Programa't, consulta de decrets de Formació Professional

M.Pilar Moreno Terricabras

Grau d'Enginyeria Informàtica

Desenvolupament web

Vicenç Font Sagrista

Santi Caballe Llobet

9 de juny de 2021

FITXA DEL TREBALL FINAL

Títol del treball:	*Programa't, consulta de decrets de FP*
Nom de l'autor:	*Maria Pilar Moreno Terricabras*
Nom del consultor/a:	*Vicenç Font Sagristà*
Nom del PRA:	*Santi Caballé Llobet*
Data de lliurament (mm/aaaa):	*06/2021*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament Web*
Idioma del treball:	*Català*
Paraules clau	*MEAN STACK FP*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

L'aplicatiu desenvolupat és una eina de consulta de Decrets que regulen la formació professional. L'eina és una ajuda pel docent que ha de consultar aquest tipus d'informació constantment durant l'elaboració de les programacions didàctiques. A més l'aplicatiu genera els documents pertinents en una base de dades no relacional automàticament a partir del pdf del decret amb les informacions necessàries del cicle formatiu en qüestió en un primer nivell de concreció i de cada mòdul que forma part del cicle posteriorment i finalment permet la consulta àgil i ràpida d'aquesta informació emmagatzemada a la base de dades.

La tecnologia utilitzada per desenvolupar l'app és una MEAN stack, Mongo Express Angular i Nodejs.

Abstract (in English, 250 words or less):

The application developed is a tool for consulting the papers that regulate vocational training, known as FP. The tool is an aid for the teacher who has to consult this type of information constantly during the elaboration of the didactic curricula. In addition, the application generates the relevant documents in a non-relational database automatically from the pdf of the paper with the necessary information of the subject in question in a first level of concretion and of each module that is part of the subject later and finally allows the agile and fast consultation of this information stored in the database.

The technology used to develop the app is a MEAN stack, Mongo Express Angular and Nodejs.

Índex

1. Introducció

1.1 Context i justificació del Treball

Els docents de formació professional, ens veiem amb l'obligació d'elaborar les programacions didàctiques dels mòduls que impartim cada curs. Per fer-ho hem de consultar els decrets corresponents que regulen el cicle. Un decret és un document pdf d'aproximadament 80-90 pàgines, i la navegació pel mateix és bastant tediosa. Actualment no hi ha cap altra manera de consultar aquesta informació. Per tant existeix una necessitat clara que ajudi al docent a la consulta de la informació.

1.2 Objectius del Treball

L'objectiu principal del treball és l'elaboració d'una aplicació web que en primer lloc permet a qualsevol usuari la consulta dels decrets en versió pdf, a més als usuaris autentificats els permet la consulta de les dades necessàries pel docent per l'elaboració de programacions didàctiques, i el més important que aquestes dades s'han generat automàticament a través de parsejar el text dels arxius pdf amb expressions regulars i generar una base de dades documental.

1.3 Enfocament i mètode seguit

He desenvolupant un aplicatiu MEAN Stack, és a dir basat en la base de dades no relacional MongoDB, amb un backend desenvolupant en NodeJs sobre el framework Express, i un frontend desenvolupat en Angular 2.x, i Angular Material. La naturalesa de les dades és documental, per tant encaixa perfectament una base de dades no relacional com és Mongo, el treball sobre MongoDB encaixa perfectament amb el format dels objectes javascript de NodeJs. I Angular és un framework de frontend que permet fer peticions api al backend i un treball amb components que dóna com a resultat una aplicació modular, fàcilment escalable.

El mètode de treball ha estat primer el plantejament del problema i la necessitat a cobrir, la familiarització i l'estudi de les eines a utilitzar, el desenvolupament de l'aplicatiu (de manera paral·lela el backend i el frontend), les proves de funcionament en diferents casos i l'elaboració de la documentació: la memòria i la presentació.

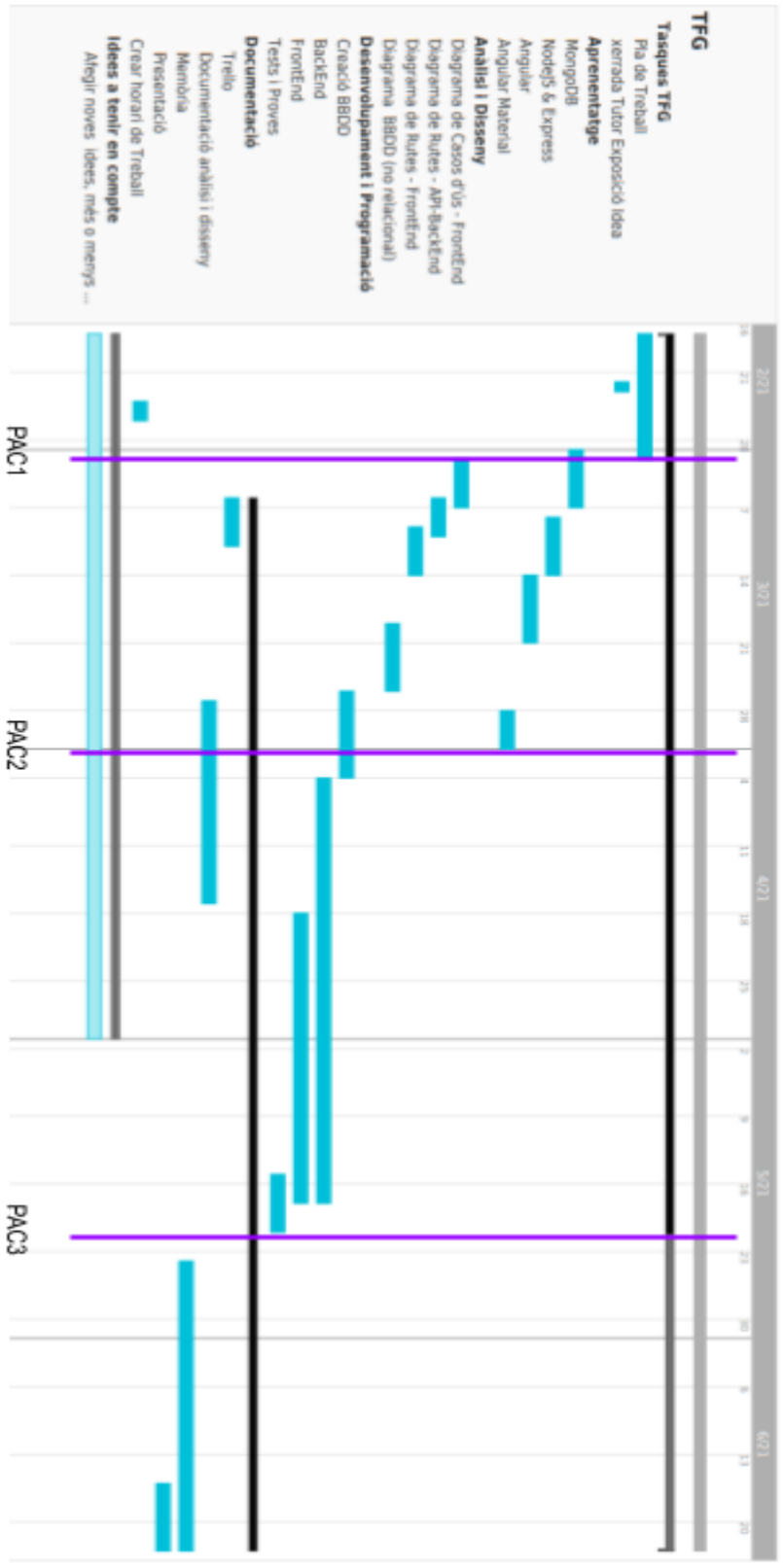
1.4 Planificació del Treball

Recursos necessaris per realitzar el projecte:

- Pc amb sistema operatiu Linux, MacOs o Windows. Mínim 4Gb de RAM, aconsellable 8GB RAM, i 50Gb d'espai lliure al disc
- NodeJS v14 i NPM v7
- Angular CLI 11
- Mongo Atlas
- Editor de text, preferiblement IDE Visual Studio Code

Tasques a realitzar:

- Pla de Treball
- Crear horari personal de Treball
- Xerrada Tutor Exposició Idea
- Aprenentatge
    - MongoDB
    - NodeJS & Express
    - Angular
    - Angular Material
- Anàlisi i Disseny
    - Diagrama de Casos d'ús - FrontEnd
    - Diagrama de Rutes - API-BackEnd
    - Diagrama de Rutes - FrontEnd
    - Diagrama BBDD (no relacional)
- Desenvolupament i Programació
    - Creació de la Base de Dades
    - Back End
    - Front End
    - Tests i Proves
- Documentació
    - Trello
    - Documentació anàlisi i disseny
    - Memòria
    - Presentació
- Idees a tenir en compte
    - Afegir noves idees, més o menys ...


