UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU EN ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Creació d'una Plataforma Web de Cursos Online d'Estudiants per a Estudiants Universitaris

---

Autor: Pablo María Arranz Pou

Director:	Dr. Xavier Baró Solé
Realitzat a:	Departament de Matemàtiques i Informàtica
Barcelona,	Juny 2024

Abstract

This project develops an online course platform that allows users to access, manage, and interact with digital educational materials. Utilizing a combination of technologies, including Django for the backend, React for the frontend, and Amazon Web Services for data storage and video streaming.

Resum

Aquest projecte desenvolupa una plataforma de cursos en línia que permet als usuaris accedir, gestionar i interactuar amb materials educatius digitals. Utilitzant una combinació de tecnologies, incloent-hi Django per al backend, React per al frontend, i Amazon Web Services per l'emmagatzematge de dades i la transmissió de vídeo.

Resumen

Este proyecto desarrolla una plataforma de cursos en línea que permite a los usuarios acceder, gestionar e interactuar con materiales educativos digitales. Utilizando una combinación de tecnologías, incluyendo Django para el backend, React para el frontend y Amazon Web Services para el almacenamiento de datos y la transmisión de vídeo.

Índex

1 Introducció

La plataforma proposada es concep com un complement al panorama educatiu actual, especialment en l'àmbit universitari. Es dissenya com un sistema per facilitar un espai on els estudiants universitaris puguin aprofundir i enriquir el seu aprenentatge en assignatures específiques, així com permetre als exestudiants d'assignatures o fins i tot de la universitat, utilitzar els seus coneixements previs mitjançant la creació de cursos. Aquest enfocament pretén promocionar un desenvolupament acadèmic col·laboratiu, on l'intercanvi de coneixements i experiències entre estudiants i exestudiants esdevé una font de valor afegit.

El nucli de la plataforma es centra en la interacció estudiant-estudiant, un enfocament que facilita que els usuaris adquireixin coneixements de manera més efectiva i que també desenvolupin habilitats com la comunicació, el raonament crític i la capacitat d'ensenyament. Addicionalment, permetent als exestudiants crear i oferir contingut, la plataforma obre una oportunitat per a aquests individus de generar ingressos recurrents, així com de contribuir de manera significativa al seu entorn acadèmic, establint un ecosistema d'aprenentatge compartit.

Els estudiants disposen de l'autonomia per escollir quins continguts consumir i determinar quins creadors expliquen millor les assignatures o s'adapten millor a les seves necessitats d'aprenentatge. Aquest enfocament vol promoure que l'educació pot ser personalitzada i dirigida pels mateixos aprenents, respectant així les diverses maneres d'aprendre i les preferències individuals respecte a l'estil d'ensenyament.

Subratllant que la plataforma està pensada com a complement i no com a substitut de l'educació universitària tradicional, es vol aportar aquest valor addicional dins de l'ecosistema educatiu. L'objectiu no és suplantar les metodologies d'ensenyament existents ni els procediments oficials, sinó ampliar l'experiència educativa oferint recursos addicionals i perspectives diferents que poden facilitar la consolidació del coneixement i la resolució de dubtes que puguin emergir dins del marc d'una educació formal.

2 Objectius

En aquest projecte s'han establert uns objectius per guiar el desenvolupament i garantir que es compleixin els objectius plantejats per cobrir les necessitats d'una plataforma d'aquest tipus. A continuació, es detallen els objectius específics a assolir:

- **Aprendre Amazon AWS**: Un dels objectius principals és adquirir coneixements pràctics sobre Amazon Web Services (AWS), ja que el cloud computing és una habilitat molt valuosa en el món laboral actual.
- **Desenvolupar un MVP o una Fase Avançada del Producte**: L'objectiu és aconseguir un producte mínim viable (MVP) o arribar a una fase avançada de desenvolupament que permeti la validació del concepte i la demostració de les funcionalitats clau de la plataforma.
- **Assolir els Requisits Principals i Secundaris**: L'objectiu és complir amb tots els requisits funcionals i no funcionals definits, prioritzant els principals però també abordant els secundaris si és possible. Aquests requisits es definiran a la fase d' anàlisi.
- **Aprendre Tecnologia HLS i Seguretat en Vídeos**: Aprofundir en la tecnologia HTTP Live Streaming (HLS) per a la transmissió de vídeos i explorar tècniques per protegir els vídeos contra la pirateria.

3 Anàlisi

3.1 Requisits

En aquesta secció, definirem els requisits funcionals i no funcionals de la nostra plataforma. Aquests requisits defineixen les necessitats dels usuaris perquè sigui capaç de proporcionar una experiència completa i funcional al producte final.

3.1.1 Funcionals

Els requisits funcionals descriuen les funcionalitats específiques que la plataforma ha de proporcionar per satisfer les necessitats generals dels usuaris. Aquests requisits inclouen:

- **Registre i Accés**: Capacitat per crear un compte proporcionant informació bàsica com nom d'usuari, contrasenya, correu electrònic etc.
- **Gestió de Cursos**: Possibilitat de crear, modificar i eliminar cursos, incloent-hi la càrrega de vídeos, documents i qüestionaris.
- **Visualització de Cursos**: Accés a una biblioteca de cursos amb opcions per filtrar segons diversos criteris com grau d'estudis, universitat i assignatura.
- **Subscripció a Cursos**: Opció per adquirir accés complet als cursos a través d'un procés de subscripció.
- **Interacció Dinàmica**: Capacitat per participar en qüestionaris, comentar sobre els cursos i compartir valoracions.
- **Seguiment del Progrés**: Funcionalitats per monitorar el progrés personal dins dels cursos i proporcionar analítiques d'aprenentatge.
- **Consum de Vídeos**: Funcionalitat per al consum de vídeos de forma segura i interactiva amb preguntes incrustades.
- **Gestió de Subscripcions**: Crear un sistema de subscripció per facilitar la subscripció a cursos i la gestió dels ingressos generats pels creadors de contingut.

3.1.2 No Funcionals

Els requisits no funcionals descriuen els criteris de funcionament del sistema que no estan directament relacionats amb funcionalitats específiques, però que són requerits per al correcte funcionament de la plataforma. Aquests requisits inclouen:

- **Web**: La plataforma ha d'estar disponible com a aplicació web.
- **Escalabilitat**: Capacitat de gestionar augments substancials de càrrega de treball per acomodar un creixement significatiu del volum d'usuaris. Això inclou l'ús de serveis de computació en núvol per escalar automàticament els recursos de la plataforma.

- **Compatibilitat**: Capacitat per funcionar correctament en diversos navegadors, assegurant una experiència consistent per a tots els usuaris. Garantir la compatibilitat en Google Chrome principalment i fer-ho funcionar a altres plataformes.
- **Rendiment**: Resposta eficient a les sol·licituds dels usuaris, assegurant temps de càrrega mínims i una experiència d'usuari fluida.
- **Testing**: Implementar tests unitaris per augmentar la cobertura de codi i garantir la consistència d'aquest.

3.2 Històries d'Usuari

En aquesta plataforma tenim dos tipus principals d'usuaris que interactuen dins de l'ecosistema: l'Usuari Consumidor i l'Usuari Creador. Cada un d'aquests té un conjunt específic de necessitats i objectius dins de la plataforma, que s'han de satisfer mitjançant les funcionalitats i característiques definides en les històries d'usuari.

A continuació, es descriuen diverses històries d'usuari que descriuen les principals interaccions dels usuaris amb la plataforma, específicament des de la perspectiva de l'Usuari Consumidor, l'Usuari Creador i l'Usuari No Autenticat.

3.2.1 Usuari No Autenticat

Els Usuaris No Autenticats són aquells que naveguen per la plataforma sense tenir un compte o no haver accedit a ella. Tot i que tenen accés limitat, poden explorar els cursos disponibles. Totes les funcionalitats dels Usuaris no Autenticats també les tenen tant l'Usuari Creador com l'Usuari Consumidor.

Principals

1. **Veure i Buscar Cursos:** Com a Usuari No Autenticat, vull poder veure i buscar cursos disponibles a la plataforma per decidir si val la pena registrar-me.

2. **Veure Informació del Curs:** Com a Usuari No Autenticat, vull poder veure la informació general dels cursos, com el temari i les valoracions, i informar-me de què tipus de contingut hi ha.

3.2.2 Usuari Consumidor

Els Usuaris Consumidors són els estudiants universitaris que utilitzen la plataforma per complementar els seus coneixements de les assignatures de les quals estan matriculats o simplement persones amb ganes d'aprendre, que volen aprendre contingut a través dels cursos oferts. Aquests usuaris busquen una experiència d'aprenentatge flexible però estructurada, que pugui adaptar-se als seus horaris i necessitats d'aprenentatge individuals. Les següents històries d'usuari descriuen les interaccions clau per a l'Usuari Consumidor dins de la plataforma.

Principals

1. **Accedir al contingut del curs:** Com a Usuari Consumidor, vull poder accedir al contingut gratuït del curs per poder veure de què va o com és per a decidir si m'interessaria subscriurem al curs.

2. **Visualitzar vídeos amb preguntes integrades:** Com a Usuari Consumidor, vull poder visualitzar vídeos que s'aturin per mostrar-me preguntes relacionades amb el contingut, per millorar la meva comprensió de la matèria.

3. **Participar en qüestionaris:** Com a Usuari Consumidor, vull poder respondre qüestionaris per autoavaluar el meu aprenentatge i entendre millor els conceptes del curs.

4. **Cursos del meu grau de la meva Universitat:** Com a Usuari Consumidor, vull poder veure els cursos disponibles segons el meu grau i universitat, per trobar el contingut més rellevant i aplicable a les meves necessitats acadèmiques.

5. **Veure Documents:** Com a Usuari Consumidor, vull poder veure documents, per adquirir coneixements en els apunts o esquemes proporcionats pel creador.

6. **Subscripció a curs:** Com a Usuari Consumidor, vull poder subscriure'm a un curs per poder veure tot el contingut del curs inclòs el contingut no gratuït.

Secundaris

1. **Valorar i comentar en els cursos:** Com a Usuari Consumidor, vull poder valorar i comentar en els cursos als quals m'he subscrit, per compartir la meva opinió i experiència amb altres usuaris.

2. **Veure el progrés del meu aprenentatge:** Com a Usuari Consumidor, vull poder veure el meu progrés dins del curs, per ser conscient de la meva evolució i motivar-me a continuar aprenent.

3.2.3 Usuari Creador

Els Usuaris Creadors comparteixen els seus coneixements a través de la creació de cursos. Aquests usuaris aspiren a distribuir el seu contingut educatiu arribant a un públic ampli de futurs estudiants de l'assignatura. Esperen que els seus cursos estiguin segurs i que no es distribueixi a persones no subscrites al curs. Les següents històries d'usuari descriuen les interaccions clau per a l'Usuari Creador dins de la plataforma.

Principals

1. **Crear i gestionar cursos:** Com a Usuari Creador, vull poder crear i gestionar els meus cursos, incloent la càrrega de vídeos i documents, així com la creació de qüestionaris, per oferir contingut educatiu de qualitat als estudiants.

2. **Establir preus per als cursos:** Com a Usuari Creador, vull poder establir preus pels meus cursos, per poder monetitzar el meu esforç i el contingut que produeixo i poder fer ofertes dels preus del curs.

3. **Revisar ingressos generats:** Com a Usuari Creador, vull poder revisar els ingressos generats pels meus cursos, per tenir un control sobre la meva activitat econòmica dins de la plataforma.

4. **Gestionar qüestionaris integrats en vídeos:** Com a Usuari Creador, vull poder integrar qüestionaris dins dels vídeos dels meus cursos, per fomentar un aprenentatge interactiu i permetre als estudiants autoavaluar-se en temps real.

5. **Gestionar Documents:** Com a Usuari Creador, vull gestionar els documents per compartir i revisar el material que ofereixo als meus estudiants.

6. **Gestionar Vídeos:** Com a Usuari Creador, vull gestionar els vídeos del meu curs per compartir i revisar els vídeos que estic proporcionant als meus estudiants.

7. **Gestionar Qüestionaris:** Com a Usuari Creador, vull gestionar els qüestionaris del meu curs per compartir i revisar les avaluacions que estic oferint als meus estudiants.

Secundaris

1. **Veure analítiques del curs:** Com a Usuari Creador, vull poder accedir a dades i analítiques detallades sobre el rendiment dels meus cursos, incloent-hi subscripcions i valoracions, per entendre millor el meu públic i millorar el contingut ofert.

2. **Personalitzar el meu perfil:** Com a Usuari Creador, vull poder personalitzar el meu perfil dins de la plataforma, incloent-hi enllaços a les meves xarxes socials i una descripció personal, per augmentar la meva visibilitat i connectar amb els estudiants.

3. **Respondre als comentaris i valoracions:** Com a Usuari Creador, vull poder respondre als comentaris i valoracions dels meus cursos, per mantenir una comunicació activa amb els estudiants i construir una comunitat d'aprenentatge.

4 Tecnologies

Per construir la nostra plataforma, hem seleccionat un conjunt de tecnologies que ens permeten oferir una bona experiència d'usuari i una arquitectura escalable. En aquesta secció, discutirem les tecnologies clau utilitzades, incloent Django amb Django REST Framework per al backend i React per al frontend.

4.1 Backend

Django és un framework de desenvolupament web de codi obert escrit en Python, destacat per promoure un desenvolupament ràpid i un disseny clar i pragmàtic. A més hem escollit Django a causa del temps disponible per fer el projecte, fer ús de les eines que ens proporciona per centrar-nos a programar funcionalitats en lloc de gestionar tasques inicials.[1]

4.1.1 Django

- **ORM Potent:** El sistema d'Object-Relational Mapping (ORM) de Django permet als desenvolupadors interactuar amb bases de dades de manera intuïtiva i eficient, facilitant la creació i manipulació de models de dades sense escriure SQL.
- **Rendiment i Escalabilitat:** Django és capaç de gestionar augments substancials de càrrega de treball, fet que el fa ideal per a aplicacions que anticipen un creixement significatiu del volum d'usuaris.
- **Seguretat:** Django inclou mecanismes de seguretat integrats que protegeixen les aplicacions contra diversos atacs comuns, com ara SQL injection, cross-site scripting, cross-site request forgery i clickjacking. Això permet als desenvolupadors concentrar-se en la construcció de funcionalitats sense haver de preocupar-se excessivament per vulnerabilitats comunes.

Django REST Framework (DRF) amplia les capacitats de Django, proporcionant eines robustes per a la construcció d'APIs web escalables i mantenibles.[1]

4.1.2 Django REST Framework

- **Autenticació i Permisos:** DRF implementa una àmplia gamma de mecanismes d'autenticació i permisos que asseguren que només els usuaris autoritzats poden realitzar operacions sensibles, protegint així tant les dades com les funcionalitats de l'aplicació.
- **Serialització:** Ofereix suport potent per a la serialització de dades complexes, facilitant la integració amb diferents tipus de bases de dades, tant SQL com NoSQL, a través de l'ORM integrat de Django.

4.2 Base de Dades

Per al maneig de les dades, hem optat per utilitzar SQLite, una base de dades relacional lleugera que s'integra de manera nativa amb Django. SQLite ofereix una solució per a aplicacions de grandària mitjana sense la necessitat d'un servidor de base de dades separat. És important destacar que SQLite s'utilitza principalment en l'entorn de desenvolupament i proves, mentre que per a l'entorn de producció es consideraran altres opcions més robustes.

4.2.1 SQLite

- **Facilitat d'Integració:** Django ofereix suport incorporat per SQLite, el que facilita la configuració i el maneig de la base de dades sense necessitat d'instal·lacions complexes o configuracions addicionals.
- **Gestió Eficient de Recursos:** SQLite consumeix molt pocs recursos del sistema, el que la fa ideal per a desenvolupaments que requereixen una configuració ràpida i una gestió eficient dels recursos.
- **Ideal per a Desenvolupament i Proves:** La simplicitat de SQLite la fa perfecta per a l'ús en entorns de desenvolupament i proves, permetent als desenvolupadors implementar i provar funcionalitats ràpidament.

Tot i que SQLite és ideal per al desenvolupament i les proves, no és la millor opció per a l'entorn de producció. Per a la fase de desplegament, s'estudiaran altres opcions de bases de dades més robustes com PostgreSQL o MySQL, que ofereixen millor rendiment, escalabilitat i capacitat de gestió de grans volums de dades.

4.3 Frontend

React és una biblioteca de JavaScript desenvolupada per Facebook per construir interfícies d'usuari de manera eficient i amb un codi previsible. És àmpliament mantinguda per Facebook juntament amb la seva comunitat de desenvolupadors,[2]

4.3.1 React

- **Declaratiu:** React facilita la creació d'interfícies d'usuari interactives. Permet dissenyar vistes simples per a cada estat de l'aplicació, i React s'encarrega d'actualitzar i renderitzar de manera eficient els components adequats quan les dades canvien. Les vistes declaratives fan que el codi sigui més previsible, simple d'entendre i fàcil de depurar.[2]
- **Basat en Components:** Permet construir components encapsulats que gestionen el seu propi estat, que després es poden compondre per crear interfícies d'usuari complexes. Com que la lògica del component està escrita en JavaScript en lloc de plantilles, és fàcil passar dades complexes a través de l'aplicació i mantenir l'estat fora del DOM.[2]

- **Aprèn un cop, escriu arreu:** React no fa suposicions sobre la resta del teu stack tecnològic, així que pots desenvolupar noves funcionalitats en React sense reescriure codi existent. React també pot renderitzar-se en el servidor usant Node i potenciar aplicacions mòbils usant React Native.[2]

4.4 AWS

Amazon Web Services (AWS) és una plataforma de serveis al núvol àmpliament utilitzada per la seva robustesa, escalabilitat i la gran varietat de serveis que ofereix. AWS proporciona una infraestructura global que permet desplegar aplicacions de manera ràpida i segura, facilitant així la gestió de recursos i serveis al núvol.[3]Hem seleccionat AWS per diverses raons tècniques:

- **Àmplia Gamma de Serveis:** AWS ofereix una extensa gamma de serveis, incloent-hi emmagatzematge, computació, bases de dades, analítica, xarxes i molt més. Això permet que la nostra plataforma es beneficiï de solucions integrades i altament especialitzades, adaptades a les nostres necessitats.

- **Escalabilitat:** La infraestructura d'AWS està dissenyada per escalar de manera automàtica segons la demanda de l'aplicació, assegurant així que la plataforma pugui gestionar increments significatius en el volum de trànsit i dades sense comprometre el rendiment.

- **Seguretat:** AWS inclou mesures de seguretat avançades, com ara el xifratge de dades en trànsit i en repòs, controls d'accés rigorosos i auditories de seguretat. Aquestes característiques asseguren que les dades dels usuaris estiguin protegides contra accessos no autoritzats i vulnerabilitats.

