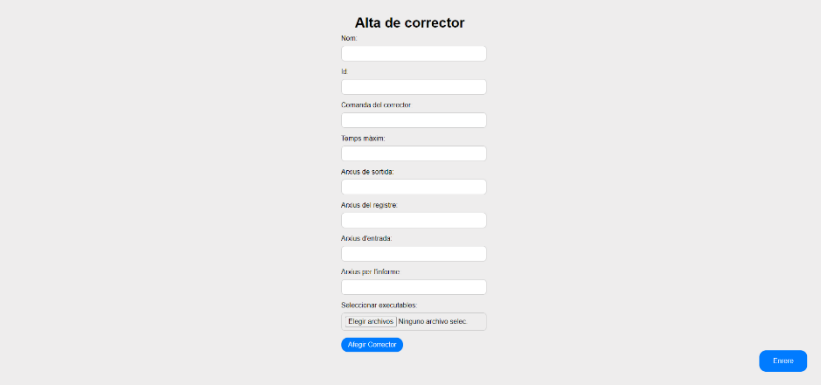
Vista dels correctors

Vista de les correccions d'un estudiant

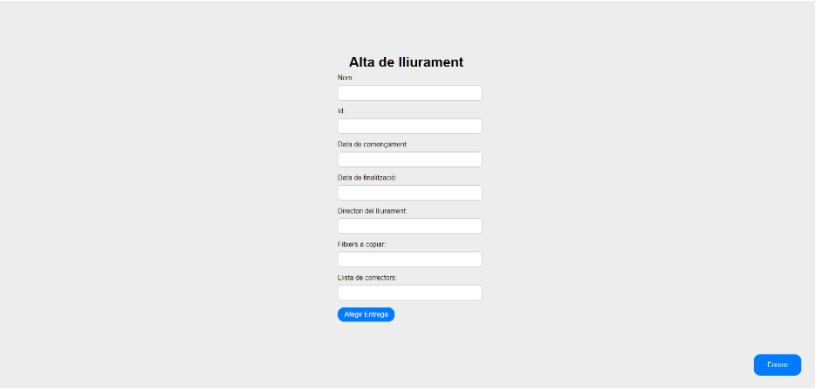


Vista per canviar contrasenya de l'estudiant

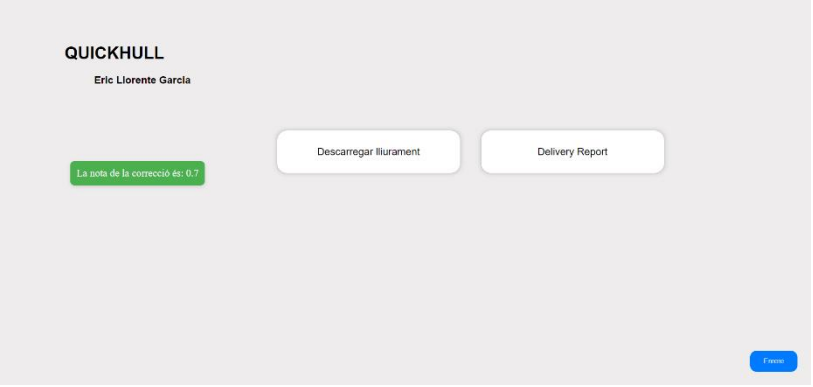
Vista per donar d'alta una assignatura



Vista per donar d'alta un corrector



Vista per donar d'alta lliurament



Vista del lliurament d'un estudiant