TFG
Tasques TFG
Pla de Treball
xerrada Tutor Exposició Idea
Aprenentatge
MongoDB
NodeJS & Express
Angular
Angular Material
Anàlisi i Disseny
Diagrama de Casos d'Ús - FrontEnd
Diagrama de Rutes - API-BackEnd
Diagrama de Rutes - FrontEnd
Diagrama BBDD (no relacional)
Desenvolupament i Programació
Creació BBDD
BackEnd
FrontEnd
Tests i Proves
Documentació
Trello
Documentació anàlisi i disseny
Memòria
Presentació
Crear horari de Treball
Idees a tenir en compte
Afegir noves idees, més o menys ...
PAC1
PAC2
PAC3
2021
3021
4021
5021
6021

1.5 Breu sumari de productes obtinguts

El producte obtingut és un aplicatiu web que permet la consulta dels decrets que regulen la formació professional agrupats per famílies amb un visualitzador de pdf integrat, que també en permet la impressió i la descàrrega. A més a més pels usuaris autentificats permet la consulta de la informació més rellevant en un format àgil i ràpid gràcies a consultes a la base de dades documental que es genera automàticament a partir dels pdf dels decrets. L'elaboració de la base de dades automàtica és possible perquè els decrets segueixen el mateix format, i es parseja la informació a partir d'expressions regulars.

Inicialment vaig plantejar la possibilitat de crear programacions didàctiques a partir de la BD generada, però una vegada examinat l'abast d'aquesta branca, la descarto i la deixo com una via d'ampliació del projecte.

En principi només es genera la base de dades d'un format concret de decrets, el motiu és que hi ha una reforma curricular que ha d'entrar en vigor aquest curs que ve, però encara no han sortit els decrets pertinents. Per tant no té massa sentit fer aquesta feina amb tant poca validesa en el temps.

1.6 Breu descripció dels altres capítols de la memòria

En aquest treball final de grau s'expliquen les tecnologies que s'han utilitzat per desenvolupar el projecte, el aspectes importants des del punt de vista del disseny, l'arquitectura, la implementació i les conclusions.

Les tecnologies en les que aprofundeixo són principalment Angular, Angular Material, Node, Express, MongoDB i les Expressions Regulars.

Es mostra també el diagrama de casos d'ús, l'estructura de la base de dades i el routing utilitzat tant el en backend com en el frontend.

A continuació s'analitzen les parts de codi que considero més rellevants i destacables, sobretot les que fan referència als objectes d'Angular Observable i Promise, i també l'assincronicitat dels mètodes. Així com l'ús d'interceptors

Finalment s'aporten les conclusions personals després d'haver treballat en aquest projecte, i també s'exposen algunes idees sobre possibles altres usos de l'aplicació o ampliacions que podrien aportar-li valor al projecte.

2. Tecnologies

2.1 SPA

Single Page Application. Consisteix en una app web que en lloc d'utilitzar un nombre de pàgines HTML, és una aplicació web encabida en una sola pàgina HTML. L'objectiu principal de SPA és proporcionar una experiència d'usuari més fluida i una interfície molt més dinàmica. No hi ha limitacions per a l'SPA. Pot ser una petita aplicació que simplement proporciona funcionalitat CRUD fins a un nivell molt més complex amb grans quantitats de sol·licituds. Una aplicació de pàgina única conté moltes biblioteques, plantilles i scripts en diferents carpetes en funció de la seva complexitat.

L'avantatge d'escollir aquest tipus d'aplicació web és poder actualitzar parts d'un lloc web sense enviant la sol·licitud i recarregant una pàgina completa. Aquesta característica és la més interessant d'SPA. Utilitza Ajax (per interactuar amb el servidor), plantilles HTML, un bon marc MVC.

2.2 MEAN Stack

És una aplicació web on JavaScript és el llenguatge utlitzat tant el el frontEnd com en el backEnd. La pila MEAN està formada per tres tecnologies principals orientades a JavaScript, on MongoDB n'és la base de dades, Express el framework de backEnd per interactuar amb node, Angular és el framework de frontEnd i Node.js és l'entorn on s'executa el backend.

2.3 Node.js

Què és Node.js

Node.js és un entorn de temps d'execució de JavaScript (d'aquí la terminació en .js fent al·lusió al llenguatge JavaScript). Aquest entorn inclou tot el que es necessita per executar un programa escrit en JavaScript.

Node.js va ser creat pels desenvolupadors originals de JavaScript. Van transformar javascript d'un llenguatge que només podia executar-se en el

navegador a un llenguatge que es podria executar en els ordinadors com si d'aplicacions independents es tractés.

Tant JavaScript com Node.js s'executen en el motor de temps d'execució JavaScript. Aquest motor agafa el codi JavaScript i el converteix en un codi de màquina més ràpid.

Per a què serveix Node.js?

NODE.JS utilitza un model d'entrada i sortida sense bloqueig controlat per esdeveniments que ho fa lleuger i eficient (amb entrada ens referim a sol·licituds i amb sortida a respostes). Pot utilitzar-se en qualsevol operació, des de llegir o escriure arxius de qualsevol tipus fins a fer una sol·licitud HTTP.

La idea principal de Node.js és usar el model d'entrada i sortida sense bloqueig i controlat per esdeveniments per seguir sent lleuger i eficient davant de les aplicacions en temps real permeten un flux de dades initerromput.

Node.js està completament controlat per esdeveniments. Resumint podem dir que el servidor consta d'un subprocés que processa un esdeveniment rere l'altre.

Avantatges de Node.js

Des d'una perspectiva de desenvolupament de servidor web, Node té un gran nombre d'avantatges:

- Gran rendiment. Node ha estat dissenyat per optimitzar el rendiment i l'escalabilitat en aplicacions web i és un molt bon complement per a molts problemes comuns de desenvolupament web (ex, aplicacions web en temps real).
- JavaScript és un llenguatge de programació relativament nou i es beneficia dels avenços en disseny de llenguatges quan es compara amb altres llenguatges de servidor web tradicionals.
- El gestor de paquets de Node (NPM de l'anglès: Node Packet Manager) permet accedir a centenars o milers de paquets reutilitzables.

- És portable, amb versions que funcionen en Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, i NonStop OS. A més, funciona bé a molts dels proveïdors d'allotjament web, que proporcionen infraestructura específica i documentació per a allotjament de llocs Node.
- Té un ecosistema i comunitat de desenvolupadors de tercers molt activa, amb quantitat de gent disposada a ajudar.

2.4 Express

Express és el framework web més popular de Node ens proporciona mecanismes per programar controladors de peticions amb diferents accions HTTP en diferents rutes URL. També per afegir processament de peticions "middleware" addicional en qualsevol punt dins de la pipe que gestiona la petició.

Tot i que Express és en si mateix bastant minimalista, els desenvolupadors han creat molts paquets de middleware compatibles per abordar gairebé qualsevol problema de desenvolupament web. Hi ha llibreries per treballar amb galetes, sessions, inicis de sessió d'usuari, paràmetres URL, dades POST, capçaleres de seguretat i molts més.

2.5 Angular 2.x

Angular és un framework opensource desenvolupat per Google per facilitar la creació i programació d'aplicacions web SPA. Angular separa completament el frontend i el backend en l'aplicació, evita escriure codi repetitiu i manté tot més ordenat gràcies al seu patró MVC assegurant els desenvolupaments amb rapidesa, alhora que possibilita modificacions i actualitzacions.

Quan es canvia una ruta, Angular el que fa 'per sota' és canviar la vista d'una part de la finestra, però de forma més dinàmica. Entre altres avantatges, aquest framework és modular i escalable adaptant-se a les nostres necessitats i a l'estar basat en l'estàndard de components web, i amb un conjunt d'interfície de

programació d'aplicacions (API) permet crear noves etiquetes HTML personalitzades que poden reutilitzar-se, són el que anomenem components.

El llenguatge principal de programació d'Angular és Typescript, i així tota la sintaxi i la manera de fer les coses en el codi és el mateix, el que afegeix coherència i consistència a la informació, permetent per exemple, la incorporació de nous programadors, en cas de ser necessaris, ja que poden continuar el seu treball sense excessiva dificultat.

Com ja s'ha indicat, les plantilles de Angular s'emmagatzemen per separat el codi de la interfície d'usuari (front-end) i el de la lògica de negoci (back-end), que entre altres beneficis permet utilitzar millor altres eines anteriorment existents.

Per la seva programació reactiva, la vista s'actualitza automàticament després de realitzar els canvis. A més Angular disposa d'assistent per línia d'ordres per poder crear projectes base i també s'integra bé amb eines de testing i amb Ionic, boostrap i **Angular Material** , cosa que facilita la creació de web-responsive, és a dir, adaptades a mòbils o a qualsevol mida de dispositiu.

Angular Material són una sèrie de components de disseny per a Angular. Aquests components són internacionalitzats i accessibles per a tothom. A més estan molt testejats per garantir el rendiment i la fiabilitat. Tenen APIs senzilles amb un comportament multiplataforma consistent. Són versàtils i proporcionen les eines que ajuden els desenvolupadors a construir els seus propis components personalitzats amb patrons d'interacció comuns. Són fàcilment personalitzables dins dels límits de l'especificació de disseny de materials. I s'integren perfectament amb Angular.

2.6 MongoDB

MongoDB és una base de dades documental amb l'escalabilitat i la flexibilitat que necessàries que permet consultes i indexacions.