- **Popularitat i Suport de la Comunitat:** AWS és la plataforma de serveis al núvol més utilitzada al món[4], amb una gran comunitat de desenvolupadors i una àmplia documentació. Això facilita l'accés a recursos, tutorials i bones pràctiques que poden ajudar en la resolució de problemes i en l'optimització de la plataforma.

5 Planificació

Abans de començar el desenvolupament de la plataforma, es va dur a terme una planificació per assegurar un desenvolupament lògic i coordinat de les diferents tasques, fent que facin abans tasques i seguidament fent tasques que depenguin de les anteriors. Aquesta planificació es va organitzar en tres grans categories: Anàlisi, Funcionalitats i Documentació, i es va distribuir en un període de tres mesos i una setmana. Cada bloc del desenvolupament inclou el frontend i el backend. Als annexos es pot veure una descripció de cada fase.

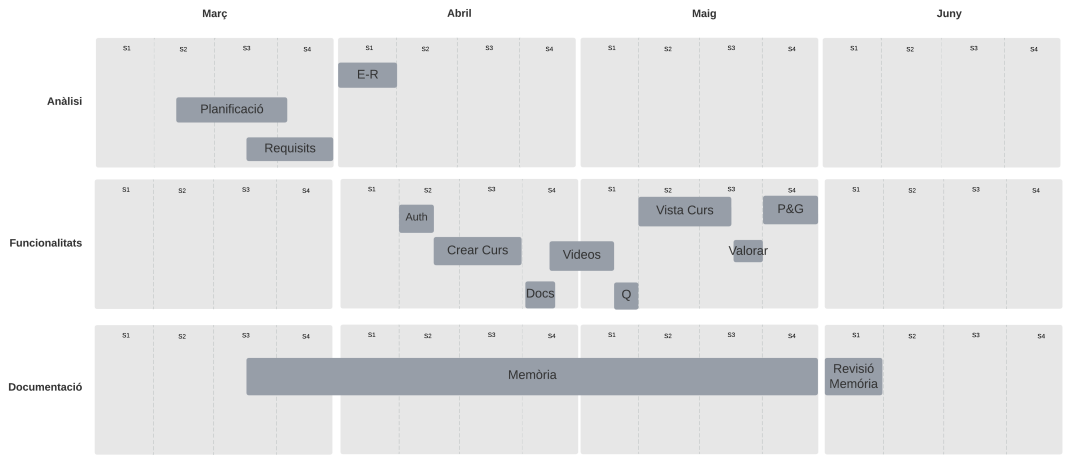

Figura 1: Planificació Original del desenvolupament de la plataforma

6 Entitats i Relacions

Després de definir les històries d'usuari i els requisits de la plataforma, passem a la fase de disseny de la base de dades. A la nostra plataforma, l'elaboració d'un esquema entitat-relació és important per a garantir que la base de dades sigui capaç de suportar les funcionalitats descrites a les històries d'usuari.

Aquest model ens ajudarà a identificar les necessitats de dades de la plataforma, com per exemple com es gestionen els cursos, com es registren i autentiquen els diferents tipus d'usuaris i com es relacionen els usuaris amb els cursos i els materials didàctics.

Els detalls complets sobre cada entitat, incloent-hi els camps específics i les seves relacions, es poden trobar a l'annex d'aquest document.

6.1 Estructura Acadèmica

A la nostra aplicació, volem garantir que els cursos es puguin diferenciar per a cada usuari, identificant clarament que pertanyen a universitats, facultats i graus específics. Per aconseguir-ho, hem creat una estructura que permet diferenciar el grau, l'assignatura, la facultat i el grau de l'assignatura concreta. Així, hem dissenyat una estructura acadèmica que permet una selecció detallada i jeràrquica, de manera que, a partir d'una assignatura, es pugui obtenir tota la informació rellevant.

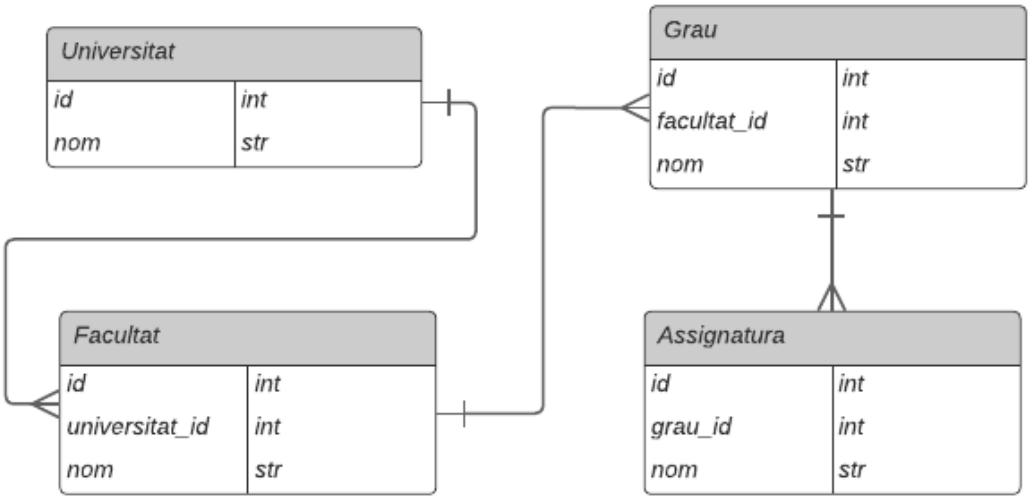

Figura 2: Esquema Entitat-Relació d'Universitats, Graus, Facultats i Assignatures

6.2 Usuaris: Consumidor i Creador

La gestió dels usuaris és important per diferenciar les dades que recollim per a cada tipus d'usuari i com cada tipus d'usuari interactua amb la plataforma. Com hem vist anteriorment, tenim dos tipus d'actors dins la plataforma: l'usuari creador i l'usuari consumidor. A l'hora de dissenyar la plataforma, s'ha decidit que tots dos actors heretin de la classe `Usuario`, la qual hereta d'`AbstractUser`, que és l'usuari base de Django. Aquí podem veure l'ús que es dona a cadascun dels models:

- **Usuari** (`Usuario`): L'entitat `Usuario` és la base per a tots els usuaris de la plataforma. Aquesta classe hereta d'`AbstractUser` de Django, que proporciona camps bàsics com nom d'usuari, correu electrònic, contrasenya, etc. Aquesta estructura permet una gestió senzilla de l'autenticació i l'autorització dels usuaris.
- **Consumidor** (`Consumidor`): L'entitat `Consumidor` està destinada als usuaris que consumeixen els continguts dels cursos. Aquest és el model que es crea quan un usuari es dona d'alta a la plataforma. Com en el cas dels creadors, cada consumidor té una relació unívoca amb l'entitat `Usuario` mitjançant una clau primària. Això permet al consumidor accedir a cursos, consumir vídeos, interactuar amb qüestionaris, etc.
- **Creador** (`Creador`): L'entitat `Creador` està destinada als usuaris que creen i gestionen cursos dins la plataforma. Cada creador té una relació unívoca amb l'entitat `Usuario` mitjançant una clau primària. Aquesta relació pot existir o no; és a dir, un usuari primer ha de ser consumidor per donar-se d'alta com a creador, però no és obligatori que tots els usuaris esdevinguin creadors. Això permet al creador tenir accés a funcionalitats específiques com la creació, l'edició i la publicació de cursos, però també la possibilitat de consumir cursos com Usuari Consumidor.

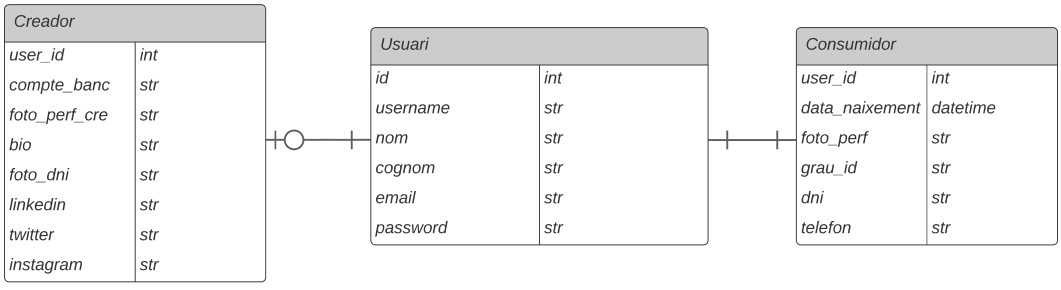

Figura 3: Esquema E-R d'Usuaris, Creadors i Consumidors

6.3 Cursos

A la nostra plataforma, els cursos són l'element central i es relacionen amb diverses entitats que defineixen el contingut didàctic associat a cada curs. Els cursos es componen de diverses entitats anomenades Lliçons, necessàries perquè el creador pugui

estructurar de manera ordenada i seguint un objectiu d'aprenentatge el contingut del curs. Aquestes lliçons poden contenir diferents tipus de contingut: vídeos, documents o qüestionaris, que constitueixen el contingut del curs perquè els usuaris consumidors puguin accedir-hi. A més, l'usuari creador pot decidir deixar elements d'una lliçó de forma gratuïta, és a dir que només farà falta, estar autenticat, per poder consumir-ho.

- **Curs** (`Curso`): És l'entitat principal que agrupa tot el contingut didàctic. Cada curs està associat a un creador i a una assignatura específica.
- **Lliçó** (`Leccion`): Les lliçons representen les unitats didàctiques del curs. Cada lliçó està associada a un curs específic i pot contenir diversos tipus de contingut didàctic.
- **Document** (`Documento`): Les lliçons poden incloure documents que proporcionen material de lectura o recursos addicionals.
- **Qüestionari** (`Cuestionario`): Les lliçons poden incloure qüestionaris són utilitzats per què els alumnes del curs puguin autoavaluar-se o consolidar els seus coneixements.
- **Vídeo** (`Video`): Les lliçons poden incloure vídeos educatius que proporcionen una explicació visual del contingut, i poden incloure preguntes per poder aprendre d'una forma més interactiva.

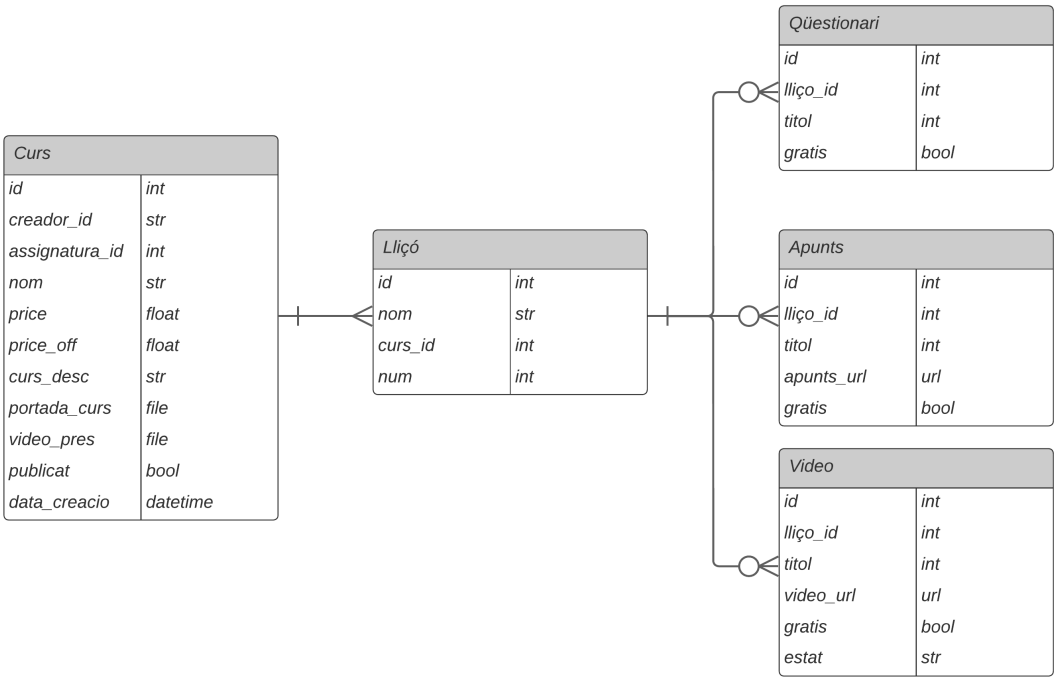

Figura 4: Esquema E-R de Cursos i Continguts Didàctics

6.4 Subscripcions i Valoracions

Les subscripcions i valoracions són elements que gestionen la interacció dels usuaris amb els cursos. Permeten registrar quins usuaris consumidors està subscrit a quins cursos i quines valoracions els hi han donat.

- **Subscripció** (`Suscripcion`): Els usuaris consumidors poden subscriure's a diversos cursos. Aquesta relació es representa mitjançant l'entitat `Suscripcion`, que vincula un consumidor amb un curs, incloent-hi informació sobre la data de subscripció, la data de finalització, si el curs ha estat pagat al creador i per quina quantitat s'ha pagat aquella subscripció. La variable booleana que indica si s'ha pagat o no és útil per gestionar els pagaments als creadors. Un creador ha de poder cobrar les subscripcions encara després que la subscripció hagi caducat i hem de controlar quines subscripcions li han estat pagades.
- **Valoració** (`Valoracion`): Els consumidors poden valorar els cursos als quals estan subscrits. Aquesta relació es representa mitjançant l'entitat `Valoracion`, que vincula un consumidor amb un curs, incloent-hi informació sobre el text de la valoració, la data i la puntuació donada al curs.

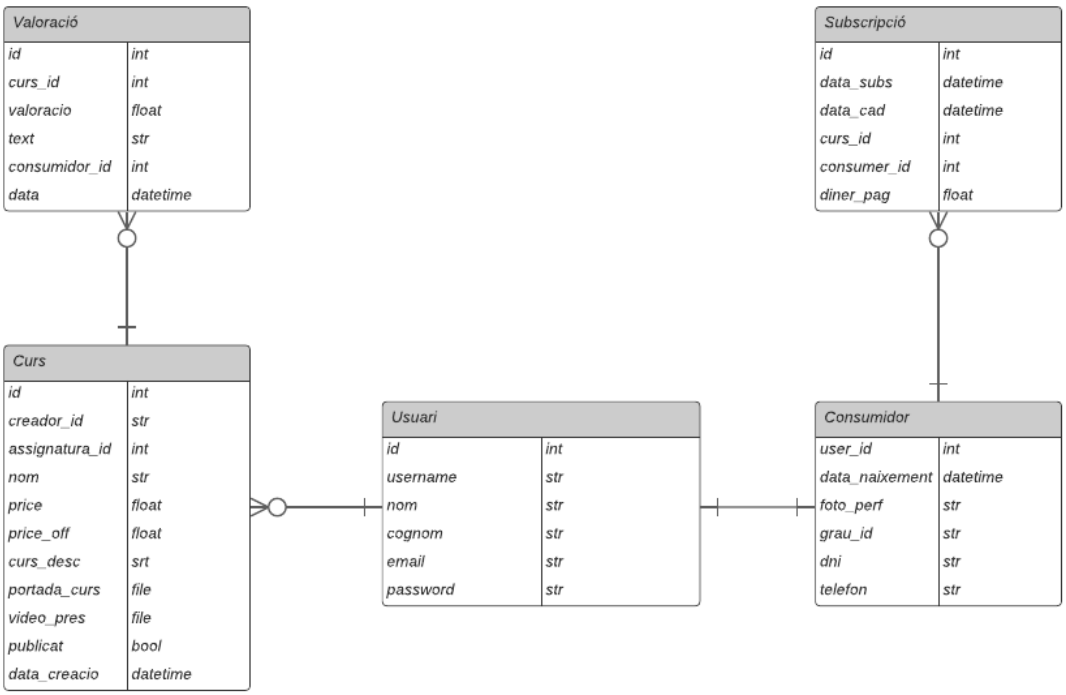

Figura 5: Esquema E-R de Cursos, Subscripcions i Valoracions

6.5 Gestió de Contingut: Preguntes de Qüestionaris i Vídeos

A la nostra plataforma, les preguntes de qüestionaris i vídeos són elements que serveixen per avaluar o enriquir el coneixement dels estudiants i oferir una experiència

d'aprenentatge interactiva. Aquestes preguntes estan estructurades de manera que permeten una gestió eficient i una integració adequada amb les lliçons i els cursos corresponents.

- **Pregunta de Qüestionari** (`PreguntaCuestionario`): Les preguntes de qüestionaris estan dissenyades per avaluar el coneixement dels estudiants sobre una lliçó específica. Cada pregunta està associada a un qüestionari concret.
- **Pregunta de Vídeo** (`PreguntasVideo`): Les preguntes de vídeo aporten una interactivitat addicional als vídeos educatius. Aquestes preguntes poden aparèixer en moments específics del vídeo per avaluar la comprensió del contingut mentre l'usuari el consumeix.

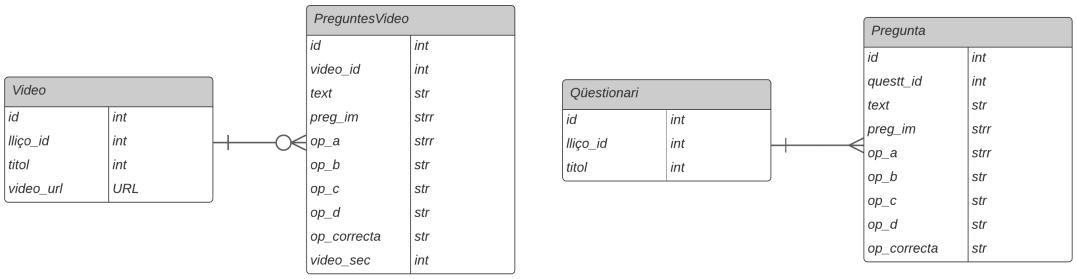

Figura 6: Esquema E-R de Preguntes de Qüestionaris i Preguntes de Vídeo

6.6 Resum esquema d'entitat relació

- **Estructura Acadèmica**: Els cursos es diferencien clarament per universitats, facultats i graus. Això es gestiona a través de les entitats `Universitat`, `Facultat`, `Grau` i `Assignatura`. Aquestes entitats estan jeràrquicament relacionades per assegurar una estructura acadèmica clara i coherent, on cada assignatura indica inequívocament la universitat, facultat i grau a la qual pertany.
- **Gestió d'Usuaris**: Hi ha dues classes principals d'usuaris: els consumidors i els creadors. Tots dos hereten de la classe `Usuario`, que al seu torn hereta d'`AbstractUser`. Cada usuari sempre té un rol de consumidor quan es registra a la plataforma i pot donar-se d'alta com a creador si ho desitja.
- **Cursos**: Els cursos són l'element central de la plataforma. Cada curs està associat a un creador i a una assignatura específica. A més, un curs està compost per diverses lliçons (`Leccion`), cadascuna de les quals pot contenir documents (`Documento`), qüestionaris (`Cuestionario`) i vídeos (`Video`).
- **Subscripcions**: Les subscripcions permeten als consumidors accedir als cursos. Cada subscripció està associada a un consumidor i a un curs, incloent-hi informació sobre la data de subscripció, la data de caducitat i l'estat de pagament.