El model de documents de MongoDB és senzill perquè els desenvolupadors l'aprenguin i l'utilitzin, tot i que proporciona totes les capacitats necessàries per

satisfer els requisits més complexos a qualsevol escala. Ofereix controladors per a més de deu llenguatges de programació i la comunitat n'ha construït dotzenes més.

MongoDB emmagatzema dades en documents semblants a JSON flexibles, és a dir, els camps poden variar d'un document a un altre i l'estructura de dades es pot canviar amb el pas del temps.

El model de document s'assigna als objectes del codi de l'aplicació, cosa que facilita el treball de les dades

Les consultes, la indexació i l'agregació en temps real proporcionen maneres potents d'accedir i analitzar les dades.

MongoDB és una base de dades distribuïda, de manera que hi ha una alta disponibilitat, escalat horitzontal i distribució geogràfica integrada i fàcil d'utilitzar.

Mongo té un servei de base de dades al núvol: MongoAtlas. La seva versió gratuita ofereix:

- 512 MB d'emmagatzematge
- RAM compartida
- Conjunts de rèpliques molt disponibles, xifratge d'extrem a extrem, patches automatitzats, API REST

2.7 Expressions Regulars

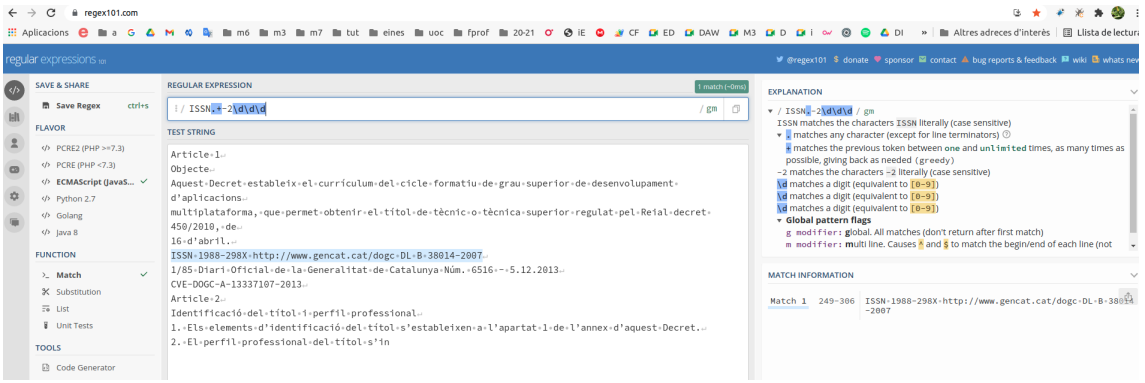
En informàtica, una expressió regular (o col·loquialment anomenades regexp, acrònim de l'anglès regular expression) és una representació, segons unes regles sintàctiques d'un llenguatge formal, d'una porció de text genèric a buscar dins d'un altre text, com per exemple uns caràcters, paraules o patrons de text concrets.

El text genèric de l'expressió regular pot representar patrons (en anglès patterns) amb determinats caràcters que tenen un significat especial (per exemple, el caràcter comodí "*" per representar un nombre qualsevol de

caràcters, o classes com "[abc]" per representar qualsevol dels caràcters 'a', 'b' o 'c').

Una expressió regular està escrita seguint les regles d'un llenguatge formal, que poden ser interpretades per un llenguatge de programació com ara Javascript que inclou un processador d'expressions regulars, i ser capaç d'examinar un text i reconèixer-hi les parts que es corresponen (en anglès match).

L'eina utilitzada per testejar les expressions regulars utilitzades és regex101.com



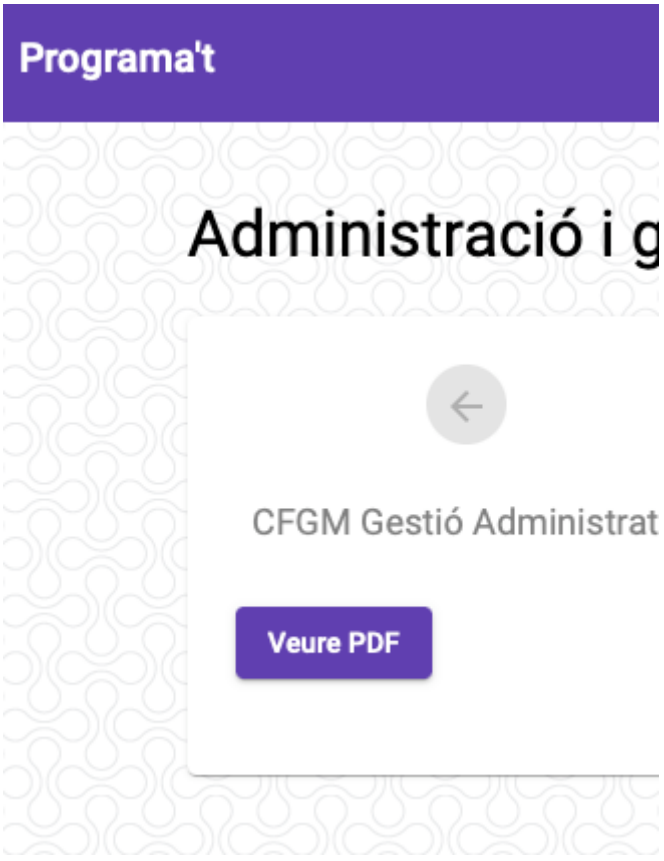
3. Consideracions del disseny

3.1 Objectiu de disseny

La idea és una aplicació neta, clara i molt fàcil d'utilitzar, on no hi hagi res que distraigui a l'usuari de la seva tasca.

A més també es busca una aplicació responsive, que s'adapti a la majoria de dispositius, està pensada especialment per a pc, portàtils, i tauletes. L'ús en mòbil és possible, però no està optimitzat

3.2 Colors

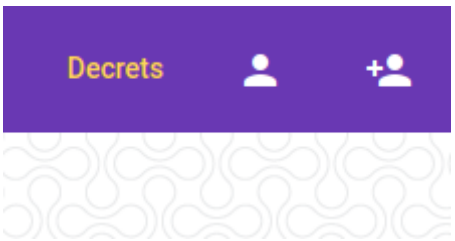


La base de l'aplicatiu és blanc, buscant una estètica google, els colors que donen el contrast són el lila i el gris. És una combinació poc arriscada, però que funciona molt bé. Afegeixo un fons amb un patró molt lleuger per donar un punt de constrast amb el marc central, en blanc

Colors de Marca

Addicionalment afegeixo un color per denotar l'opció escollida dins la web, aquest és el taronja, que contrasta bé amb el lila,

per exemple quan triem una icona, o bé per mostrar l'opció del menú escollida



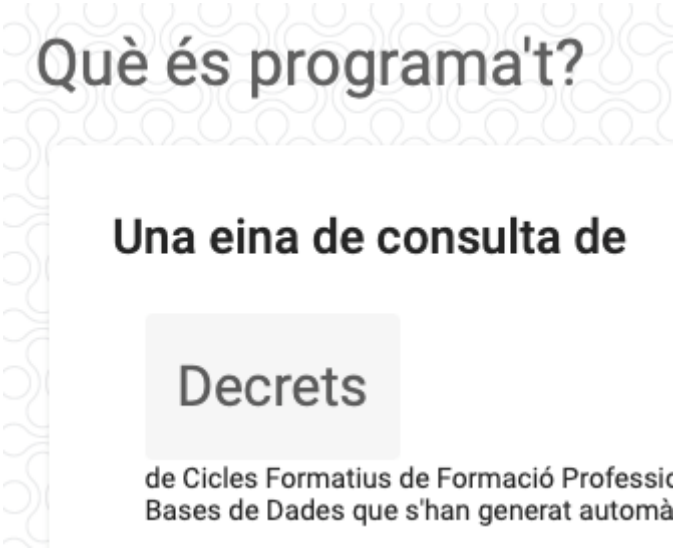
3.3 Tipografia

El valor predeterminat de la font es configura amb el tipus de lletra Roboto de Google amb els estils de pes de tipus de lletra 300, 400 i 500.

Per què Roboto?

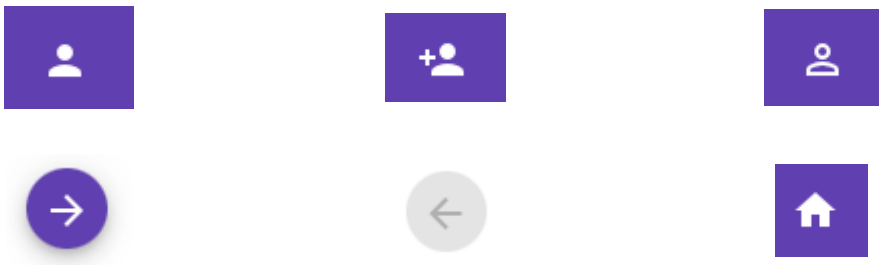
Tenint un disseny clar i directe, no hi ha dubte que Roboto s'ha convertit en un dels tipus de lletra més populars del disseny web. Voldria destacar tres dels principals motius pels quals l'utilitzo:

1. Té una forma llegible, en què l'espai en blanc pren la iniciativa per crear un disseny nítid.
2. L'elegant contrast dels seus traços facilita la lectura del text corporal.
3. El fet de ser un tipus de lletra familiar complet el converteix en una opció fantàstica per al disseny web.



3.4 Icones

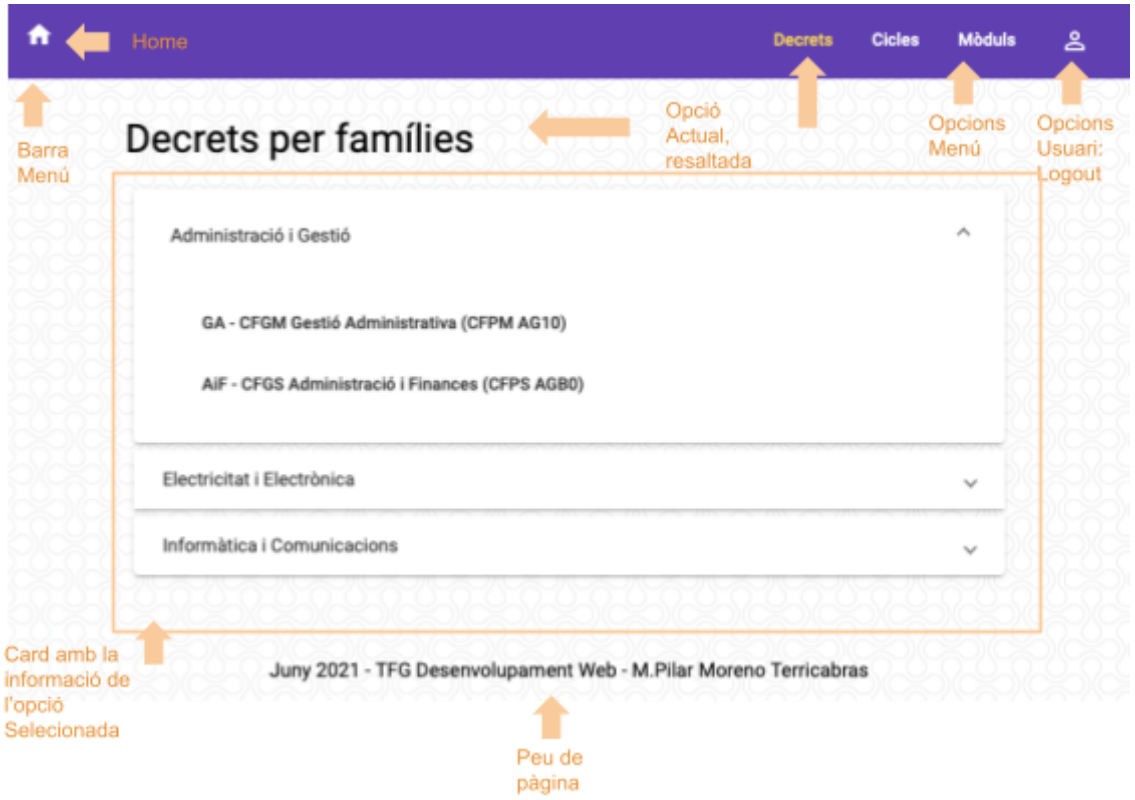
L'ús d'icones per accedir a diferents funcionalitats aporta usabilitat a l'aplicatiu. Algunes de les icones utilitzades són:



3.5 Organització de la finestra de l'aplicatiu

L'organització de la finestra del navegador és uniforme en tot l'aplicatiu. Està formada per :

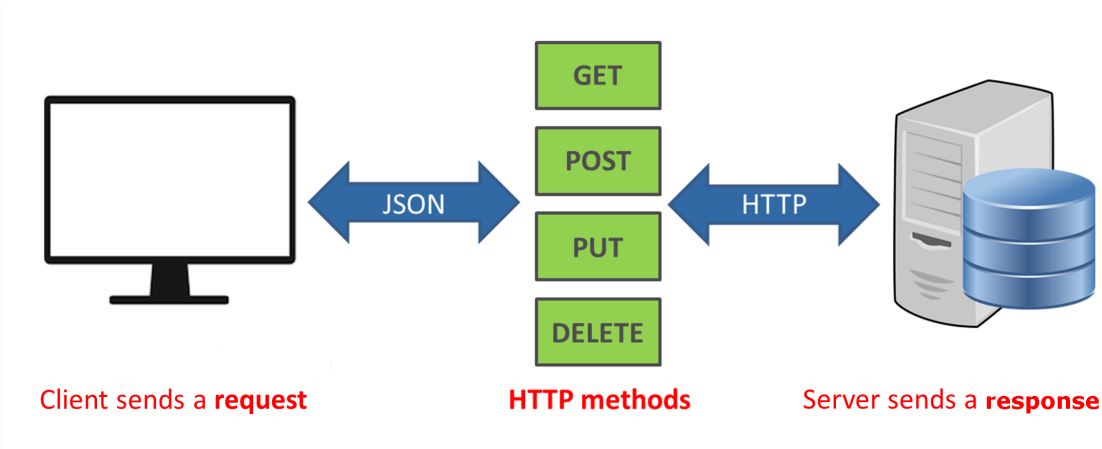
- La Barra de menú. Amb una part a l'esquerra amb el botó Home si estem loguejats o el text programa't. La resta d'opcions es troben a la dreta. per diferenciar les opcions de l'aplicatiu de la part d'autentificació d'usuaris
- La part principal de la pàgina amb una card central que conté la informació seleccionada. Aquesta es mostra amb desplegables que deixen una visió molt neta, i només es mostra la informació rellevant.
- Finalment un peu de pàgina amb la informació mínima.



4. Arquitectura

4.1 Què és una API REST

REST significa Representation State Transfer, que és una connexió sense estat entre clients i servidors. És una arquitectura moderna client-servidor que utilitza el protocol HTTP per comunicar-se i JSON, com a representació de dades. En una simple aplicació CRUD com a exemple, l'API REST s'utilitza per crear les interfícies per a POST (crear), PUT (actualitzar), GET (recuperar) i DELETE (eliminar). La imatge mostra l'exemple del disseny típic de l'API REST.

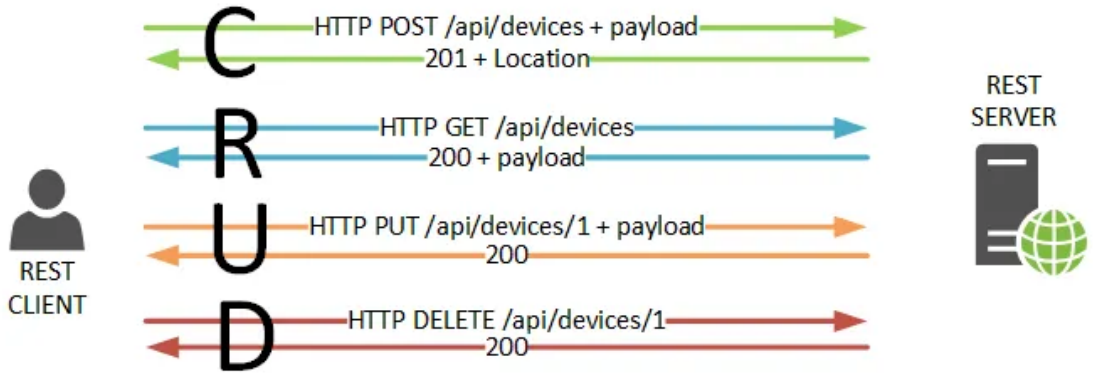


4.2 CRUD

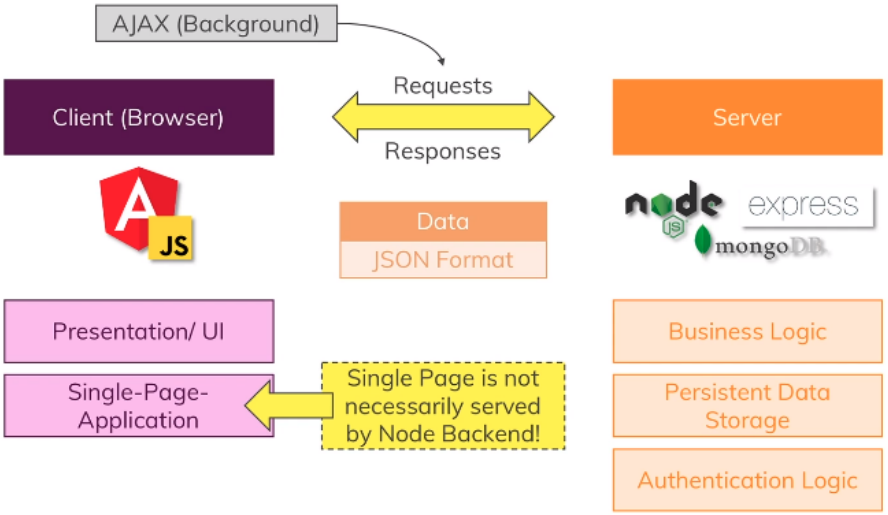
El concepte CRUD està estretament vinculat a la gestió de dades. CRUD fa referència a un acrònim en què es reuneixen les primeres lletres de les quatre operacions fonamentals d'aplicacions persistents en sistemes de bases de dades:

- Create (Crear registres)
- Read / Retrieve (Llegir registres)
- Update (Actualitza registres)
- Delete / Destroy (Esborrar registres)

En poques paraules, CRUD resumeix les funcions requerides per un usuari per crear i gestionar dades. Diversos processos de gestió de dades estan basades en CRUD, en els quals aquestes operacions estan específicament adaptades als requisits de sistema i d'usuari, ja sigui per a la gestió de bases de dades o per a l'ús d'aplicacions. Per als usuaris, CRUD significa crear un document (create) i utilitzar-lo (read), actualitzar-lo (update) o esborrar-lo (delete) en qualsevol moment.