- **Valoracions**: Les valoracions permeten als consumidors avaluar els cursos. Cada valoració està associada a un consumidor i a un curs, i inclou una puntuació, un text descriptiu i la data de la valoració.

- **Preguntes de Qüestionaris i Vídeos**: Les preguntes de qüestionaris (`PreguntaCuestionario`) i les preguntes de vídeos (`PreguntasVideo`) són elements que permeten als alumnes autoavaluar-se. Aquestes preguntes estan associades respectivament a qüestionaris i vídeos específics, oferint així una experiència d'aprenentatge interactiva.

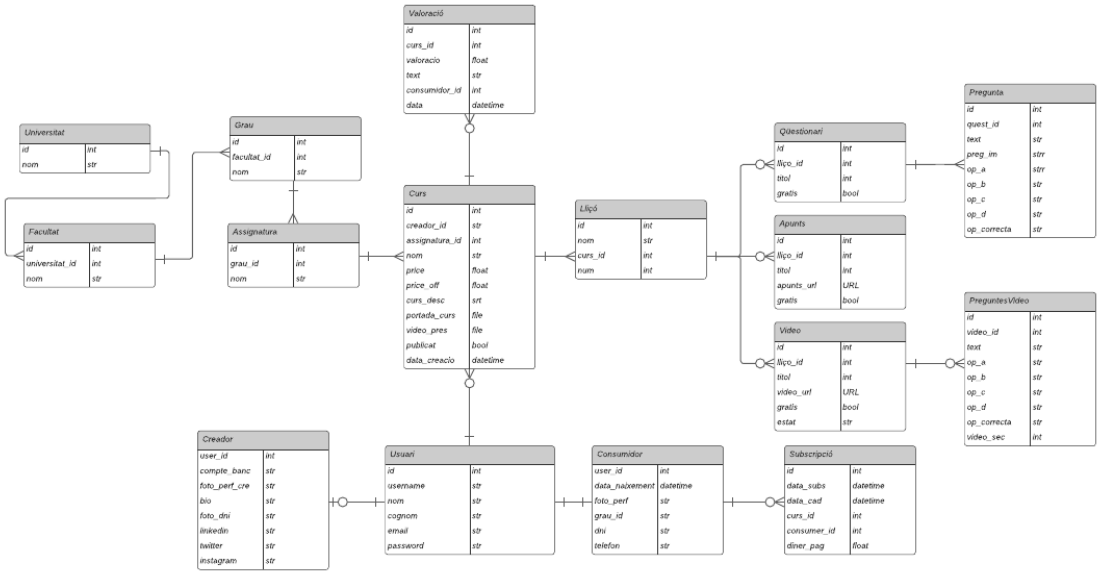

Figura 7: Esquema E-R Complet de la Plataforma

7 Funcionalitats

7.1 Autenticació

L'autenticació en la nostra plataforma és el procés mitjançant el qual verifiquem la identitat dels usuaris que intenten accedir a ella. Amb això garantim que només els usuaris autoritzats puguin accedir a recursos específics i gestionar la seva informació de manera segura.

7.1.1 Implementació de l'Autenticació Simple JWT amb Django Rest Framework

A la nostra plataforma volem assegurar una comunicació segura entre frontend (React) i el backend (Django). Una de les metodologies més eficaces per a aconseguir això és a través de l'ús de Tokens Web JSON (JWT). Aquesta secció s'explica com ho hem implementat mitjançant Simple JWT dins del Django Rest Framework. Per veure una explicació més detallada de com funciona aquest sistema d'autenticació, anar als annexos.

```
path('token', TokenObtainPairView.as_view(), name='token_obtain_pair'),
path('token/refresh', TokenRefreshView.as_view(), name='token_refresh'),
```

Veiem aquí els endpoints pel token de refresh i el d'obtenir el token. A més, aquesta llibreria permet definir polítiques de token que en el nostre cas a Django les hem definit a `settings.py` de la següent forma:

```
SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(hours=5),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
    'ROTATE_REFRESH_TOKENS': True,
    'BLACKLIST_AFTER_ROTATION': True,
    'ALGORITHM': 'HS512',
}
```

Per veure que són i com funcionen aquestes polítiques de token es pot trobar una explicació als annexos d'aquest document.

7.1.2 Configuració d'Axios

Per a establir una connexió segura i configurada específicament amb el servidor, utilitzem una instància personalitzada d'Axios. A continuació es mostra el codi per configurar aquesta instància:

```
const customAxios = axios.create({
    baseURL: process.env.REACT_APP_API_URL +
        process.env.REACT_APP_API_VERSION,
});
```

Aquest codi configura una nova instància d'Axios amb una URL base del endpoint i la versió d'aquesta, que es pren de les variables d'entorn de l'aplicació, assegurant que totes les peticions es realitzin amb aquesta base.

7.1.3 Gestió de l'accessToken

L'ús de `interceptors` en Axios ens permet interceptar les peticions o respostes abans que siguin gestionades pels `then` o `catch`. En el nostre cas, utilitzem un `interceptor` de peticions per afegir l'`accessToken` a les capçaleres de les peticions sortints, sempre que aquest estigui disponible:

```
customAxios.interceptors.request.use(
    (config) => {
        const accessToken = localStorage.getItem('accessToken');
        if (accessToken) {
            config.headers.Authorization = 'Bearer ${accessToken}';
        }
        return config;
    },
    (error) => {
        return Promise.reject(error);
    }
);
```

Aquest fragment de codi comprova si existeix un `accessToken` emmagatzemat en el `localStorage`. Si es troba, l'afegeix a la capçalera `Authorization` de les peticions HTTP com un token Bearer. Això permet que les peticions siguin autenticades pel servidor de manera automàtica, facilitant l'accés a recursos protegits sense necessitat de validar l'usuari en cada petició.

Gestió del Refresh Token Per assegurar que l'usuari es mantingui autenticat fins i tot quan l'`accessToken` expiri, hem implementat un mecanisme per refrescar el token automàticament:

```
const refreshToken = async () => {
    const refreshToken = localStorage.getItem('refreshToken');
    if (refreshToken) {
        try {
            const response = await axios.post(REFRESH_TOKEN_URL, {
            refresh: refreshToken,
            });
            const { access } = response.data;
            localStorage.setItem('accessToken', access);
            return access;
        } catch (err) {
            console.error('Refresh token failed', err);
            localStorage.removeItem('accessToken');
            localStorage.removeItem('refreshToken');
            throw err;
        }
    }
```

```
    throw new Error('No refresh token available');
};
```

Aquest codi defineix una funció `refreshToken` que comprova si hi ha un `refreshToken` emmagatzemat en el `localStorage`. Si es troba, fa una petició a l'endpoint de refresc del token per obtenir un nou `accessToken`, que s'emmagatzema de nou en el `localStorage`.

```
customAxios.interceptors.response.use((response) => response,
    async (error) => {
    const originalRequest = error.config;
    if (error.response.status === 401 && !originalRequest._retry) {
        originalRequest._retry = true;
        try {
            const newAccessToken = await refreshToken();
            customAxios.defaults.headers.common['Authorization'] =
                'Bearer ${newAccessToken}';
            originalRequest.headers['Authorization'] = 'Bearer
                ${newAccessToken}';
            return customAxios(originalRequest);
        } catch (err) {
            console.error('Token refresh failed', err);
        }
    }
    return Promise.reject(error);
    }
);
```

Aquest interceptor de respostes intercepta les respostes amb un error 401 (no autoritzat). Si es troba amb aquest error, intenta refrescar l'`accessToken` utilitzant la funció `refreshToken`. Si la renovació té èxit, actualitza les capçaleres de la petició original amb el nou `accessToken` i reenvia la petició. En cas que no tingui èxit eliminem els tokens que tinguem guardats perquè hem vist que ja no funcionen.

7.1.4 Login i Registre

Quan ens loguejem, demanem un token si l'usuari i la contrasenya són correctes. El backend gestiona la sol·licitud de login de la forma següent:

```
@api_view(['POST'])
def login(request):
    User = get_user_model()
    username = request.data.get('username')
    password = request.data.get('password')

    try:
        user = User.objects.get(username=username)
```

```
        if user.check_password(password):
            refresh = RefreshToken.for_user(user)
            tokens = {
                'refresh': str(refresh),
                'access': str(refresh.access_token),
            }
            return Response({
                'message': 'User authenticated successfully',
                'tokens': tokens,
            })
        else:
            return Response(
                {'error': 'Usuario y/o contraseña inválido/s'},
                status=400
            )

    except ObjectDoesNotExist:
        return Response(
        {'error': 'Usuario y/o contraseña inválido/s'},
        status=400
    )
```

Aquest codi gestiona el procés d'inici de sessió comprovant si l'usuari existeix i si la contrasenya proporcionada és correcta. Si l'usuari és vàlid, es generen els tokens d'accés i de refresc, que són retornats al client. Si les credencials no són correctes, es retorna un missatge d'error.

Pel que fa al frontend, aquest s'encarrega d'enviar les credencials al backend i gestionar la resposta:

```
axios.post('${process.env.REACT_APP_API_URL}
    ${process.env.REACT_APP_API_VERSION}/login', {
        username,
        password
        }).then((res) => {
        if (res.status === 200) {
            const { access, refresh } = res.data.tokens;
            localStorage.setItem('accessToken', access);
            localStorage.setItem('refreshToken', refresh);
            setIsErrorVisible(false);
            window.location.href = '/';
            }
        }).catch((error) => {
        [...] ** GESTIÓ DEL ERROR **
});
```

El frontend envia una sol·licitud POST a l'endpoint de login del backend amb les credencials de l'usuari. Si la resposta és exitosa, s'extrauen els tokens d'accés i de refresc de la resposta i es guarden al `localStorage` per mantenir la sessió de l'usuari. En cas que hi hagi un error, es mostra un missatge d'error a l'usuari.

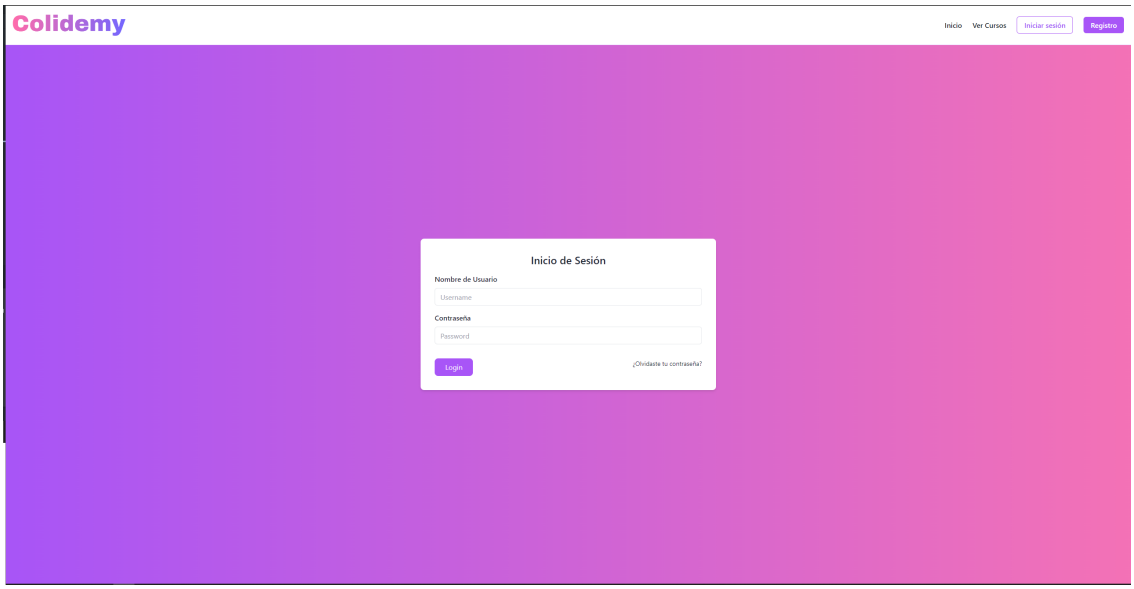

Figura 8: Captura del login a la aplicació Web

Pel que fa el registre el que fem es simplement crear un usuari i generar-li un token de accés i de refresc directament, omplint tant la informació personal necessària i la del grau del grau.

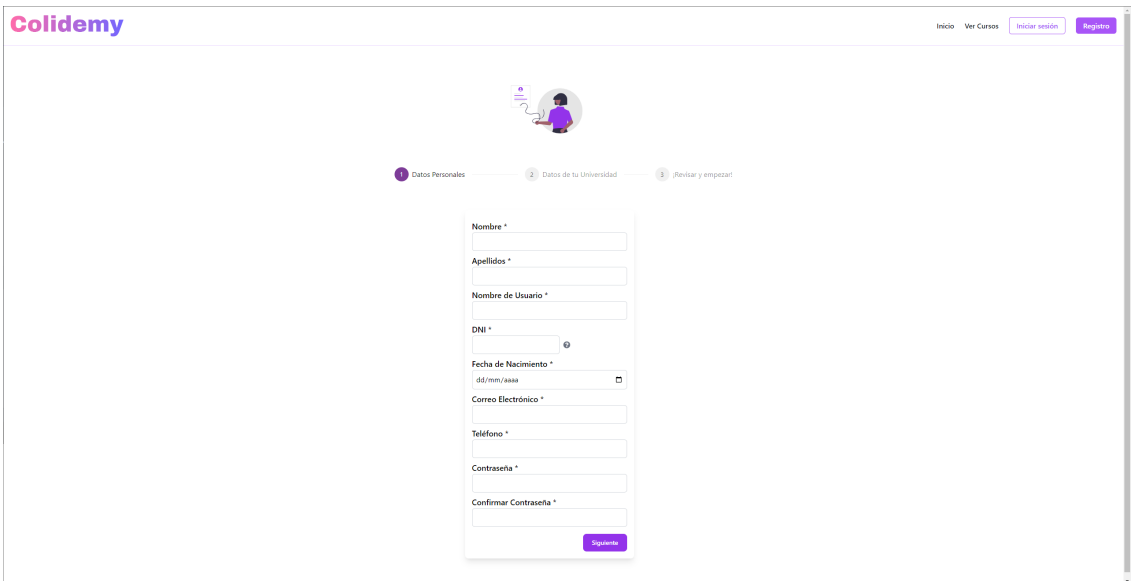

Figura 9: Primera part del Registre de Informació Personal

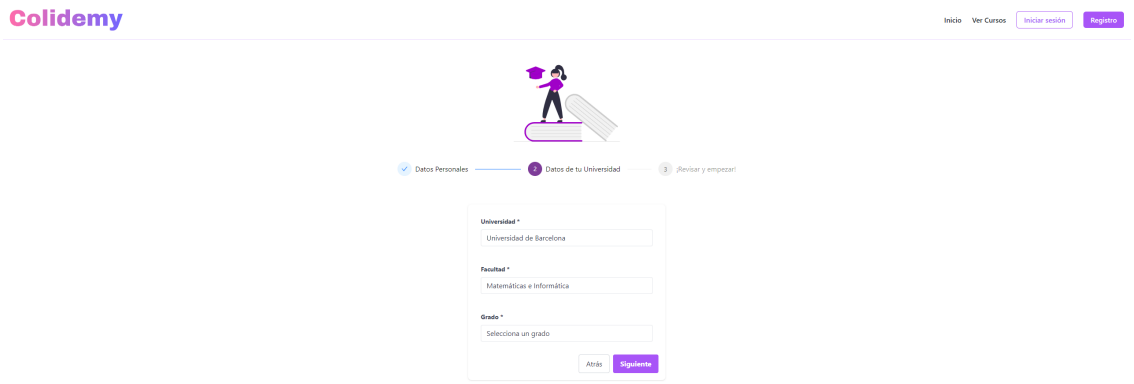

Figura 10: Segona part del Registre amb la Informació Acadèmica

7.2 Implementació de Vídeos Interactius(HLS + AWS)

Per oferir una bona experiència d'usuari a la nostra plataforma, hem implementat la transmissió de vídeos mitjançant el protocol HTTP Live Streaming (HLS).

7.2.1 HLS

HLS funciona mitjançant la segmentació del contingut de vídeo en una sèrie de fitxers petits de durada curta, habitualment de 2 a 10 segons cadascun. Aquests segments són enviats al client (com ara un navegador o aplicació mòbil) com a una sèrie de descàrregues en petits fragments a través de HTTP estàndard. Això permet al reproductor començar a reproduir el vídeo mentre segueix descarregant-se la resta del contingut. HLS també inclou la funcionalitat de reproducció adaptativa, ajustant la qualitat del vídeo en funció de la qualitat de la connexió de l'usuari.[5]

L'estructura basada en llistes de reproducció d'HLS ens permet oferir una experiència d'usuari més rica, incloent-hi opcions com àudio alternatiu per a diferents idiomes i adaptacions de qualitat dinàmica sense interrompre la reproducció del contingut.

Tot i que HLS és conegut pel seu ús en streaming adaptatiu, en el nostre cas específic, el vídeo es codifica a una sola qualitat. Això vol dir que no hi ha ajustament dinàmic de la qualitat del vídeo segons l'ample de banda de l'usuari; en canvi, es proporciona una única llista de reproducció M3U8 i diversos fitxers .TS corresponents a aquesta qualitat fixa. Aquesta elecció es fa per donar seguretat als creadors de contingut i dificultar la descàrrega del vídeo.

7.2.2 Fitxer M3U8 i la seva estructura

Un fitxer M3U8 és un format de fitxer basat en text que s'utilitza per llistar els segments de fitxers de mitjans de comunicació per a la transmissió en viu a través del protocol HLS. Aquí tens un exemple d'un fitxer M3U8 que mostra com es podria estructurar per utilitzar-se en la transmissió de contingut de vídeo.