4.3 Funcionament d'una MEAN app



En una MEAN stack, recordem la pila formada per MongoDB Express Angular 2x i Nodejs:

Angular és la tecnologia que utilitzem en el client, es proporciona una interfície d'usari i presentació de l'app d'una manera dinàmica, en format SPA. Per tant Angular s'encarrega de fer peticions al backend, construït amb Node.js i Express, on hi ha la lògica del l'aplicatiu. El backend s'encarregarà autenticar usuaris, generar i relacionar-se amb la base de dades.

Angular funciona en el client, és a dir en el navegador. És un framework que ens permet crear SPA, és a dir que gestiona tot el frontend. Angular necessita obtenir les dades de manera dinàmica per mostrar a l'usuari. Per exemple quan l'administrador puja un nou decret, s'encarrega de gestionar-ho, comunicant-se amb el backend. L'avantatge és que mai hem de recarregar la pàgina, és molt ràpid i reactiu.

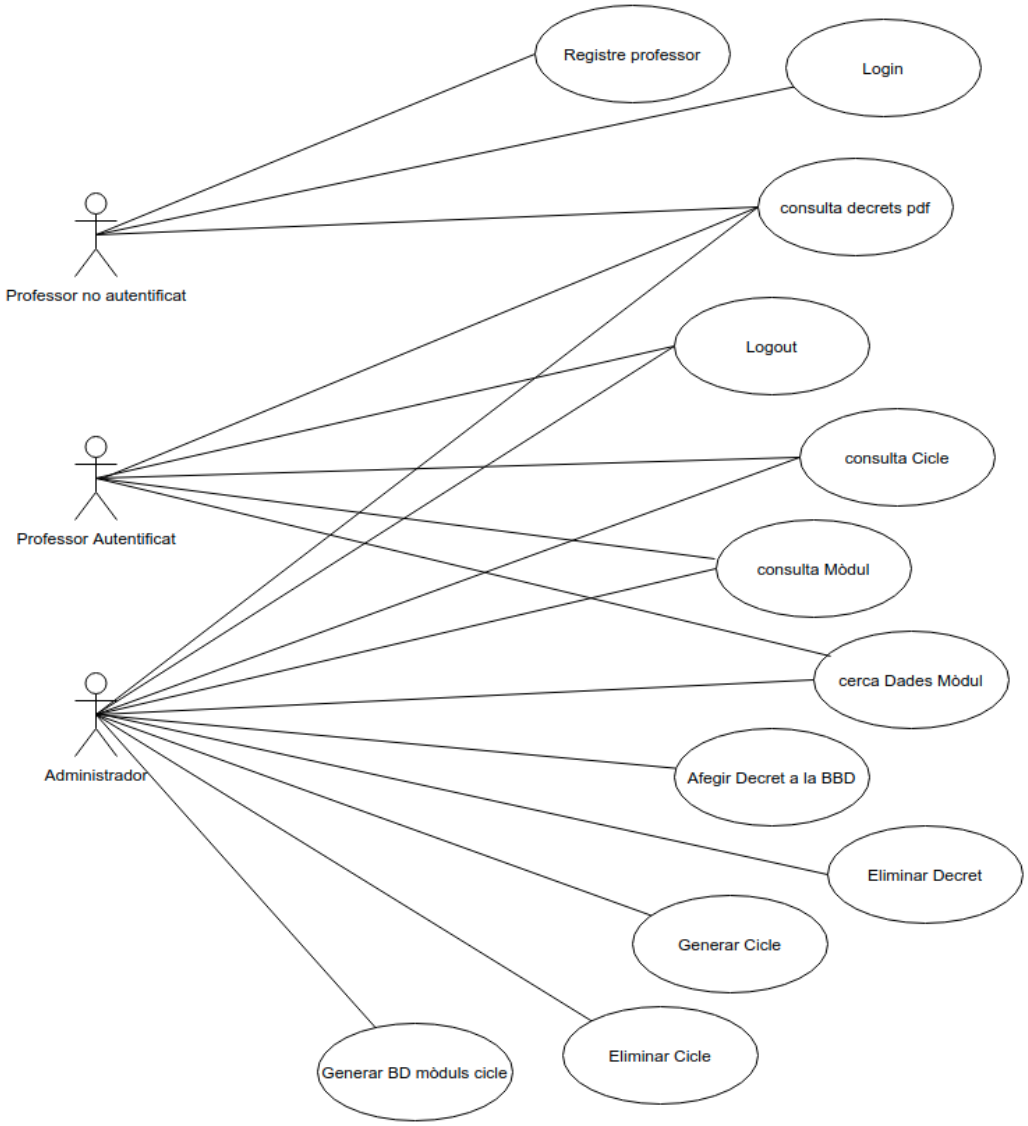
Node.js és el llenguatge de backend, i ens permet utilitzar javascript en el llenguatge. Node.js escoltarà a les peticions del client, i enviarà respostes. També executarà la lògica de la banda del servidor, com per exemple l'autentificació, la generació i la comunicació de la base de dades amb la BBDD, i els arxius.

Utilitzem Node.js amb Express.js que simplifica la programació de la lògica de servidor. Express està basat en middleware, és a dir utilitza funcions per filtrar peticions, com si fos un embut. Es treballa com si fos una cadena que permet escriure codi molt estructurat. Express inclou routing, així podem manipular diferents request a diferents end-points i les peticions a través de urls.

MongoDB és la base de dades no SQL que emmagatzema documents i col·leccions, té una lògica diferent que sql , però finalment emmagatzema dades que volem que persisteixin a la base de dades, podem tenir diferents esquemes, el que ens permet ser més flexibles. Es fàcilment integrable amb Express i Node.js, però mai la connectarem directament amb Angular (tot i que es podria fer), però no volem exposar les nostres credencials en el navegador. Mongo DB és molt flexible i escalable.

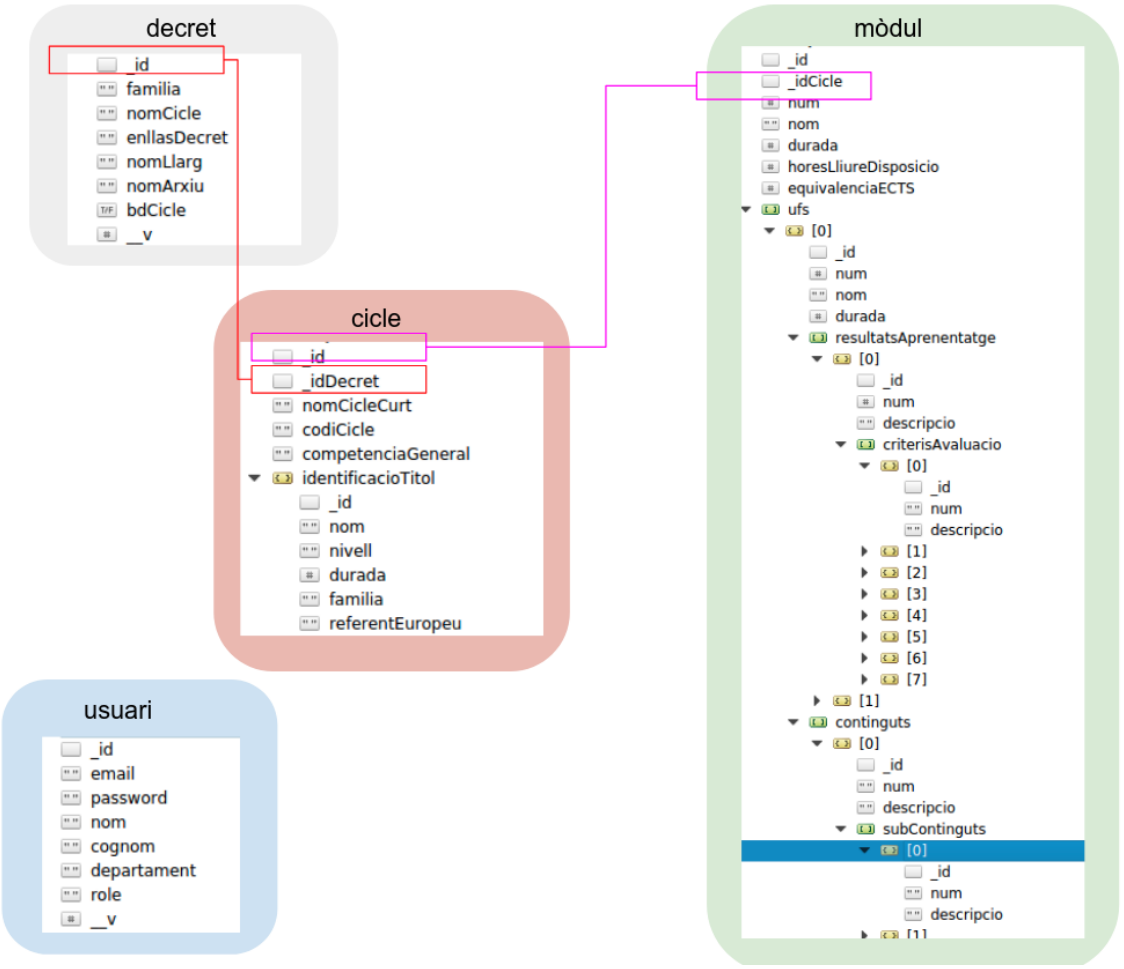
Resumint, la banda del client (frontEnd) seria el que l'usuari veu en el navegador, i a l'altra banda tenim el backEnd on hi ha el servidor amb la lògica de l'aplicatiu, i on l'usuari accedeix indirectament. En el client utilitzem Angular per crear la interfície d'usuari, mentre que al backend utilitzem Node.js/Express per la lògica i Mongo DB com a base de dades. No volem connectar Angular directament a la BBDD. Angular gestiona la presentació de les dades i podria estar allotjat en un ordinador totalment diferent del backend, en l'aplicació node tenim el cor de la lògica, que no ha d'estar exposada al client per raons de seguretat, i per eficiencia, i és on tenim la connexió amb la BBDD, i també l'autentificació. Com es connecten aquestes dues peces? S'intercanvien peticions i respostes, utilitzant AJAX, són peticions que poden ser enviades sense haver de recarregar la pàgina. Les dades es passen en format JSON que es un format eficient, i molt similar a com Mongo emmagatzema les dades.

4.4 Diagrama de casos d'ús



4.5 Estructura de la Base de Dades

Visió general i relacions entre els esquemes



La base de dades és una base de dades documental, en MongoDb cada document té una **_id,** que el permet identificar de manera unívoca, i que es genera automàticament i un camp **_v** que és la clau de versió, una propietat establerta a cada document quan es crea per primera vegada per Mongoose. Aquest valor de claus conté la revisió interna del document. El nom d'aquesta propietat del document és configurable. El valor per defecte és __v.

Esquema usuari

En aquest esquema desarem la informació dels usuaris registrats, desem el seu email, i password, que són els que utilitzarà per loguejar-se, a més nom cognom i departament, cosa que permetria en versions futures només mostrar

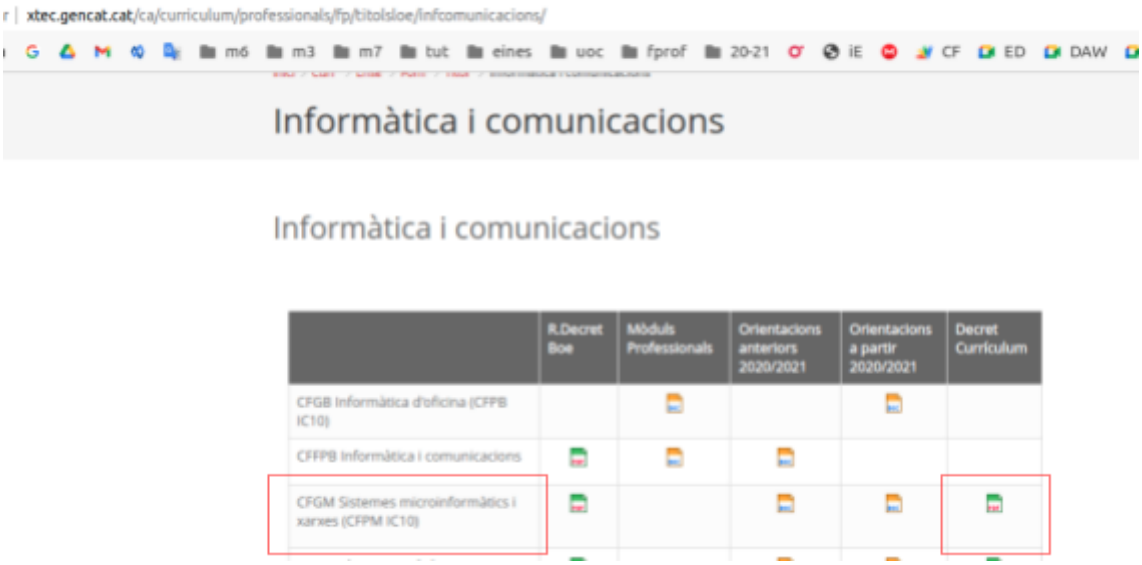
la informació pertinent a la família dels departament, i role, per distingir a usuaris normals (professors) de l'usuari administrador.

Aquí tenim exemple d'un document

```
/* 1 */
{
    "_id" : ObjectId("609fec47028ea5408432f63d"),
    "email" : "pmt@pmt.com",
    "password" : "$2b$10$zo1kJeKv0aXmWD5SHi1qL.bJDSEOopp97FhZhIZiNgNHN2V0UBNVi",
    "nom" : "Pilar",
    "cognom" : "MoTe",
    "departament" : "Informàtica",
    "role" : "user",
    "__v" : 0
}
```

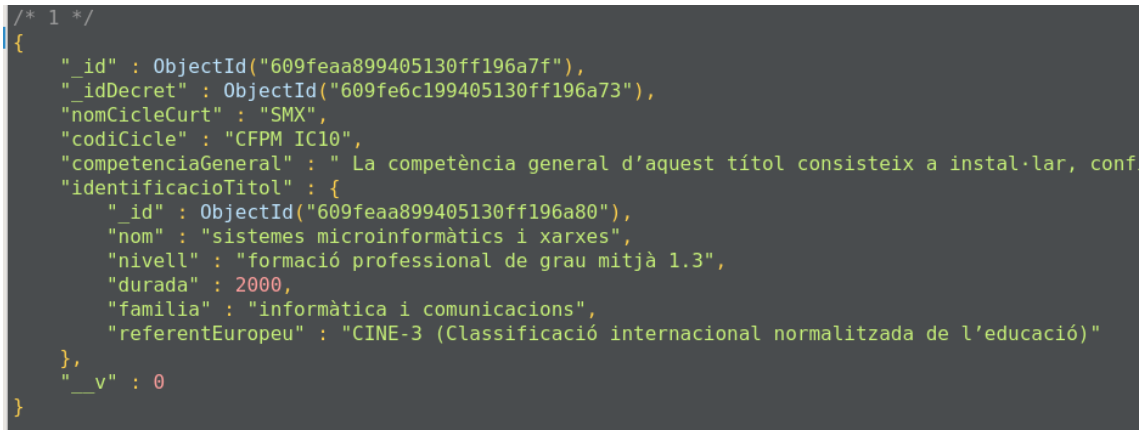
Esquema Decrets

En aquest esquema es desa per cada decret la família, el nom del cicle, enllaç al decret a la xtec, el nom llarg que hi ha a la web de la xtec tal i com mostra la imatge, i la petició que serveix l'arxiu al servidor. Finalemnt un camp booleà indica si s'ha generat o no la Base de dades d'aquest decret.