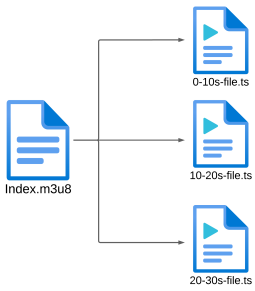

Figura 11: Esquema de com funcionen els arxius m3u8 i ts en HLS

Aquest tipus de fitxer conté referències a diversos fitxers de segments (generalment en format TS - Transport Stream) i opcions de configuració que el reproductor utilitza per reproduir el vídeo o l'àudio. Aquest és un exemple simple d'un fitxer M3U8:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.000,
segment0.ts
#EXTINF:10.000,
segment1.ts
#EXTINF:10.000,
segment2.ts
#EXTINF:10.000,
segment3.ts
#EXT-X-ENDLIST
```

Descripció dels Components:

- **#EXTM3U:** Aquesta etiqueta és l'encapçalament i indica que el fitxer és una llista de reproducció M3U.[6]
- **#EXT-X-VERSION:** Especifica la versió de la llista de reproducció.[6]
- **#EXT-X-TARGETDURATION:** Aquesta etiqueta indica la durada màxima de qualsevol segment de mitjans en la llista de reproducció.[6]
- **#EXT-X-MEDIA-SEQUENCE:** Especifica el número de seqüència del primer segment de TS en la llista de reproducció. Aquesta seqüència incrementa amb cada nou segment.[6]

- **#EXTINF:** Aquesta etiqueta especifica la durada del segment següent en segons. El valor que segueix a #EXTINF és la durada del segment i es mesura en segons. Després d'aquesta etiqueta es troba l'URL o el camí del fitxer del segment.[6]
- **#EXT-X-ENDLIST:** Aquesta etiqueta indica que no hi ha més segments en la llista de reproducció i que és la fi de la llista.[6]

7.2.3 Integració amb AWS S3 Utilitzant Boto3

Per a la gestió dels vídeos, utilitzem AWS S3 (Simple Storage Service), un servei d'emmagatzematge en el núvol ofert per Amazon Web Services. La biblioteca Boto3 de Python facilita la interacció amb AWS, permetent pujar i gestionar els fitxers.[7] A S3 creem un bucket, que és un contenidor bàsic en el servei Amazon S3. Es pot comparar amb un directori en un sistema de fitxers, però a escala del núvol. Aquesta estructura de bucket permet a AWS S3 organitzar i gestionar les dades de forma eficient. Cada bucket que crees en AWS S3 té un nom únic global i pot contenir un nombre il·limitat de fitxers, anomenats objectes en aquest context.

7.2.4 Implementació amb Boto3

Primerament, generem el client boto3 amb les credencials que hem creat (vegeu configuració d'AWS en els annexos), on hem creat un usuari amb drets d'administració i li hem generat unes claus que afegim al nostre backend. Per seguretat en pujar el codi a Internet, hem definit variables d'entorn.

```
import boto3
from botocore.exceptions import ClientError
from django.conf import settings

client = boto3.client('s3', region_name='eu-west-1',
                      aws_access_key_id=settings.AWS_ACCESS_KEY_ID,
                      aws_secret_access_key=settings.AWS_SECRET_ACCESS_KEY)

bucket_name = settings.AWS_STORAGE_BUCKET_NAME
```

7.2.5 AWS Elastic Transcoder

AWS Elastic Transcoder és un servei a la núvol que ens permet convertir arxius multimèdia d'un format a un altre de manera eficient i escalable[8]. En el nostre cas, utilitzem aquest servei per convertir els vídeos dels usuaris al format HLS.

Al principi del desenvolupament es feia la transcodificació en local, però veient que podíem aprofitar el servei oferit per AWS vam decidir fer-ho allà. En lloc de processar els vídeos directament en el nostre servidor, transferim aquesta tasca a AWS Elastic Transcoder. Això redueix la càrrega del nostre backend i ens permet gestionar millor els recursos.

El funcionament és senzill: simplement enviem els vídeos a Elastic Transcoder juntament amb les especificacions del format de sortida que volem (que és M3U8 i arxius .ts com hem vist abans), i el servei es fa càrrec de la conversió. Un cop finalitzada la conversió, els nous vídeos en format estaran directament al nostre bucket creat anteriorment per rebre convertir i guardar els resultats del nostre Elastic Transcoder. Tot això ho farem en una funció Lambda que veurem en la següent secció.

7.2.6 Funcions Lambda

Les funcions Lambda d'AWS les fem servir per automatitzar el procés de transcodificació i gestió de vídeos HLS en la nostra plataforma. Lambda és un servei de computació serverless que permet executar codi en resposta a esdeveniments sense necessitat d'administrar servidors.[9] Les funcions Lambda tenen tres components principals:

- **El Desencadenador (Trigger)**: és el component que defineix l'esdeveniment que ha de passar per activar la funció Lambda. Aquest esdeveniment pot ser diverses accions, com ara la pujada d'un fitxer a un bucket S3, una actualització en una base de dades, una notificació d'un servei, entre altres.
- **La Funció**: La funció Lambda és el codi que s'executa quan es dispara el desencadenador. Aquesta funció defineix les accions que es prendran quan es detecti l'esdeveniment esperat. La funció pot realitzar diverses operacions, com ara processar dades, executar operacions en bases de dades, interaccionar amb altres serveis d'AWS, enviar notificacions, etc.
- **El Destí (Destination)**: El destí és el component que defineix què passa o què es fa amb el resultat després que la funció Lambda hagi completat la seva execució. El destí pot ser un altre servei d'AWS, com S3, DynamoDB, entre altres.

A continuació, es descriu com s'utilitzen les funcions Lambda en el nostre flux de treball de processament de vídeos.

7.2.7 Configuració de Funció Lambda

Quan un usuari carrega un vídeo a través del frontend, el fitxer es desa al nostre bucket S3 predefinit. Aquest esdeveniment de pujada desencadena una funció Lambda que inicia el procés de transcodificació.

El desencadenador de la nostra funció Lambda és un esdeveniment de pujada (PUT) d'un fitxer amb l'extensió .mp4 a un bucket S3. Cada vegada que es puja un fitxer amb aquesta extensió al bucket especificat, es dispara la funció Lambda configurada per aquest esdeveniment. La configuració que s'ha fet servir per desencadenar la funció Lambda en el nostre projecte és la següent:

```
ARN del bucket: arn:aws:s3:::colidemy-v2-api
Cuenta de origen: ***************************
Entidad principal del servicio: s3.amazonaws.com
ID de instrucción: ***************************
isComplexStatement: No
Nombre de la notificación: ***************************
Prefijo: videos/
Sufijo: .mp4
Tipos de eventos: s3:ObjectCreated:Put
```

Figura 12: Esdeveniment que activa funció lambda

7.2.8 Implementació de Funcions Lambda

Com hem vist en l'apartat anterior el moment en el qual un fitxer .mp4 es desa en el nostre bucket S3, una funció Lambda s'activa, ara veurem que és el que volem que la nostra funció Lambda faci per la nostra funcionalitat.

Després de definir el client d'S3 i el Elastic Transcoder, la funció verifica que el fitxer pujat és un .mp4 i que segueix la convenció de nomenclatura esperada (`upload.mp4`). Seguidament la funció llista tots els fitxers presents en el directori del bucket S3 i elimina tots els fitxers excepte el `upload.mp4` perquè la nova pujada substituirà a tots els m3u8 i .ts existents. Després, la funció inicia un treball de transcodificació utilitzant AWS Elastic Transcoder, especificant el pipeline amb el preset `1351620000001-200010` que és el de HLS amb un bitrate de 2 megabits per segon[10] de forma constant, així com els paràmetres de sortida.

```
 # Iniciar el trabajo de transcodificación
job_response = et_client.create_job(
            PipelineId=pipeline_id,
            Input={'Key': key, 'FrameRate': 'auto',
            'Resolution': 'auto', 'AspectRatio': 'auto',
            'Interlaced': 'auto', 'Container': 'auto'},
                Outputs=[{'Key': 'output_video',
                'PresetId': '1351620000001-200010',
```

```
                'SegmentDuration': '10'}],
                OutputKeyPrefix=directory
            )
job_id = job_response['Job']['Id']
print(f"Trabajo de transcodificación iniciado con ID: {job_id}")

# Esperar a que el trabajo de transcodificación se complete
while True:
    job_status = et_client.read_job(Id=job_id)
    status = job_status['Job']['Status']
    if status in ['Complete', 'Canceled', 'Error']:
        break
    time.sleep(30)
    print(f"Trabajo de transcodificación finalizado con estado: {status}")
```

La funció espera que la transcodificació es completi i verifica l'estat del treball. Si és completat correctament, imprimeix un missatge d'èxit. Tot i que és possible configurar un destí diferent per als fitxers transcodificació, en aquest cas decidim utilitzar el mateix bucket S3.

7.2.9 Pujada del video desde el frontend

En la nostra plataforma, els vídeos es carreguen al costat del client per no saturar el backend i optimitzar el procés. Quan un usuari vol carregar un vídeo, el client demana al backend una URL preautoritzada al nostre bucket de S3. Aquesta URL permet fer un PUT del vídeo directament a S3. Un cop el backend proporciona la URL preautoritzada, el client utilitza aquesta URL per fer un PUT, carregant així el vídeo directament a S3. Amb aquest mètode, deleguem tota la càrrega de la pujada del vídeo a AWS i al Client, garantint una pujada eficient i fiable sense sobrecarregar els nostres recursos del backend. En aquest esquema es veu el procediment:

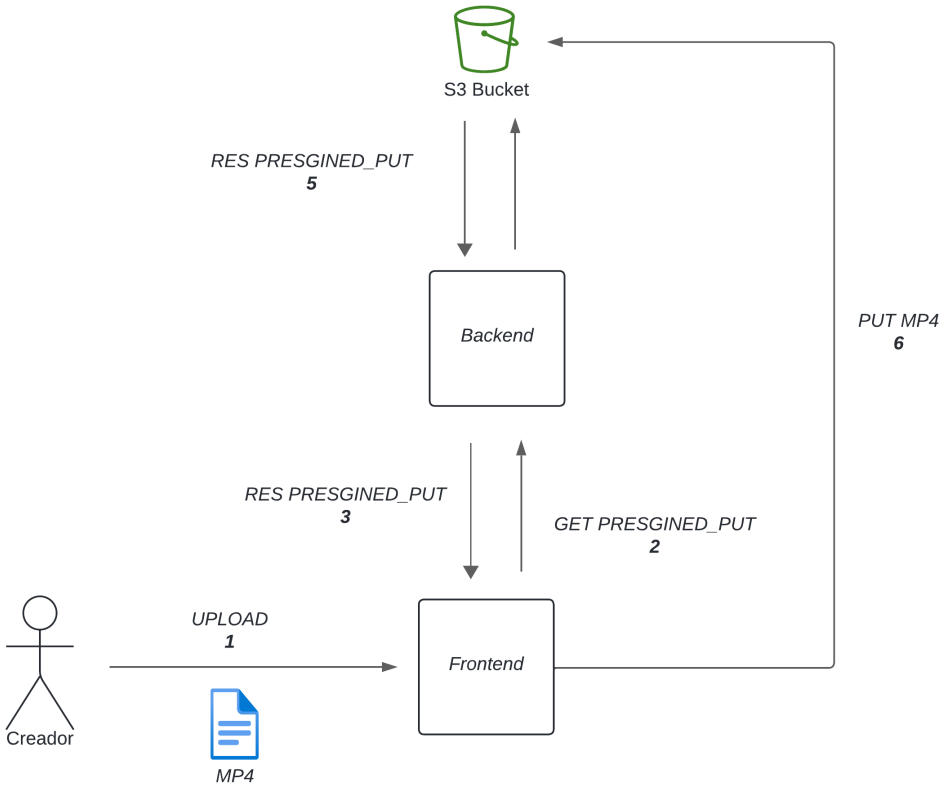

Figura 13: Esquema de pujada de vídeos

En el frontend, gestionem la pujada de l'arxiu demanant al backend la URL preautoritzada de tipus PUT. Quan ens la retorna, enviem el fitxer seleccionat amb una funció PUT d'axios amb els headers i el tipus d'arxiu que estem pujant.

```
const handleUpload = async () => {
        [...]
        const response = await customAxios.get(‘sign-put-url/video/${video_id}‘);
        const presignedUrl = response.data

        await axios.put(presignedUrl, selectedFile, {
            headers: {
                    ’Content-Type’: selectedFile.type
            },
            [...]
            });
            [...]
    };
```

Aquí es mostra la funció que demana la URL PUT per poder pujar el vídeo a s3 del endpoint que truquem des del frontend, també li enviem el `content-type` perquè volem fer-la servir per als documents també la funció d'AWS.

```
@api_view([’GET’])
```

```
@permission_classes([IsAuthenticated])
def generate_presigned_url_put(request, file_type, id):

    [:..] ** COMPROVACIO USUSARI I SI ÉS SEU EL CONTINGUT **


    object_name = f"videos/{id}/upload.mp4"
    content_type = 'video/mp4'
    presigned_post = generate_presigned_put(object_name, content_type)

    if presigned_post is None:
        return Response(
        {'error': 'Error al generar la URL para subir'}
        , status=500
        )

    file_url = f"https://{settings.AWS_STORAGE_BUCKET_NAME}.
                    amazonaws.com/videos/{id}/output_video.m3u8"
    video.video_url = file_url
    video.save()
```

Veiem que disposem de la funció `generate_presigned_put` definida al nostre fitxer `aws.py`. Aquesta funció genera una URL preautoritzada que permet pujar fitxers a una ubicació específica dins del bucket S3. Donat un tipus d'arxiu i l'objecte concret (és a dir, la ruta dins de S3), la funció crea aquesta URL, facilitant així la càrrega de contingut directament a la ubicació desitjada a S3.

```
    def generate_presigned_put(object_name, content_type):

    try:
        response = client.generate_presigned_url('put_object',
                                    Params={'Bucket': bucket_name,
                                            'Key': object_name,
                                            'ContentType': content_type},
                                    ExpiresIn=3600)
    except ClientError as e:
        return None

    return response
```

7.2.10 Integració amb l'Aplicació

Amb els apartats anteriors, hem definit com gestionem la pujada de vídeos i com cada part de l'arquitectura que hem creat funciona perquè tot funcioni correctament i de forma òptima. En el següent esquema podem veure el comportament global del procés de pujada de vídeos:

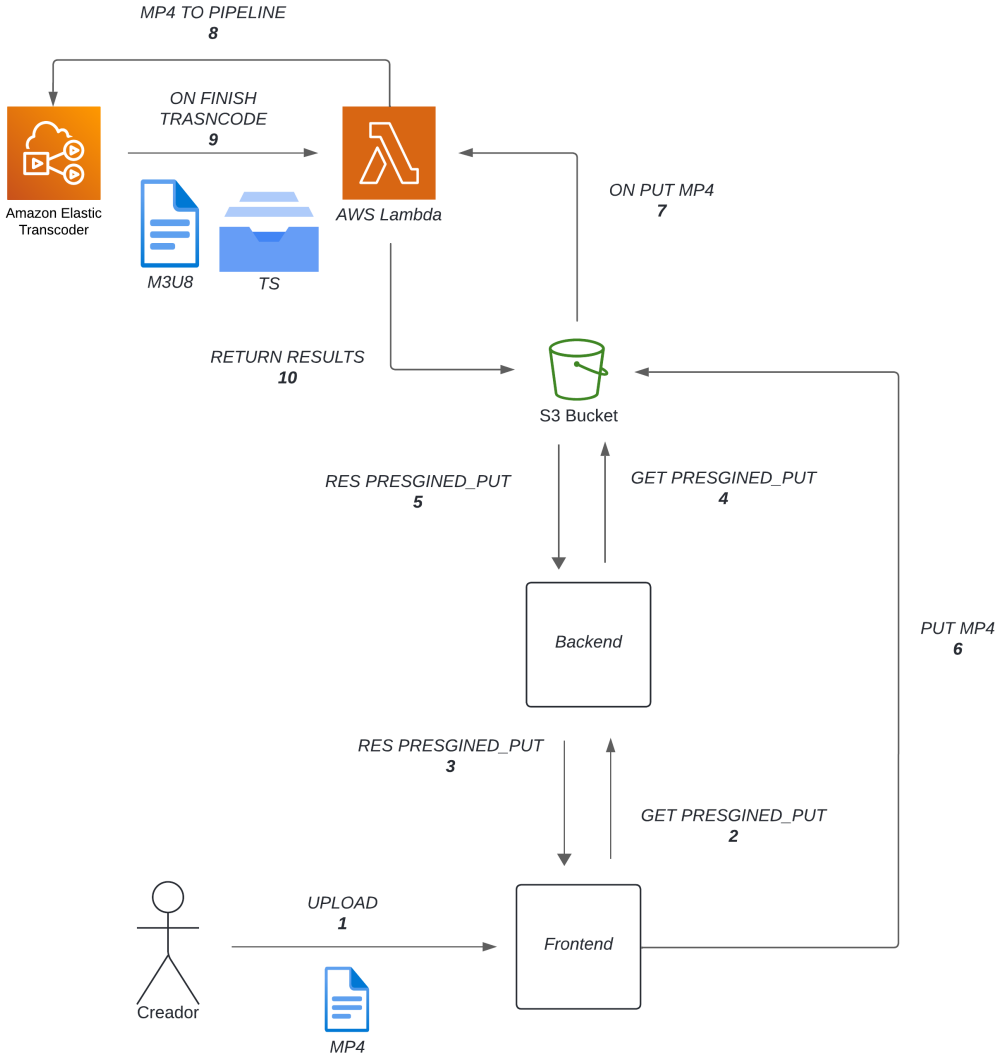

Figura 14: Esquema de pujada de vídeos

Aquí teniu un resum de què fa cada part per tenir un coneixement global de com funciona l'arquitectura:

- **1 UPLOAD**: El client carrega un vídeo a través del frontend.
- **2 BACKEND Get Presigned PUT**: El client demana una URL preautoritzada per fer un PUT del vídeo.
- **3 S3 GET PRESIGNED PUT**: El backend demana a S3 una URL preautoritzada per pujar el vídeo.
- **4 S3 RES PRESIGNED PUT**: S3 retorna la URL preautoritzada al backend.
- **5 BACKEND RES PRESIGNED PUT**: El backend envia la URL preautoritzada al client.

- **6 PUT MP4 S3**: El client utilitza la URL preautoritzada per pujar el vídeo directament a S3.
- **7 LAMBDA ON PUT A S3**: La pujada del vídeo a S3 desencadena una funció Lambda.
- **8 MP4 TO AWS ELASTIC TRANSCODER PIPELINE**: La funció Lambda envia el vídeo a AWS Elastic Transcoder per a la transcodificació.
- **9 ON FINISH TRANSCODE**: Un cop finalitzada la transcodificació, Elastic Transcoder envia el vídeo transcodificat a S3.
- **10 SEND RESULT S3**: Els vídeos transcodificats es guarden al bucket S3 original.

7.2.11 Consum de Vídeo amb HLS en React

Implementem la lògica per a la càrrega i reproducció dels vídeos HLS en la nostra aplicació web, utilitzant el hook `useEffect` per gestionar canvis dinàmics en l'URL del vídeo. Com hem vist a l'apartat anterior com es gestiona les URL presingades també o farem amb la gestió del consum dels vídeos, quan un usuari vol consumir un vídeo cada arxiu .ts i el mateix arxiu .ts es demanarà una signatura a AWS per poder consumir-ho. Aquí tenim el codi en el frontend:

```
useEffect(() => {
    if (videoRef.current && videoUrl && Hls.isSupported()) {
        const hls = new Hls({
            xhrSetup: (xhr, url) => {
                if (url.endsWith('.ts')) {
                    customAxios.get('/sign-url', {
                        params: { url }
                    }).then(response => {
                        xhr.open('GET', response.data.signedUrl, true);
                        xhr.send();
                    }).catch(error => {
                        *** GESTIÓ DEL ERROR ***
                    });
                } else {
                    xhr.open('GET', url, true);
                    xhr.send();
                }
            }
        });
        hls.loadSource(videoUrl);
        hls.attachMedia(videoRef.current);
    } else if (videoRef.current?.canPlayType('application/vnd.apple.mpegurl')) {
        videoRef.current.src = videoUrl;
    }
}, [videoUrl]);
```

Ara veurem l'estructura del consum de vídeos per entendre com funciona. Encara que pugui semblar un esquema complicat a primera vista, el desglossarem pas a pas per explicar cada fase del procés:

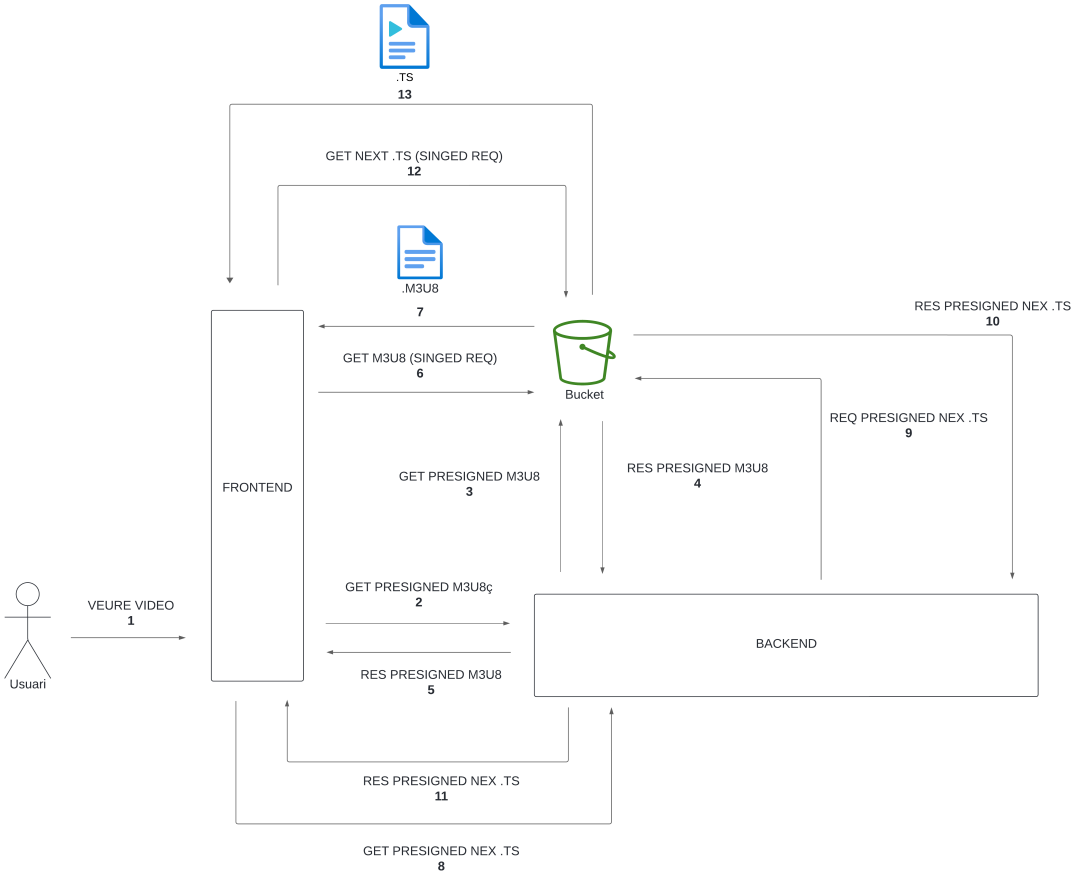

Figura 15: Esquema del consum de vídeos amb HLS

- `1.Usuari:` L'usuari vol veure un vídeo a la plataforma.
- `2.Frontend:` El client (frontend) demana al backend una URL preautoritzada per al fitxer M3U8, que és la llista de reproducció del vídeo.
- `3.Backend:` El backend genera una URL preautoritzada per al fitxer M3U8 utilitzant el client S3 de Boto3.

  ```
  video_data['video_url'] = get_authenticated_url_video(video.video_url)
  ```

- `4.S3:` El bucket de S3 respon amb la URL preautoritzada per al fitxer M3U8.

  ```
  def get_authenticated_url_video(url):
  try:
      url = url.split('/')[-3] + '/' + url.split('/')[-2] + '/'
      + url.split('/')
  ```

```
        [-1]
        response = client.generate_presigned_url('get_object',
                                        Params={
                                            'Bucket': bucket_name,
                                            'Key': url,
                                            'ResponseContentType':
                                            'application/vnd.apple.mpegurl',
                                            'ResponseContentDisposition':
                                            'inline'
                                            },
                                        ExpiresIn=3)
        return response
```

- `5.Backend`: El backend retorna la URL preautoritzada al client (frontend).

```
return Response({
        'video': video_data,
        'questions': PreguntasVideoSerializer(preguntas, many=True).data
    })
```

- `6.Frontend`: El client utilitza aquesta URL per obtenir el fitxer M3U8 des de S3.

```
customAxios.get(`creator-get-video/${video_id}`).then((response) => {
            const videoData = response.data.video;
            setVideoUrl(videoData.video_url || '');
```

- `7.S3`: El bucket S3 envia el fitxer M3U8 al client.
- `8.Frontend`: El client comença la reproducció del vídeo HLS utilitzant el fitxer M3U8.
- `9.Frontend`: Durant la reproducció, el client demana segments de vídeo .ts en ordre.
- `10.Backend`: El client demana una URL preautoritzada per al següent segment de vídeo .ts.
- `11.Backend`: El backend genera una URL preautoritzada per al segment .ts i la retorna al client.
- `12.Frontend`: El client utilitza aquesta URL per obtenir el segment de vídeo .ts des de S3.
- `13.S3`: El bucket S3 envia el segment de vídeo .ts al client.
- `14.Frontend`: El client continua la reproducció del vídeo amb el nou segment .ts.

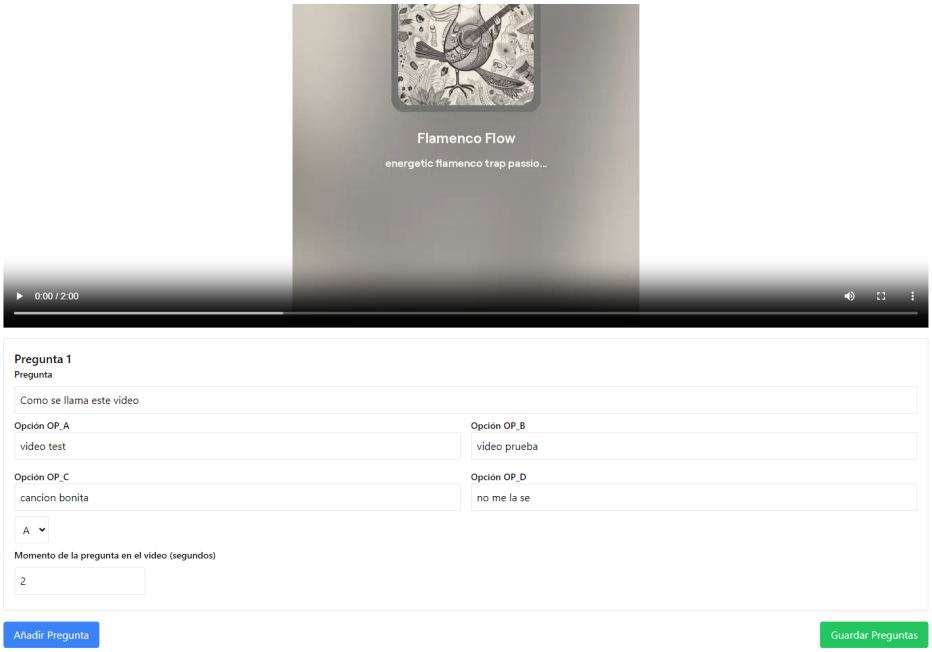

Figura 16: Editor de preguntes del vídeo d'un Creador.

7.2.12 Preguntes als Vídeos

En aquesta secció es detalla la implementació de la funcionalitat de preguntes als vídeos a la nostra plataforma. Aquesta funcionalitat permet afegir preguntes que apareixen durant la reproducció del vídeo, fent que sigui una experiència interactiva per als usuaris. El creador indica en crear un curs si vol afegir preguntes en quin segon exacte del vídeo han d'aparèixer. Això ho fa servir l'entitat Preguntes vídeos descrita en l'anàlisi de l'aplicació web.

Comprovació de Preguntes Durant la reproducció del vídeo, es comprova periòdicament si hi ha alguna pregunta associada al temps actual del vídeo. Si es troba una pregunta, el vídeo es pausa i es mostra la pregunta a l'usuari, sortint de pantalla completa, no hem aconseguit afegir la pregunta en pantalla completa i aquesta és la millor solució trobada.

```
const checkForQuestions = () => {
    const currentTime = Math.floor(videoRef.current.currentTime);
    const question = questions.find(q => q.time === currentTime &&
        !questionsAnswered.has(q.id));
    if (question) {
        setShowQuestion(true);
        setCurrentQuestion(question);
        videoRef.current.pause();
    if (document.fullscreenElement) {
    document.exitFullscreen();
    }
```

```
    }
};

useEffect(() => {
    const video = videoRef.current;
    video.addEventListener('timeupdate', checkForQuestions);
    return () => video.removeEventListener('timeupdate',
                                checkForQuestions);
}, [questions, questionsAnswered]);
```

Gestió de Respostes Quan l'usuari respon una pregunta, es guarda la resposta i es reprèn la reproducció del vídeo:

```
const handleAnswer = () => {
    const currentTime = videoRef.current.currentTime;
    setQuestionsAnswered(new Set([...questionsAnswered,
                    currentQuestion.id]));
    setShowQuestion(false);
    setTimeout(() => {
        videoRef.current.currentTime = currentTime;
        videoRef.current.play();
    }, 0);
};
```

Component `QuestionVideo` Aquest component mostra la pregunta i les opcions de resposta. Quan l'usuari selecciona una resposta, es comprova si és correcta i es notifica l'usuari:

```
const handleSubmit = (option, options) => {
    if (option.toUpperCase() === correctAnswer) {
         ** GESTIÓ RESPOSTA INCORRECTA **
    } else {
        ** GESTIÓ RESPOSTA CORRECTA **
    }
    onAnswer();
};
```

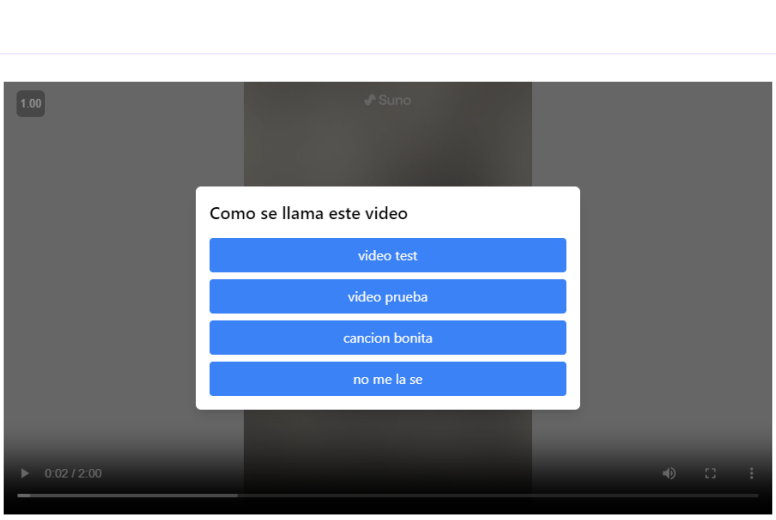

Figura 17: Captura que mostra una pregunta en el segon 2 del consum d'un vídeo.

7.2.13 Gestió de l'Estat de la Pujada del Vídeo

Per gestionar l'estat de la pujada i processament del vídeo, utilitzem un fitxer de text anomenat `upload_status.txt` que indica si un vídeo s'està processant o no. Aquest fitxer es desa al bucket S3 i permet a la nostra aplicació indicar a l'usuari si un vídeo s'està transcodificant o no. No indica si hi ha hagut cap error, aquesta és la solució que s'ha trobat per a la fase de desenvolupament.

- **1. Creació del Fitxer de Estatus:** Quan un usuari inicia la pujada d'un vídeo, es crea un fitxer de text amb el nom `upload_status.txt` al bucket S3. Aquest fitxer indica que el vídeo s'està processant.

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def get_video_status_urls(request, video_id):
    object_name = f'videos/{video_id}/upload_status.txt'
    status_put = generate_presigned_status_put(object_name, 'text/plain')
    status_get = generate_presigned_status_get(object_name)

    if not status_put or not status_get:
        return Response({'error': 'No se pudo generar las URLs de status'}
        , status=500)

    return Response({
        'status_put': status_put,
        'status_get': status_get
    })
```

- **2. Eliminació del Fitxer de Estatus:** Una vegada que la transcodificació del vídeo es completa, el fitxer `upload_status.txt` s'elimina per indicar que el procés ha finalitzat. Aquesta operació es gestiona dins de la nostra funció Lambda.

```
import boto3
import time

def lambda_handler(event, context):
    [...]
    if key.endswith('upload.mp4'):
        directory = key.rsplit('/', 1)[0] + '/'
        status_key = directory + 'upload_status.txt'
        try:
            [...]
            Transcodificació i substitució d' arxius a S3
            [...]
        finally:
            try:
                s3_client.delete_objects(
                    Bucket=bucket_name,
                    Delete={'Objects': [{'Key': status_key}]}
                )
                print("Archivo de estado eliminado.")
[...]
```

- **3. Verificació de l'Estat del Fitxer:** El frontend pot consultar l'existència del fitxer `upload_status.txt` per determinar si el vídeo encara està en procés o si la transcodificació ha estat completada. Tenim dues funcions una mica diferents: la comprovació inicial d'estatus i la de comprovació de l'estatus en la pujada de vídeo, l'inicial el que fa és que en carregar el component l'executa per veure si aquest vídeo s'està processant, si no hi és no fa res i fem fetch del vídeo.

```
const checkInitialTranscodingStatus = async () => {
    try {
        const response = await customAxios
        .get(`get-video-status-urls/${video_id}`);
        const { status_get } = response.data;

        try {
            await axios.get(status_get);
            setIsTranscoding(true);
            startTranscodingStatusCheck();
        } catch (error) {
            if (error.response && error.response.status === 404) {
```

```
                setIsTranscoding(false);
                fetchVideoData();
            }
        }
    } catch (error) {
        console.error('Error fetching status URLs:', error);
    }
};

const checkTranscodingStatus = async () => {
    try {
        const response = await customAxios
        .get(`get-video-status-urls/${video_id}`);
        const { status_get } = response.data;

        try {
            await axios.get(status_get);
        } catch (error) {
            if (error.response && error.response.status === 404) {
                setIsTranscoding(false);
                clearInterval(intervalId);
                setIntervalId(null);
                fetchVideoData();
            }
        }
    } catch (error) {
        console.error('Error fetching status URLs:', error);
    }
};
```

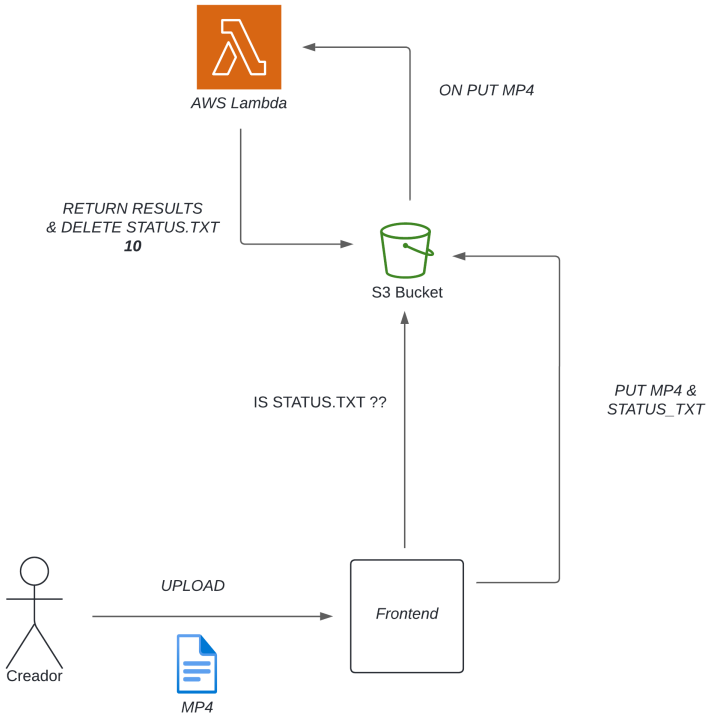

Figura 18: Gestió de l'estatus de la pujada d'un arxiu mp4 a la plataforma per part d'un creador

Amb aquest procés permetem als creadors veure l'estat actual de la pujada dels seus vídeos. La funció mantindrà l'estat a "processing"fins que el vídeo està completament processat i l'arxiu `upload_status.txt` eliminat.

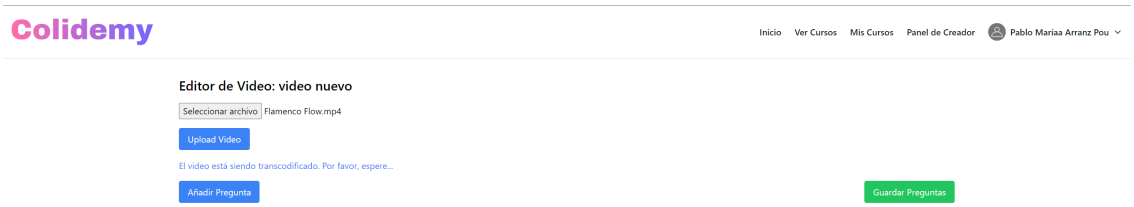

Figura 19: Captura de com el Client informa al Creador que el vídeo s'està transcodificant.

7.3 Consum de Documents

En aquesta secció es descriu el procés de consum de documents a la nostra plataforma. Aquesta funcionalitat permet als usuaris visualitzar documents PDF amb informació específica de l'usuari incrustada en el document, com s'ha definit als requisits de l'aplicació. A continuació es detalla el funcionament del frontend i del backend que permeten aquesta funcionalitat.