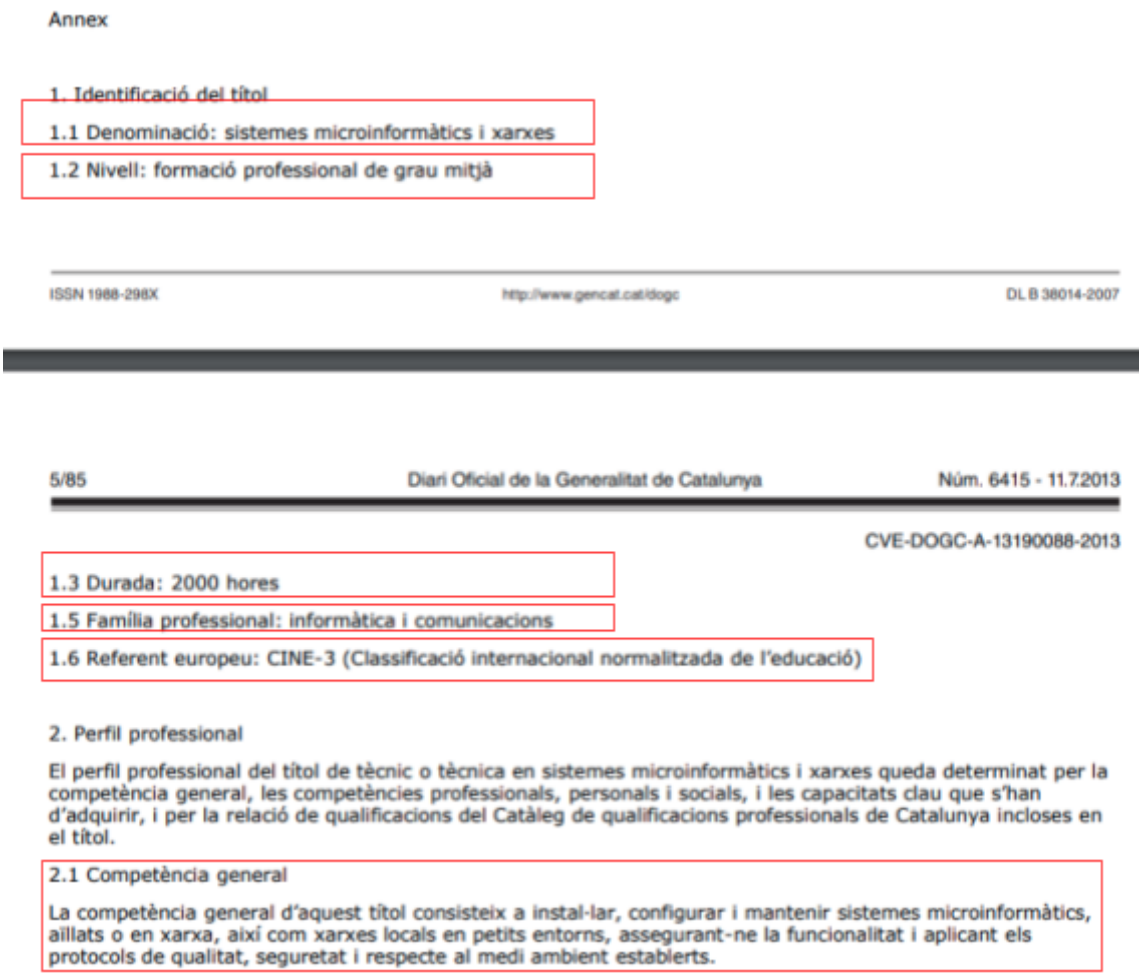


Esquema Cicles

Aquest esquema es genera automàticament a partir de les dades del decret, extraient-net un petit resum amb els següent camps: _idDecret del qual prové, per poder accedir al pdf, el nom curt, el codi del cicle que s'extreu del nomLlarg del decret, la competencia general, que ja trobem al pdf, i finalment un subesquema que és la identificació del títol formada pel nom, el nivell, la durada, la família i el referent europeu, que s'extreuen del pdf.



aquí tenim la imatge d'on s'extreu la informació en el pdf.



Annex

1. Identificació del títol

1.1 Denominació: sistemes microinformàtics i xarxes

1.2 Nivell: formació professional de grau mitjà

ISSN 1988-298X http://www.gencat.cat/dogc DL B 38014-2007

CVE-DOGC-A-13190088-2013

1.3 Durada: 2000 hores

1.5 Família professional: informàtica i comunicacions

1.6 Referent europeu: CINE-3 (Classificació internacional normalitzada de l'educació)

2. Perfil professional

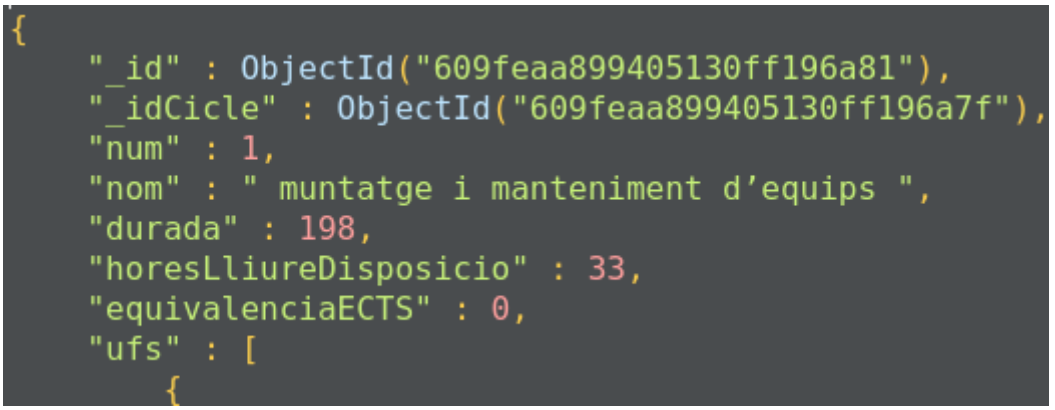
El perfil professional del títol de tècnic o tècnica en sistemes microinformàtics i xarxes queda determinat per la competència general, les competències professionals, personals i socials, i les capacitats clau que s'han d'adquirir, i per la relació de qualificacions del Catàleg de qualificacions professionals de Catalunya incloses en el títol.

2.1 Competència general

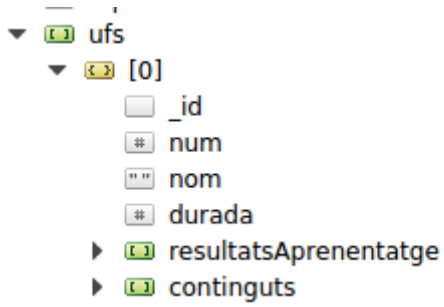
La competència general d'aquest títol consisteix a instal·lar, configurar i mantenir sistemes microinformàtics, aïllats o en xarxa, així com xarxes locals en petits entorns, assegurant-ne la funcionalitat i aplicant els protocols de qualitat, seguretat i respecte al medi ambient establerts.

Esquema Mòduls

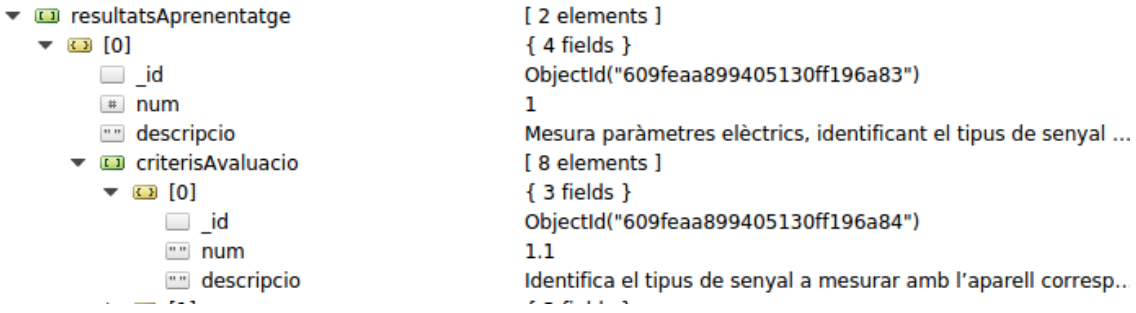
Que representa un mòdul d'un cicle formatiu, està format pels camps _idCicle, que relaciona tots els mòduls d'un mateix cicle, el número amb el qual s'identifica, el nom, la durada en hores, les hores de lliure disposició, si n'hi ha, l'equivalència en ECTS i un array de ufs, que són el subdocument que es genera per cada uf.



per cada uf que compon un mòdul tenim el número, el nom i la durada, a part dos arrays de subdocuments que són els resultats d'aprenentatge i els continguts:



alhora cada resultat d'aprenentatge té un número, una descripció i un array de criteris d'avaluació. i cada criteri d'avaluació té un núm i una descripció.

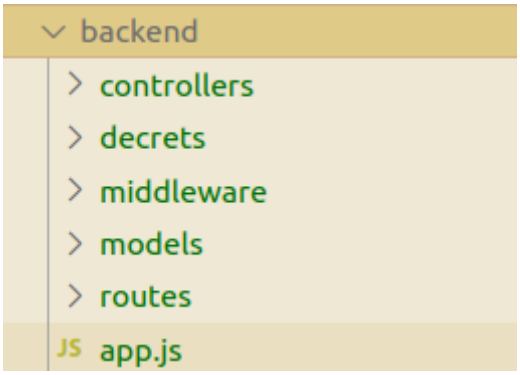


per altra banda els continguts tenen un número i una descripció i un ara de sub continguts format cadascun per un número i una descripció

```
▼ continguts                    [ 2 elements ]
  ▼ [0]                         { 4 fields }
      _id                       ObjectId("609feaa899405130ff196a95")
      num                       1
      descripcio                Mesurament de paràmetres elèctrics:
    ▼ subContinguts             [ 12 elements ]
      ▼ [0]                     { 3 fields }
          _id                   ObjectId("609feaa899405130ff196a96")
          num                   1.1
          descripcio            Tipus de senyals.
```

4.6 Backend

4.6.1 Estructura de Carpetes



El disseny del Backend segueix l'estructura ruta - controlador, on cada ruta fa una crida a un mètode d'un controlador, i a més gestiona tota la comunicació amb la base de dades.