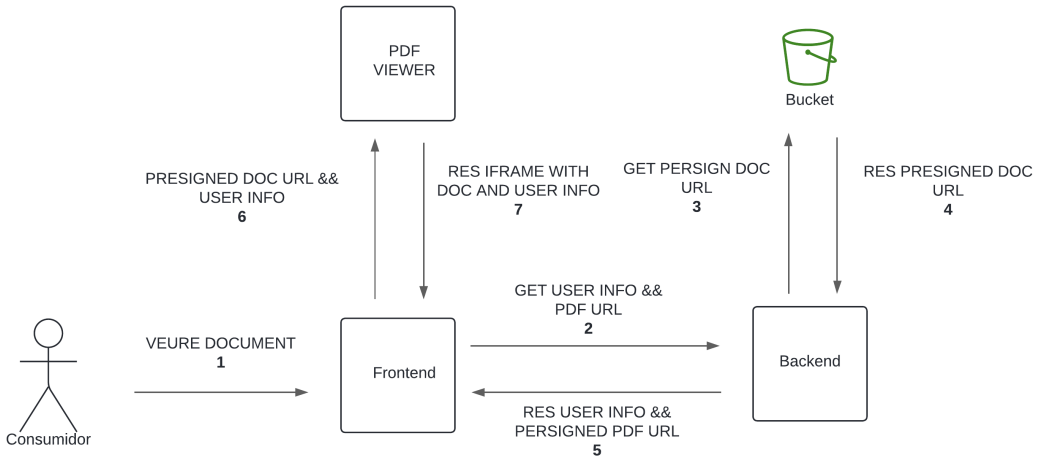

Figura 20: Esquema de consum de documents

7.3.1 Funcionalitat en el Backend

El backend és responsable de proporcionar la informació del document sol·licitat i generar una URL preautoritzada per accedir al PDF. Com hem fet en els vídeos fent servir URLs preautoritzades, utilitzem el mateix procediment. Fem servir aquests endpoints que es poden consultar als annexos del document, però bàsicament fem el mateix que amb els vídeos: agafem l'URL i l'autoritzem amb claus d'AWS. Els endpoints són `get_document(request, document_id)` i `get_authenticated_url_pdf(url)`.

7.3.2 Funcionalitat en el Frontend

El frontend està dissenyat per carregar i mostrar els documents PDF amb informació incrustada de l'usuari. Consisteix en dos components principals: `Document` i `PdfViewer`.

7.3.3 Component Document

El component `Document` és el principal encarregat de gestionar la visualització dels documents PDF. Aquest component utilitza `useEffect` per carregar la informació del document quan es carrega la pàgina. Es fa una crida a l'endpoint `get-course-document` per obtenir l'URL del document i les dades de l'usuari. El component `Document` mostra el document PDF utilitzant el component `PdfViewer` quan les dades de l'usuari i l'URL del document estan disponibles.

7.3.4 Component PdfViewer

El component `PdfViewer` és el responsable de carregar i modificar el PDF per incloure la informació de l'usuari. Aquest component utilitza `PDFDocument` de la llibreria pdf-lib [11] per carregar el PDF i modificar-lo afegint el DNI, el telèfon i l'e-mail de l'usuari en 6 posicions aleatòries de cada pàgina. El component `PdfViewer` mostra el PDF modificat utilitzant un element `iframe` per incrustar el document en la pàgina de la vista del document.

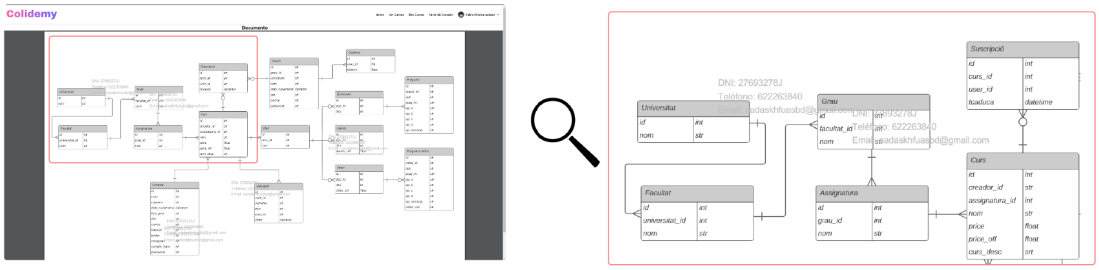

Figura 21: Marca d'aigua a un document amb la informació de l'usuari que l'està consumint.

7.4 Qüestionaris

La gestió de qüestionaris a la plataforma es realitza a través de diverses funcionalitats backend que permeten la creació, modificació i visualització dels qüestionaris i les seves preguntes. A continuació es descriuen els principals punts de la implementació dels qüestionaris:

7.4.1 Endpoints

Els endpoints per a la gestió de qüestionaris es defineixen a `cuestionario_editor` i proporcionen les següents funcionalitats:

Obtenir Preguntes del Qüestionari del Creador Aquest endpoint permet als creadors obtenir les preguntes associades a un qüestionari específic.

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def creatorGetPreguntasCuestionario(request, cuestionario_id):
    [...]** COMPROVACIÓ QUE EL CREADOR SIGUI EL PROPIETARI DEL CUESTIONARI **

    preguntas = PreguntaCuestionario.objects.filter(cuestionario=cuestionario)
    preguntas_serializer = PreguntaCuestionarioSerializer(preguntas, many=True)
    cuestionario_serializer = CuestionarioSerializer(cuestionario)

    return Response({
        'cuestionario': cuestionario_serializer.data,
```

```
        'preguntas': preguntas_serializer.data
    })
```

Guardar Qüestionari del Creador Aquest endpoint permet als creadors desar les preguntes d'un qüestionari. Si la pregunta ja existeix, s'actualitza; si no, es crea una nova entrada.

```
@api_view(['POST'])
@permission_classes([IsAuthenticated])
def creatorSaveCuestionario(request, cuestionario_id):
    [...]** COMPROVACIÓ DE QUE EL CREADOR SIGUI EL PROPIETARI DEL QÜESTIONARI **

    data = request.data.get('preguntas', [])

    for pregunta_data in data:
        pregunta_id = pregunta_data.get('id')
        defaults = {
            'texto': pregunta_data.get('texto'),
            'op_a': pregunta_data.get('op_a'),
            'op_b': pregunta_data.get('op_b'),
            'op_c': pregunta_data.get('op_c'),
            'op_d': pregunta_data.get('op_d'),
            'op_correcta': pregunta_data.get('op_correcta'),
            'cuestionario': cuestionario
        }
        if pregunta_id:
            pregunta, created = PreguntaCuestionario.objects.update_or_create(
                id=pregunta_id,
                defaults=defaults
            )
        else:
            pregunta = PreguntaCuestionario.objects.create(**defaults)

    return Response({'message': 'Cuestionario actualizado correctamente'})
```

Obtenir Qüestionari Aquest endpoint permet als consumidors obtenir un qüestionari complet amb les seves preguntes, sempre que tinguin els permisos adequats.

```
@api_view(['GET'])
@permission_classes([IsAuthenticated])
def get_cuestionario(request, quiz_id):
    [...]** COMPROVACIO DE QUE EL QÜESTIONARI SIGUI GRATUIT O ESTIGUI PAGAT **
    preguntas_data = []

    for pregunta in preguntas:
        opciones = [pregunta.op_a, pregunta.op_b,
```

```
        pregunta.op_c, pregunta.op_d]
        preguntas_data.append({
            'pregunta': pregunta.texto,
            'opciones': opciones,
            'respuesta_correcta': pregunta.op_correcta
        })

    return Response({
        'titulo': cuestionario.titulo,
        'preguntas': preguntas_data
    })
```

7.4.2 Frontend

El component `Quiz` controla la visualització i la interacció amb els qüestionaris. El codi complet del component es pot veure als annexos.

Funcions Principals

- `fetchQuiz`: Aquesta funció es crida quan es carrega el component i fa una petició a l'endpoint `'get-course-quiz/<int:quiz_id>'` per obtenir el qüestionari i les seves preguntes. Les dades es guarden en l'estat del component.

```
useEffect(() => {
    const fetchQuiz = async () => {
        try {
            setIsLoading(true);
            const response = await customAxios
                .get(`/get-course-quiz/${quizID}`);
            setTestName(response.data.titulo);
            setQuestions(response.data.preguntas);
        } catch (error) {
            console.error("Error al hacer fetch del quiz", error)
        } finally {
            setIsLoading(false);
        }
    };
    fetchQuiz();
}, [quizID]);
```

- `handleChangeQuizMode`: Aquesta funció permet canviar entre els modes de qüestionari 'normal' i 'examen'. Aquesta funcionalitat no estava planejada, però la vaig fer per enriquir els qüestionaris, bàsicament és un toggle que el que fa és donar retroalimentació en fer el qüestionari després de cada pregunta o només al final com si fos un examen.

```
    const handleChangeQuizMode = () => {
```

```
        setQuizMode(prevMode => prevMode === 'normal' ?
            'exam' : 'normal');
        setProgressPercentage(0);
        setShowResults(false);
        setCurrentQuestionIndex(0);
        setFeedback({ show: false,
        correct: false,
        correctAnswer: '' }
        );
        setCorrectAnswersCount(0);
    };
```

- `handleAnswer`: Aquesta funció es crida quan l'usuari selecciona una resposta. Comprova si la resposta és correcta i actualitza l'estat del component en conseqüència.
- `handleNextQuestion`: Aquesta funció passa a la següent pregunta del qüestionari. Si l'usuari està en el mode "normal", mostra la següent pregunta després de proporcionar feedback sobre la resposta seleccionada.

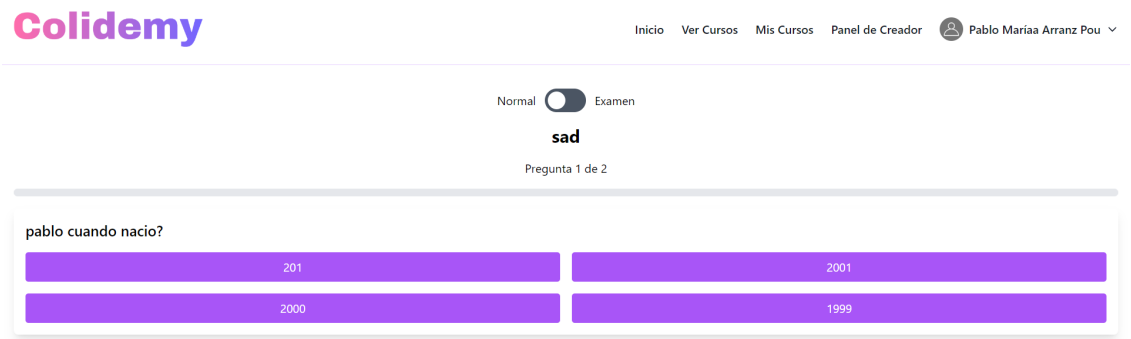

Figura 22: Vista del consumidor d' un curs

7.5 Creació de cursos

Ja sabent com funciona cada part del curs, ara definirem com un creador pot crear i estructurar un curs basant-se en qüestionaris, vídeos i documents, establir preus i posar en oferta el seu curs personalitzant-lo.

7.5.1 Passos per crear un curs

Per crear un curs, un usuari ha de seguir els següents passos:

1. **Donar-se d'alta com a Creador:** L'usuari que vol crear un curs s'ha de donar d'alta com a creador. Això ho aconseguim fent servir l'endpoint `/register-creator`, que crea la instància d'Usuari Creador associada al id del Usuari.

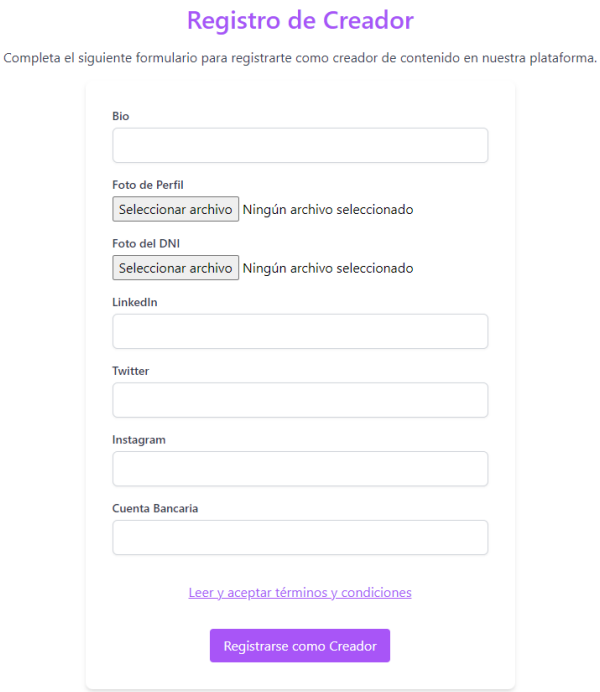

Figura 23: Registre com a Creador a la plataforma

2. **Accés a la secció de creació de cursos:** L'usuari creador ha d'iniciar sessió a la plataforma i accedir a la secció de creació de cursos. Per obtenir tota aquesta informació fem ús de l'endpoint `/get-courses-creator`. També pot personalitzar el seu perfil de creador fent ús de l'endpoint `/update-creator-info`. A més, veu unes petites estadístiques que mostren les últimes subscripcions i els ingressos generats en l'última setmana. Aquestes dades setmanals venen de l'endpoint `/get-last-week-stats`.

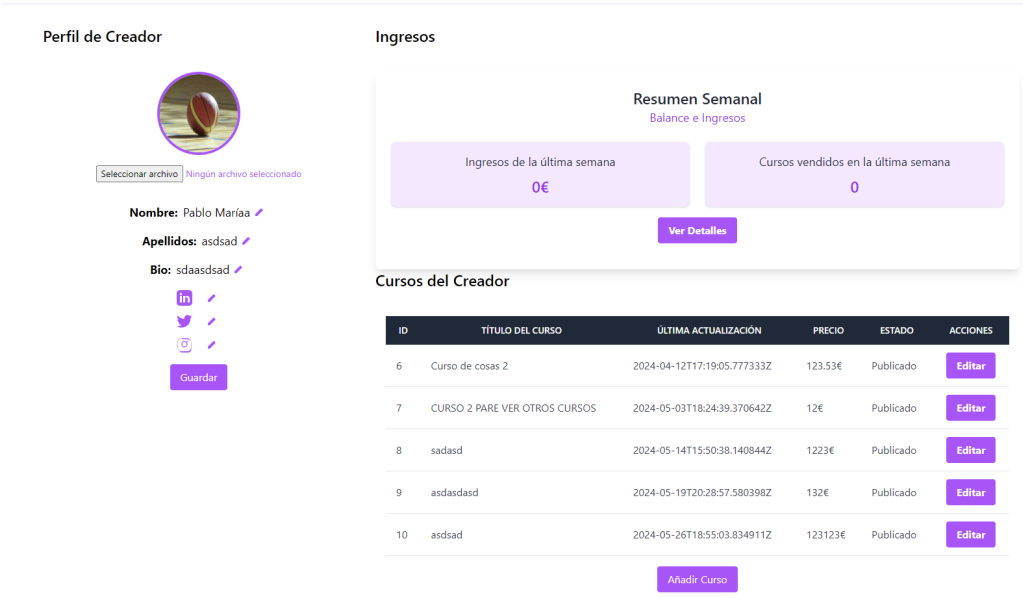

Figura 24: Panell del Creador

3. **Definició del curs:** Es proporcionarà informació bàsica sobre el curs, com ara el títol, la descripció, la universitat, la facultat i el grau associat. Quan el formulari estigui complet, crearem el curs fent ús de l'endpoint `/create-course`, comprovant que l'usuari que fa la petició sigui un creador.

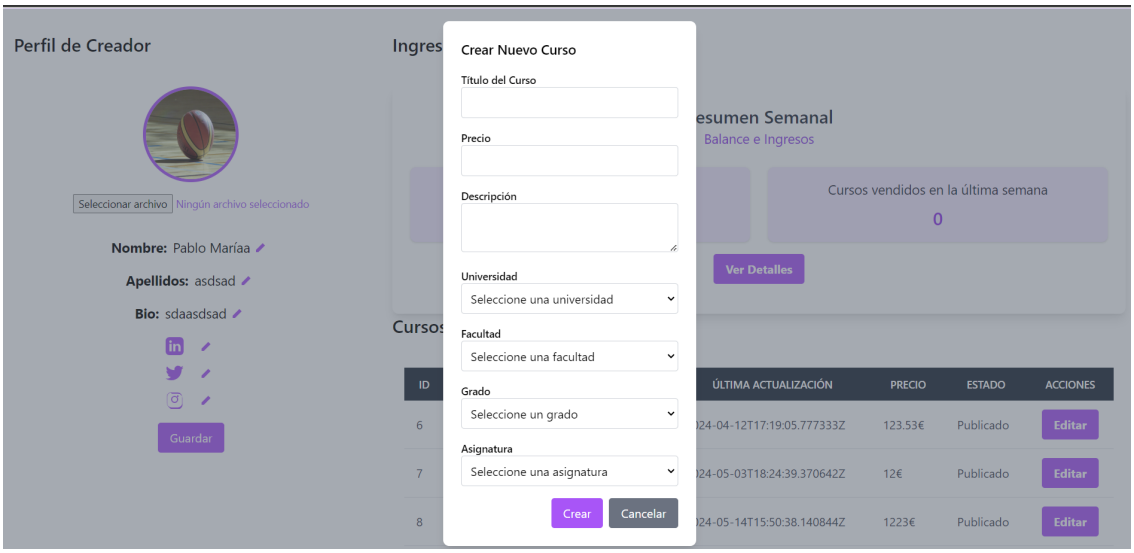

Figura 25: Formulari per crear un nou curs

4. **Estructuració del contingut:** El creador podrà afegir lliçons, que es compondran de diversos tipus de contingut que hem desenvolupat als apartats anteriors (Vídeos, Qüestionaris i Documents). Si volem crear nous continguts a una lliçó, fem ús de l'endpoint `/creator-get-course/{curso_id}`, ja que

en el frontend anem guardant les dates de les noves lliçons i el seu ordre. També hi ha un checkbox per indicar si el creador vol deixar algun element de forma gratuïta.

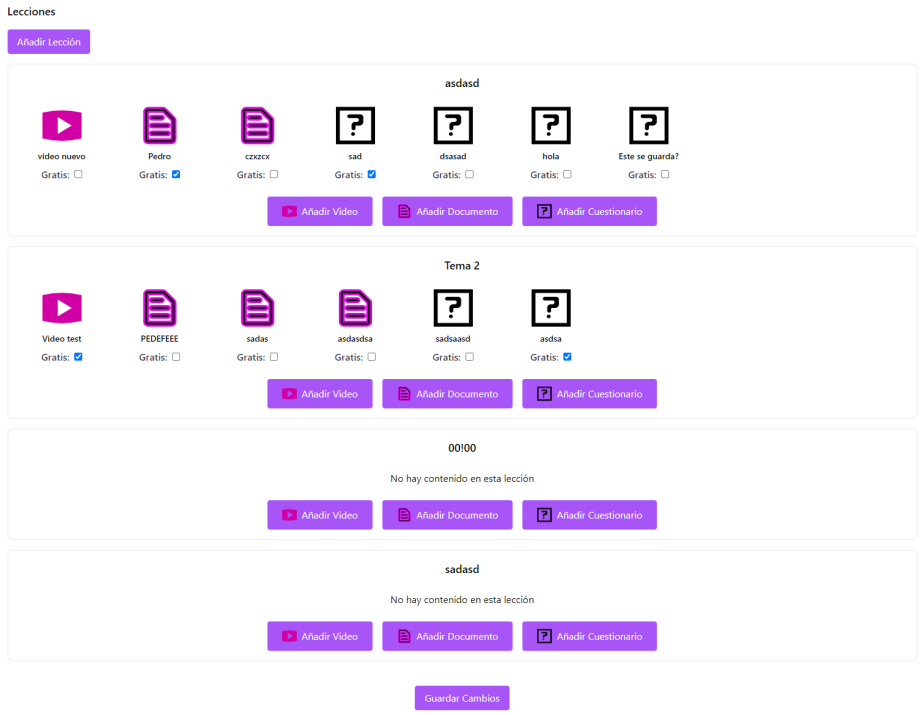

Figura 26: Lloc on el creador estructura el contingut en seccions

5. **Personalització i publicació del curs:** El creador també pot editar la descripció, el nom del curs, el preu i pot crear una oferta del curs. Aquesta funcionalitat fa servir el mateix endpoint que al guardar les lliçons i el seu contingut del pas anterior. Un cop completats tots els passos anteriors, el creador podrà publicar el curs, que passarà a estar disponible a la biblioteca de cursos de la plataforma utilitzant l'endpoint `/creator-publish-course/{curso_id}`.

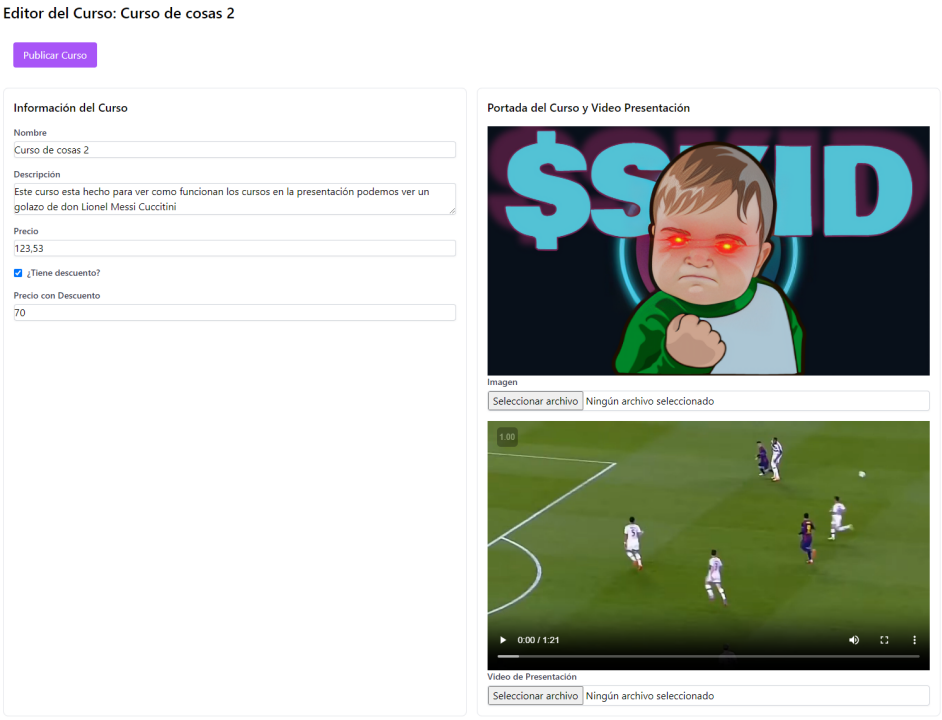

Figura 27: Personalització del curs

8 Testing

Per garantir la qualitat i fiabilitat de la plataforma desenvolupada, s'ha implementat un procés de testing utilitzant les eines proporcionades per *Django Rest Framework* i *coverage*. Aquestes eines permeten assegurar que el codi funcioni correctament i que totes les funcionalitats estiguin degudament cobertes per tests.

8.1 Eines Utilitzades

Django Rest Framework ofereix una classe de tests que facilita la creació de tests per a APIs, permetent assegurar que les rutes, les vistes i les funcionalitats de l'API es comportin com s'espera. Aquesta classe proporciona mètodes per realitzar sol·licituds HTTP i verificar les respostes.

Coverage és una eina que mesura la quantitat de codi executat per les proves. Aquests informes ajuden a identificar parts del codi que no estan cobertes per cap test, permetent millorar la qualitat del software.

8.2 Configuració amb Django Rest Framework

Per configurar els tests amb Django Rest Framework, es crea una classe de tests que hereta de `APITestCase`. Aquesta classe proporciona mètodes per simular sol·licituds HTTP i verificar les respostes.

En el cas de la nostra plataforma hem implementat diferents `APITestCase`'s per provar diferents casuístiques i accions per separat. Són els següents:

- `test_centros`: Aquests tests estan destinats a gestionar i comprovar les funcionalitats relacionades amb la gestió d'universitats, facultats, graus i assignatures, la seva creació i el seu consum.
- `test_consumidor`: Aquest conjunt de tests verifica les funcionalitats destinades als usuaris consumidors, com ara el consum de cursos, lliçons, vídeos, qüestionaris i documents.
- `test_creador`: Aquests tests estan enfocats a assegurar que les funcionalitats destinades als creadors de contingut funcionin correctament com ara la creació de cursos, de qüestionaris o de documents.
- `test_otros`: Aquests tests cobreixen diverses funcionalitats addicionals com les subscripcions, les valoracions i el càlcul de mitjanes de valoració.

8.3 Coverage

8.3.1 Configuració de Coverage

Per mesurar la cobertura dels tests, s'utilitza *coverage*. Donat que hi ha molts arxius de Django que ja han estat testejats que no són codi propi i arxius com les

migracions o el pycache que tampoc ha de ser testejat, no volem que siguin inclosos en els tests. Per a això, creem un arxiu de configuració anomenat `.coveragerc`. El nostre `.coveragerc` és el següent:

```
[run]
source = .

[report]
omit =
    */migrations/*
    */tests/*
    manage.py
    colinemy\wsgi.py
    colinemy\asgi.py
    colinemy\settings.py
    colinemy\urls.py
    colinemy\__init__.py
    */__init__.py
    */admin.py
    */apps.py
    */models.py
    */serializers.py
    */urls.py
```

8.3.2 Execució de Coverage

Per mesurar la cobertura dels tests, s'utilitza la comanda `coverage` amb l'arxiu `.coveragerc` que hem creat anteriorment. Per executar els tests amb *coverage*, utilitzem la següent comanda:

```
coverage run --rcfile=.coveragerc manage.py test
```

Per veure com de cobert està el nostre codi, executem:

```
coverage report --rcfile=.coveragerc
```

Aquest comandament genera un informe que proporciona detalls sobre el percentatge de línies de codi cobertes pels tests per a cada fitxer. El nostre d'informe de cobertura és el següent:

```
Name                          Stmts   Miss  Cover
-----------------------------------------------
api\aws.py                       49     19    61%
api\course_editor.py             71     14    80%
api\course_get.py               166     33    80%
api\course_rating.py             53     11    79%
api\creator_endpoints.py        108     23    79%
```

```
api\creator_stats.py             21      4    81%
api\cuestionario_editor.py       43     11    74%
api\document_editor.py           50     12    76%
api\home_gets.py                 34      0   100%
api\video_editor.py              94     32    66%
api\views.py                    226     28    88%
-------------------------------------------------
TOTAL                           915    187    80%
```

8.3.3 Anàlisi de Resultats

L'informe de cobertura reporta quines parts del codi estan cobertes pels tests i quines no. A continuació, s'explica la informació proporcionada per l'informe:

- **Name**: El nom del fitxer de codi font.
- **Stmts**: El nombre total d'instruccions en el fitxer.
- **Miss**: El nombre d'instruccions que no han estat executades durant els tests.
- **Cover**: El percentatge de cobertura que és el percentatge d'instruccions que han estat executades durant els tests.

9 Conclusions

Durant el desenvolupament del projecte, s'han aconseguit assolir tots els objectius principals i la majoria dels secundaris plantejats inicialment. S'han implementat les funcionalitats de registre i accés, gestió i visualització de cursos, subscripció, interacció dinàmica en vídeos, consum de qüestionaris i documents, i gestió de pagaments (aquesta està simulada). A més, la integració amb AWS ha sigut un gran aprenentatge personal i ha donat solució al problema plantejat.

No obstant això, cal destacar que no s'han pogut assolir algunes històries d'usuari secundàries, especialment aquelles relacionades amb l'extracció de diners, coneixement dels diners generats pel creador, i el seguiment del progrés del curs per part de l'usuari consumidor. Aquestes funcionalitats, tot i ser importants per completar l'experiència de l'usuari creador, no s'han pogut implementar dins del temps disponible per a aquest projecte.

De cara al futur, es desitja continuar desenvolupant el projecte amb diverses millores. Una d'elles és la migració de més contingut a AWS, com ara fotografies i vídeos de presentació dels cursos, que actualment es gestionen a la carpeta `media` del backend. Aquesta migració permetrà una gestió més eficient i escalable dels recursos multimèdia, assolint encara més l'objectiu inicial plantejat. A més es té la intenció d'incloure eines externes per gestionar els pagaments perquè ara mateix aquesta funcionalitat està simulada.

A més, el frontend de la plataforma és encara molt simple i necessita una millora significativa per fer-lo més atractiu i accessible. Es vol treballar en una interfície d'usuari més moderna i intuïtiva, que faciliti l'experiència de navegació i ús per part dels estudiants i creadors de contingut.

També, es considera ampliar la cobertura dels tests per assegurar la qualitat del codi. Es proposa augmentar la cobertura de codi amb tests unitaris addicionals. Es considera correcte el 80% assolit, però s'espera poder pujar aquest percentatge.

Abans d'acabar, s'és conscient de l'existència d'alguns bugs que han estat detectats i documentats com a issues a GitHub dins del projecte. Aquests bugs seran abordats en futurs desenvolupaments per millorar l'experiència de l'usuari.

Finalment, es desitja explorar totes les possibilitats que ofereix la tecnologia HLS per a la transmissió de vídeo com l'adaptació de la qualitat del vídeo en temps real segons l'ample de banda disponible, la inclusió de múltiples pistes d'àudio i subtítols.

10 Annexos

10.1 Planificació

10.1.1 Planificació i Anàlisi

- **Març: S2-S3: Planificació**
  - Planificació inicial del projecte. Això inclou definir l'abast del projecte i identificar les necessitats dels usuaris. Es preveu establir els objectius clau i crear un pla de treball detallat.
- **Març: S3-S4: Requisits**
  - Recopilar i documentar els requisits funcionals i no funcionals necessaris per al desenvolupament de la plataforma. Aquesta fase inclourà l'anàlisi de plataformes similars per assegurar que es compleixin les necessitats dels usuaris.
- **Abril: S1: E-R (Entitat-Relació)**
  - Durant tot el mes d'abril, es planeja dissenyar el model de dades de la plataforma utilitzant diagrames d'entitat-relació. Aquesta tasca inclourà definir les entitats, les relacions i les restriccions, assegurant la integritat i l'eficiència de la base de dades.

10.1.2 Funcionalitats

Abril

- **Abril: S1: Autenticació (Auth)**
  - Durant la primera setmana d'abril, es planeja implementar el sistema d'autenticació i autorització per garantir que només els usuaris registrats puguin accedir a la plataforma.
- **Abril: S2-S3: Crear Curs**
  - Desenvolupar la funcionalitat per a la creació de cursos, permetent als creadors carregar materials i estructurar el contingut del curs.
- **Abril: S4: Documents (Docs)**
  - Implementar la funcionalitat per carregar, gestionar i consumir documents associats als cursos.

Maig

- **Març S4- Maig S1: Vídeos**
  - Desenvolupar la funcionalitat per a la càrrega i gestió de vídeos, incloent la integració amb serveis de transcodificació.

- **Maig: S2: Qüestionaris (Q)**
  - Implementar la funcionalitat per a la creació, gestió i consum de qüestionaris dins dels cursos.
- **Maig: S3: Vista de Curs**
  - A partir de la tercera setmana de maig, es planeja desenvolupar la interfície d'usuari per a la visualització dels cursos, permetent als estudiants accedir als continguts de manera organitzada.
- **Maig: S4: Publicació i Gestió (P&G)**
  - Durant la quarta setmana de maig, es planeja implementar les funcionalitats per a la publicació de cursos i la seva gestió posterior, incloent el seguiment de les inscripcions i els ingressos.

Juny

- **Juny: S1-S3: Valoració**
  - Durant les tres primeres setmanes de juny, es planeja desenvolupar la funcionalitat perquè els estudiants puguin valorar els cursos i proporcionar feedback als creadors.

10.1.3 Documentació

Març - Maig

- **Març - Maig: S3 Març - S4 Maig: Memòria**
  - Durant tot el període de març a juny, es planeja documentar tot el procés de desenvolupament, incloent l'anàlisi de requisits, el disseny del sistema, la implementació i les proves.

Juny

- **Juny: S1-S4: Revisió de la Memòria**
  - Durant tot el mes de juny, es planeja revisar i refinar la documentació final, assegurant que tots els aspectes del projecte estiguin clarament explicats i documentats.

10.2 Models DB

10.2.1 Estructura Acadèmica

- **Universitat** (`Universidad`)**:**
  - `nombre`: `String` - Guarda el nom de la universitat.
- **Facultat** (`Facultad`)**:**
  - `nombre`: `String` - Guarda el nom de la facultat.
  - `universidad`: `ForeignKey(Universidad)` - Estableix una relació foreign key amb `Universidad`, indicant que cada facultat pertany a una sola universitat. La supressió en cascada implica que si s'elimina una universitat, totes les seves facultats associades també s'eliminaran.
- **Grau** (`Grado`)**:**
  - `nombre`: `String` - Guarda el nom del grau acadèmic.
  - `facultad`: `ForeignKey(Facultad)` - Connecta cada grau amb una única facultat mitjançant una foreign key, amb eliminació en cascada que elimina tots els graus associats si una facultat és eliminada.
- **Assignatura** (`Asignatura`)**:**
  - `nombre`: `String` - Guarda el nom de l'assignatura.
  - `grado`: `ForeignKey(Grado)` - Relaciona cada assignatura a un grau específic, amb la política de supressió en cascada per garantir la coherència de dades.

10.2.2 Usuaris Creador i Consumidor

- **Usuari Base** (`Usuario`)**:**
  - Hereta de `AbstractUser`, proporcionant camps bàsics com nom d'usuari, correu electrònic, contrasenya, etc.
- **Consumidor** (`Consumidor`)**:**
  - `usuario`: `Primary Key OneToOne(Usuario)` - Crea una relació unívoca amb `Usuario`, usant l'usuari com a clau primària.
  - `data_naixement`: `DateTimeField` - Data de naixement del consumidor.
  - `foto_perf`: `CharField` - Camí opcional de la imatge de perfil.
  - `grau_id`: `ForeignKey(Grado)` - Connecta el consumidor amb el grau que estudia.
  - `dni`: `CharField` - DNI del consumidor. Ha de ser únic.
  - `telefon`: `CharField` - Telèfon de contacte.
- **Creador** (`Creador`)**:**

- `usuario`: `Primary Key OneToOneField` - Relació unívoca amb `Usuario`.
- `foto_perf_cre`: `CharField` - Imatge per defecte de perfil.
- `dni`: `CharField` - DNI únic.
- `linkedin`, `twitter`, `instagram`: `URLField` - Enllaços a xarxes socials.
- `compte_banc`: `CharField` - Informació del compte bancari.
- `bio`: `CharField` - Biografia del creador.

10.2.3 Cursos i Contingut didàctic

- **Curs** (`Curso`)**:**
  - `id`: `int` - Identificador únic del curs.
  - `nombre`: `CharField` - Nom del curs.
  - `descripcion`: `TextField` - Descripció detallada del curs.
  - `precio`: `FloatField` - Preu del curs.
  - `creador_id`: `ForeignKey(Creador, on_delete=models.CASCADE)` - Relaciona cada curs amb un creador.
  - `asignatura_id`: `ForeignKey(Asignatura, on_delete=models.CASCADE)` - Relaciona el curs amb una assignatura específica.
  - `portada_curso`: `ImageField` - Imatge de portada opcional.
  - `video_presentacion`: `FileField` - Vídeo de presentació opcional.
  - `fecha_creacion`: `DateTimeField` - Data de creació automàtica del curs.
  - `publicado`: `BooleanField(default=False)` - Estat de publicació del curs.
  - `precio_desc`: `FloatField` - Preu amb descompte.
- **Lliçó** (`Leccion`)**:**
  - `id`: `int` - Identificador únic de la lliçó.
  - `nombre`: `CharField` - Nom de la lliçó.
  - `curso_id`: `ForeignKey(Curso, on_delete=models.CASCADE)` - Relaciona la lliçó amb un curs concret.
  - `numero`: `IntegerField` - Número d'ordre de la lliçó dins del curs.
- **Document** (`Documento`)**:**
  - `id`: `int` - Identificador únic del document.
  - `leccion_id`: `ForeignKey(Leccion, on_delete=models.CASCADE)` - Relaciona el document amb una lliçó específica.
  - `titulo`: `CharField` - Títol del document.
  - `apuntes_url`: `URLField` - Enllaç als apunts o documents relacionats.

  - `gratis`: `BooleanField(default=False)` - Indica si el document és gratuït.

- **Cuestionario** (`Cuestionario`)**:**
  - `id`: `int` - Identificador únic del qüestionari.
  - `leccion_id`: `ForeignKey(Leccion, on_delete=models.CASCADE)` - Relaciona el qüestionari amb una lliçó específica.
  - `titulo`: `CharField` - Títol del qüestionari.
  - `gratis`: `BooleanField(default=False)` - Indica si el qüestionari és gratuït.

- **Vídeo** (`Video`)**:**
  - `id`: `int` - Identificador únic del vídeo.
  - `leccion_id`: `ForeignKey(Leccion, on_delete=models.CASCADE)` - Relaciona el vídeo amb una lliçó específica.
  - `titulo`: `CharField` - Títol del vídeo.
  - `video_url`: `URLField` - Enllaç al vídeo.
  - `gratis`: `BooleanField(default=False)` - Indica si el vídeo és gratuït.
  - `estado`: `IntegerField` - Estat del vídeo, amb els valors possibles: `NO_VID` (0), `PENDING` (1), `PROCESSING` (2), `AVIABLE` (3), `ERROR` (4).

- **Pregunta Cuestionario** (`PreguntaCuestionario`)**:**
  - `id`: `int` - Identificador únic de la pregunta.
  - `cuestionario_id`: `ForeignKey(Cuestionario, on_delete=models.CASCADE)` - Relaciona la pregunta amb un qüestionari concret.
  - `texto`: `TextField` - Text de la pregunta.
  - `op_a`: `CharField` - Opció A.
  - `op_b`: `CharField` - Opció B.
  - `op_c`: `CharField` - Opció C.
  - `op_d`: `CharField` - Opció D.
  - `op_correcta`: `CharField` - Opció correcta.

- **Preguntes Vídeo** (`PreguntasVideo`)**:**
  - `id`: `int` - Identificador únic de la pregunta de vídeo.
  - `video_id`: `ForeignKey(Video, on_delete=models.CASCADE)` - Relaciona la pregunta amb un vídeo concret.
  - `texto`: `TextField` - Text de la pregunta.
  - `op_a`: `CharField` - Opció A.
  - `op_b`: `CharField` - Opció B.

- `op_c`: `CharField` - Opció C.
- `op_d`: `CharField` - Opció D.
- `op_correcta`: `CharField` - Opció correcta.
- `video_sec`: `IntegerField` - Segon específic del vídeo on apareix la pregunta.

10.2.4 Subscripcions i Valoracions

- **Subscripció** (`Suscripcion`)**:**
  - `id`: `int` - Identificador únic de la subscripció.
  - `consumidor_id`: `ForeignKey(Consumidor, on_delete=models.CASCADE)` - Relaciona la subscripció amb un consumidor específic.
  - `curso_id`: `ForeignKey(Curso, on_delete=models.CASCADE)` - Relaciona la subscripció amb un curs específic.
  - `fecha_suscripcion`: `DateTimeField` - Data de creació de la subscripció.
  - `fecha_final`: `DateTimeField` - Data de finalització de la subscripció.
  - `pagado`: `BooleanField(default=False)` - Indica si la subscripció ha estat pagada al creador.
  - `dinero_pagado`: `FloatField` - Quantitat de diners pagada per la subscripció.
- **Valoració** (`Valoracion`)**:**
  - `id`: `int` - Identificador únic de la valoració.
  - `curso_id`: `ForeignKey(Curso, on_delete=models.CASCADE)` - Relaciona la valoració amb un curs específic.
  - `consumidor_id`: `ForeignKey(Consumidor, on_delete=models.CASCADE)` - Relaciona la valoració amb un consumidor específic.
  - `texto`: `TextField` - Text de la valoració.
  - `fecha`: `DateTimeField` - Data de creació de la valoració.
  - `valoracion`: `FloatField` - Puntuació de la valoració.

10.3 Configuració d'AWS

Per integrar la nostra plataforma amb serveis d'AWS hem configurat diversos components per fer que tot funcioni de forma segura i eficient. A continuació, es descriuen les configuracions clau implementades.

10.3.1 Creació del Compte AWS i Usuari IAM

El primer pas que vam fer per integrar la nostra plataforma amb AWS va ser crear un compte AWS i configurar un usuari IAM (Identity and Access Management) amb els permisos necessaris. A continuació es mostren els passos fets per a aquesta configuració inicial:

1. Crear un compte a AWS.
2. Accedir al servei IAM.
3. Crear un nou usuari IAM.
4. Assignar permisos d'administrador.
5. Generar claus d'accés (Access Key ID i Secret Access Key) per a l'usuari IAM i guardar-les.

Amb les claus generades ja podrem interactuar amb AWS mitjançant el nostre codi. En les següents seccions, parlarem de com les hem fet servir.

10.3.2 CORS en S3

En el nostre S3, on tenim tots els arxius M3U8, els arxius TS, i els PDFs, hem tingut que definir el CORS (Cross-Origin Resource Sharing). Al estar en desenvolupament i el servidor al núvol, hem donat un accés complet per poder accedir localment. A continuació es mostra la configuració de CORS que hem utilitzat:

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "HEAD",
            "POST",
            "PUT"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

10.3.3 Polítiques de S3

A més de la configuració de CORS, hem definit polítiques de bucket per assegurar que només es poden accedir als recursos del bucket sota condicions específiques. Aquí teniu l'exemple de la política de bucket que hem utilitzat:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SignedAccessForM3U8Files",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/videos/*/*.m3u8",
            "Condition": {
                "StringEquals": {
                    "s3:signatureversion": "AWS4-HMAC-SHA256"
                }
            }
        },
        {
            "Sid": "DenyNonSignedAccessToM3U8Files",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/videos/*/*.m3u8",
            "Condition": {
                "Null": {
                    "s3:signatureversion": "true"
                }
            }
        },
        {
            "Sid": "AllowS3Actions",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::colidemy-v2-api/videos/*"
        },
        {
            "Sid": "SignedAccessForPDFS",
            "Effect": "Allow",
```

```
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/pdfs/*/*.pdf",
            "Condition": {
                "StringEquals": {
                    "s3:signatureversion": "AWS4-HMAC-SHA256"
                }
            }
        },
        {
            "Sid": "DenyNonSignedAccessToPDFs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/pdfs/*/*.pdf",
            "Condition": {
                "Null": {
                    "s3:signatureversion": "true"
                }
            }
        },
        {
            "Sid": "SignedAccessForTSFiles",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/videos/*/*.ts",
            "Condition": {
                "StringEquals": {
                    "s3:signatureversion": "AWS4-HMAC-SHA256"
                }
            }
        },
        {
            "Sid": "DenyNonSignedAccessToTSFiles",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::colidemy-v2-api/videos/*/*.ts",
            "Condition": {
                "Null": {
                    "s3:signatureversion": "true"
                }
            }
        }
    ]
```

```
}
```

Aquesta política fa que només es pugui accedir als fitxers M3U8, TS, i PDF mitjançant sol·licituds signades perquè un pirata no pugui anar descarregant una a una les parts del vídeo i reconstruir-ho després.

10.4 Documentació URLS API

La configuració de les URLs de la plataforma es realitza a través de l'ús de l'objecte `urlpatterns` en Django, que connecta les URLs perquè puguin ser consumides pel client.

10.4.1 Gestió d'Autenticació

Aquestes rutes manegen la creació de sessions i la gestió de l'accés d'usuaris, essencial per a la seguretat i la funcionalitat de la plataforma:

- **Registrar Usuari:**

  ```
  path('register', views.RegisterUser, name='register_user'),
  ```

  Aquesta ruta permet als usuaris nous crear un compte en la plataforma.

- **Iniciar Sessió:**

  ```
  path('login', views.login, name='login_user'),
  ```

  Aquesta ruta proporciona el mecanisme perquè els usuaris existents puguin iniciar sessió.

- **Registrar Creador de Continguts:**

  ```
  path('register-creator', creator_endpoints.RegisterCreator,
  name='register_creator'),
  ```

  Similar a la ruta de registre d'usuari, però específica per a usuaris que desitgen crear contingut dins de la plataforma, després d'haver-se registrat.

- **Obtenir Token JWT:**

  ```
  path('token', TokenObtainPairView.as_view(),
  name='token_obtain_pair'),
  ```

  Genera un token d'accés i un token de refresc per l'usuari.

- **Refrescar Token JWT:**

  ```
  path('token/refresh', TokenRefreshView.as_view(),
  name='token_refresh'),
  ```

  Permet refrescar el token fent ús del token de refresc

10.4.2 Gestió d'Usuaris Creadors

Les rutes següents faciliten la gestió d'informació relativa als usuaris que creen continguts en la plataforma:

- **Obtenir Informació del Creador:**

```
path('get-creator-info', creator_endpoints.getCreatorInfo,
name='get_creator_info'),
```

  Aquesta ruta permet als creadors obtenir informació sobre el seu perfil.

- **Actualitzar Informació del Creador:**

```
path('update-creator-info', creator_endpoints.updateCreatorInfo,
name='update_creator_info'),
```

  Permet als creadors actualitzar la seva informació de perfil.

- **Comprovar si és Creador:**

```
path('check-if-creator', creator_endpoints.checkifcreator,
name='check_if_creator'),
```

  Aquesta ruta s'utilitza per verificar si un usuari és creador.

10.4.3 Gestió de Contingut per part dels Creadors

Les següents rutes permeten als creadors de contingut gestionar documents, vídeos i qüestionaris que són part dels cursos oferts a la plataforma:

- **Crear Curs:**

```
path('create-course', creator_endpoints.createCurso,
name='create_course'),
```

  Aquesta ruta permet als creadors afegir nous cursos a la plataforma.

- **Obtenir Cursos del Creador:**

```
path('get-courses-creator', creator_endpoints.getCursosCreador,
name='get_courses'),
```

Aquesta ruta proporciona una llista de tots els cursos creats per un usuari específic.

- **Obtenir Curs del Creador per ID:**

```
path('creator-get-course/<int:curso_id>',
course_editor.getCreatorCourseById, name='get_creator_course_by_id'),
```

Permet obtenir informació específica d'un curs creat per un creador.

- **Actualitzar Curs:**

```
path('update-course/<int:curso_id>',
course_editor.update_course, name='update_course'),
```

Permet als creadors actualitzar els detalls dels seus cursos existents.

- **Publicar Curs:**

```
path('creator-publish-course/<int:curso_id>',
course_editor.creatorPublishCourse, name='creator_publish_course'),
```

Permet als creadors publicar un curs després d'haver-lo creat i revisat.

10.4.4 Gestió d'Universitats i Facultats

Les següents rutes proporcionen funcionalitats per a la gestió i obtenció d'informació sobre institucions educatives. Són utilitzades principalment durant la fase de desenvolupament i configuració de la plataforma:

- **Afegir Universitats:**

```
path('addUniversities', views.addUniversities,
name='add_universities'),
```

Permet afegir noves universitats a la base de dades.

- **Crear Facultats:**

```
path('universidades/<int:universidad_id>/creafacultades',
views.create_facultades, name='create_facultades'),
```

Ruta per afegir noves facultats a una universitat existent.

- **Crear Graus:**

```
path('universidades/<int:facultad_id>/creargrados',
views.create_grados, name='create_grados'),
```

  Permet afegir nous graus a una facultat.

- **Crear Assignatures:**

```
path('grados/<int:grado_id>/crearasignaturas',
views.create_asignaturas, name='create_asignaturas'),
```

  Permet afegir noves assignatures a un grau.

10.4.5 Consum de Cursos

Aquestes rutes permeten als usuaris consumir el contingut dels cursos, accedir als materials de suport i interactuar amb el contingut educatiu disponible a la plataforma:

- **Obtenir Informació del Curs:**

```
path('get-course-info/<int:curso_id>', course_get.getCourseInfo,
name='get_course_info'),
```

  Aquesta ruta proporciona informació detallada sobre un curs específic, incloent descripció, continguts i professorat.

- **Obtenir Lliçons del Curs:**

```
path('get-course-lecciones/<int:curso_id>',
course_get.getCourseLecciones, name='get_course_lecciones'),
```

  Permet obtenir la llista de lliçons d'un curs específic.

- **Comprovar Subscripció al Curs:**

```
path('check-if-subscribed/<int:curso_id>',
course_get.checkIfSubscribed, name='check_if_pagado'),
```

  Comprova si un usuari està subscrit a un curs específic.

- **Subscriure's al Curs:**

```
path('subscribe-course/<int:curso_id>', course_get.subscribeCourse,
name='subscribe_course'),
```

Permet als usuaris subscriure's a un curs per accedir al seu contingut.

- **Obtenir Targetes de Cursos:**

```
path('get-courses-cards', course_get.getCoursesCards,
name='get_courses_cards'),
```

Aquesta ruta proporciona una llista de targetes amb informació resumida de tots els cursos disponibles.

- **Obtenir Document del Curs:**

```
path('get-course-document/<int:document_id>',
course_get.get_document, name='get_document'),
```

Permet accedir als documents associats a un curs.

- **Obtenir Qüestionari del Curs:**

```
path('get-course-quiz/<int:quiz_id>', course_get.get_cuestionario,
name='get_quiz'),
```

Retorna la informació d'un qüestionari.

- **Obtenir Vídeo del Curs:**

```
path('get-course-video/<int:video_id>', course_get.get_video,
name='get_video'),
```

Retorna la informació d'un vídeo.

- **Obtenir Cursos de l'Usuari:**

```
path('get-user-courses', course_get.getUserCourses,
name='get_user_courses'),
```

Proporciona una llista de tots els cursos als quals un usuari està subscrit.

- **Obtenir Targetes de Cursos per Grau:**

```
path('get-courses-cards-by-grado/',
course_get.getCoursesCardsByGrado, name='get_courses_cards_by_grado'),
```

Permet obtenir targetes de cursos filtrats pel grau acadèmic.

10.4.6 Gestió de Vídeos amb AWS

Aquestes rutes gestionen la signatura de URLs per a la càrrega i descàrrega de vídeos utilitzant Amazon Web Services (AWS):

- **Obtenir URL Signada per Pujar Vídeo:**

  ```
  path('sign-put-url/<str:file_type>/<int:id>',
  video_editor.generate_presigned_url_put, name='sign_put'),
  ```

  Aquesta ruta genera una URL preautoritzada per permetre la càrrega directa de vídeos a S3.

- **Obtenir URL Signada per Descàrrega de TS:**

  ```
  path('sign-url', video_editor.getSignedUrlTs,
  name='sign_ts'),
  ```

  Permet obtenir una URL preautoritzada per obtenir segments de vídeo TS, de aws.

10.5 Testing: Coverage

Aquest apartat mostra el coverage complet del backend del projecte, indicant la cobertura de tests per a cada fitxer i funció. Indicant el fitxer, la funció, els Statements (S), Missing(M) i Coverage(C).

Fitxer	**Funció**	**S**	**M**	**C (%)**
aws.py	get_authenticated_url_pdf	7	3	57
aws.py	get_authenticated_url_video	7	3	57
aws.py	get_authenticated_url_ts	7	7	0
aws.py	generate_presigned_put	5	2	60
aws.py	generate_presigned_status_put	5	2	60
aws.py	generate_presigned_status_get	5	2	60
aws.py	(no function)	13	0	100
course_editor.py	getCreatorCourseById	12	0	100
course_editor.py	update_course	35	12	66
course_editor.py	creatorPublishCourse	7	2	71
course_editor.py	(no function)	17	0	100
course_get.py	getCoursesCards	12	12	0
course_get.py	getCourseInfo	8	8	0
course_get.py	getCourseLecciones	11	0	100
course_get.py	get_document	11	0	100
course_get.py	get_cuestionario	16	2	88
course_get.py	get_video	18	6	67
course_get.py	checkIfSubscribed	12	2	83
course_get.py	subscribeCourse	11	1	91
course_get.py	getUserCourses	11	0	100
course_get.py	getCoursesCardsByGrado	19	2	89
course_get.py	(no function)	37	0	100
course_rating.py	get_course_rating	11	2	82
course_rating.py	getMeanRating	7	0	100
course_rating.py	rate_course	22	9	59
course_rating.py	(no function)	13	0	100
creator_endpoints.py	getCursosCreador	3	0	100
creator_endpoints.py	createCurso	14	4	71
creator_endpoints.py	RegisterCreator	28	5	82
creator_endpoints.py	checkifcreator	5	2	60
creator_endpoints.py	getCursosCreatorDashboard	7	7	0
creator_endpoints.py	getCreatorInfo	6	2	67
creator_endpoints.py	updateCreatorInfo	18	3	83
creator_endpoints.py	(no function)	27	0	100
creator_stats.py	lastWeekStats	11	4	64
creator_stats.py	(no function)	10	0	100

Taula 1: Coverage Report (Part 1)

Fitxer	**Funció**	**S**	**M**	**C (%)**
cuestionario_editor.py	creatorGetPreguntasCuestionario	14	6	57
cuestionario_editor.py	creatorSaveCuestionario	17	5	71
cuestionario_editor.py	(no function)	12	0	100
document_editor.py	creatorGetDocumento	14	5	64
document_editor.py	generate_presigned_url_put	22	7	68
document_editor.py	(no function)	14	0	100
home_gets.py	getMostPopularCoursesLastMonth	15	0	100
home_gets.py	searchCourses	10	0	100
home_gets.py	(no function)	9	0	100
video_editor.py	creatorGetVideo	19	8	58
video_editor.py	creatorUpdateVideoQuiz	16	3	81
video_editor.py	handle_questions_update	8	0	100
video_editor.py	getSignedUrlTs	5	5	0
video_editor.py	generate_presigned_url_put	15	15	0
video_editor.py	get_video_status_urls	6	1	83
video_editor.py	(no function)	25	0	100
views.py	RegisterUser	31	8	74
views.py	login	12	3	75
views.py	addUniversities	6	1	83
views.py	create_facultades	16	3	81
views.py	create_grados	16	3	81
views.py	create_asignaturas	17	3	82
views.py	get_universidades	3	0	100
views.py	get_facultades_universidad	3	0	100
views.py	get_grados_facultat	3	0	100
views.py	get_assignatures_grau	3	0	100
views.py	get_cursos_universitat	13	1	92
views.py	get_cursos_facultat	13	1	92
views.py	get_cursos_grau	13	1	92
views.py	get_cursos_assignatura	13	1	92
views.py	get_user	9	2	78
views.py	save_consumer_info	10	1	90
views.py	(no function)	45	0	100
Total		**915**	**187**	**80**

Taula 2: Coverage Report (Part 2)

Referències

[1] Ahmed Waheed. How to create restful apis with django rest framework? https://apidog.com/blog/django-rest-framework-create-api/, Nov 20 2023.

[2] React: A javascript library for building user interfaces. https://github.com/facebook/react.

[3] Amazon Web Services. ¿qué es aws? https://aws.amazon.com/es/what-is-aws/.

[4] Cloud infrastructure services vendor market share worldwide from fourth quarter 2017 to first quarter 2024. https://www.statista.com/statistics/967365/worldwide-cloud-infrastructure-services-market-share-v.

[5] Inc. Cloudflare. ¿qué es la transmisión de vídeo en directo en http? https://www.cloudflare.com/es/learning/cdn/what-is-http-live-streaming/.

[6] Apple Pantos, R. (Ed.). Http live streaming 2nd edition draft-pantos-hls-rfc8216bis-14-preliminary-v2. Apple Inc. [Informational RFC], jun 2023.

[7] Boto3: The aws sdk for python. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.

[8] Aws elastic transcoder. https://docs.aws.amazon.com/elastictranscoder/latest/developerguide/introduction.html.

[9] Amazon Web Services. Aws lambda - gestión de recursos informáticos. https://aws.amazon.com/es/lambda/.

[10] Amazon Web Services. System presets - amazon elastic trasncoder. https://docs.aws.amazon.com/elastictranscoder/latest/developerguide/system-presets.html.

[11] Pdf-lib official repo. https://pdf-lib.js.org/.