Carpeta models

On definim cada model que hi ha a la base de dades, per exemple per usuari tenim l'arxiu user.js:

backend > models > JS user.js > ...

```
const mongoose = require('mongoose');
const uniqueValidator = require('mongoose-unique-validator'); // cal instal·lar el paquet

// plantilla que  li passem un obkj javascript
// hi hem de definir els camps i el q contenen

const userSchema = mongoose.Schema({
  email: { type: String, required: true, unique:true},
  password: { type: String, required: true},
  nom: { type: String, required: true},
  cognom: { type: String, required: true},
  departament: { type: String, required: true},
  role:{type:String,required:true}

});

userSchema.plugin(uniqueValidator);

// es crea el model

module.exports = mongoose.model('User',userSchema);
```

Carpeta routes

Per cada ruta descrita es crida a un mètode específic d'un controlador determinat. Per exemple seguint amb usuaris:

backend > routes > JS user.js > ...

```
// framework d backend
const express = require("express");
// el router
const router = express.Router();
const checkAuth = require('../middleware/check-auth');


// importem el controllador
const UserController = require("../controllers/user");

// assignem rutes a mètodes del controllador
router.post("/signup", UserController.createUser);
router.post("/login",UserController.userLogin);
router.get("/isadmin/:id",checkAuth,UserController.isAdmin);

module.exports = router;
```

Carpeta Controllers

En la carpeta controllers hi ha cadascun dels controladors que es criden a la ruta, seguint amb el cas de user, tenim definir el controlador on hi ha definits tots els mètodes que cridem a les tures

```
backend > controllers > JS user.js > ...
  1   const bcrypt = require("bcrypt");
  2   const jwt = require("jsonwebtoken");
  3   const User = require("../models/user");
  4
  5 > exports.createUser = (req, res, next) => {…
 35   };
 36
 37 > exports.isAdmin = (req, res, next) => {…
 49   };
 50
 51 > exports.userLogin = (req, res, next) => {…
 92   };
 93
```

Carpeta middleware

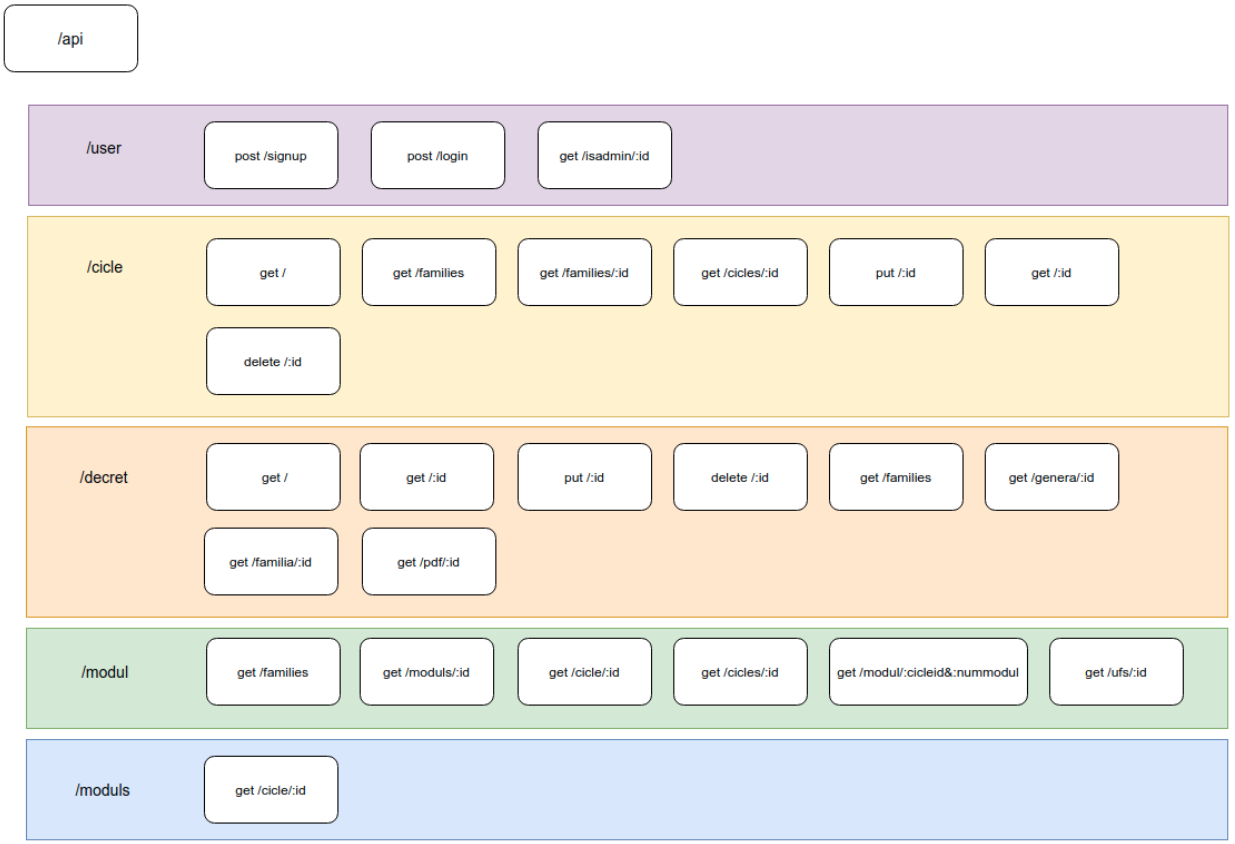
Un middleware és un mètode que apliquem a una ruta o controlador, que podem cridar des de més d'una ruta/controlador, o que la seva lògica no forma part directament del tractament de dades. Per exemple l'autentificació es fa a través d'un middleware.

Carpeta decrets

En aquesta carpeta desem en el servidor una còpia del pdf de cada decret que anem pujant.

Els arxius són una còpia del pdf que trobem a la xtec amb un identificador únic per cada arxiu.

4.6.2 Routing Backend



4.7 Frontend

4.7.1 Estructura de Carpetes del projecte Angular 2.x

Carpeta src

- carpeta app: conté els fitxers que hem creat per a components d'aplicacions.
- app.component.css: El fitxer conté el codi CSS del component de l'aplicació.
- app.component.html: El fitxer conté el fitxer HTML relacionat amb el seu component d'aplicació. Es tracta del fitxer de plantilla que és especialment utilitzat per angular per a l'enllaç de dades.
- app.component.spec.ts: Aquest fitxer és un fitxer de test unit amb l'ordre ng test.
- app.component.ts: és el fitxer typescript que inclou la lògica de visites més enllà del component.
- app.module.ts: També és un fitxer typescript on hi ha totes les dependències del lloc web. Les dades s'utilitzen per definir els mòduls necessaris que s'han importat, els components que s'han de declarar i el main.

Altres fitxers importants en Angular 2.x

- package.json: És el fitxer de configuració de npm. Inclou detalls de les dependències del nostre lloc web i dels paquets.
- angular.json: És un fitxer de configuració necessari relacionat amb la nostra aplicació angular. Defineix l'estructura de la nostra aplicació.
- .gitignore: El registre està relacionat amb el codi font git
- .editorconfig: Aquest és un fitxer estàndard que s'utilitza per mantenir la coherència en els editors de codi per organitzar alguns conceptes bàsics com ara el sagnat i els espais en blanc.

Carpeta d'assets

- per als fitxers de recursos que s'utilitzen a l'aplicació com ara imatges, locals, traduccions, etc.

4.7.2 Conceptes Angular 2.x

Component

Un component al final el que farà és controlar un tros de pantalla o de la vista. Tot el que podem veure a la pantalla ho controlen components. La lògica d'un component a dins d'una classe en Angular és el que dóna suport a uns vista interactuant amb ella a través d'un API amb propietats i mètodes. El component fa de mitjancer entre la vista a través de la plantilla i la lògica de la app on inclourem el model de dades, és a dir una espècie de controlador.

Plantilles

Les plantilles definiran la vista dels components. Són htmls i tenen la sintaxi especial d'Angular. Treballarem amb databinding i directives.

Decoradors

Amb els Decoradors (que són patrons de disseny) configurarem dinàmicament atributs/metadades de les classes i components.

Metadades

Les metadades descriuran les classes, però també descriuen relacions, per exemple si tenim un component i una plantilla la metadada serà l'encarregada de dir-li a Angular que aquest component i aquesta plantilla van junts, entre altres coses

Serveis

Els serveis són classes amb un objectiu clar, facilitar la reutilització, són un tipus d'element dins de l'arquitectura d'Angular i a través de la injecció de dependències els podrem utilitzar en altres components principals.

Providers

Els providers són serveis que ens aporten dades o funcionalitats a través dels seus mètodes. Existeixen providers/serveis propis d'Angular o els podem crear nosaltres.

Directives

Les directives són funcionalitats aplicables al DOM i als elements HTML en les plantilles d'un component. Per exemple una directiva pot servir per a controlar que un div es mostri o no, o recórrer un array a la vista (directives estructurals, estructures de control i condicionals) o fins i tot també poden servir per a interactuar amb el model de dades del component. Podem dir que són nous atributs per a aplicar a qualsevol cosa en la nostra plantilla/vista.

Interceptors

En Angular, quan fem peticions a un API, normalment perquè ens demanen una token d'autorització, o un api key, o alguna altra cosa per les capçaleres, moltes persones el que fan és que en cada servei o petició API, inclou la capçalera en els serveis, o, generen un servei general per l'API i aquí ho inclouen, però, això es pot fer més senzill usant 1 HTTP Interceptor.

I així com el seu nom mateix diu, "intercepta" la petició HTTP, la modifica (si es requereix) i llavors continua el seu camí, això és excel·lent ja que com esmento s'estalvia codi i es pot integrar per parts on hi hagi qüestions comuns i no vulguem saturar els serveis.

Com funciona?

Si ens fixem en index.html

```
src > index.html > ...
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Programa'm</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

veiem que hi ha la crida al component root, aquest és el punt de partida de la nostra aplicació. Si mirem a app.component.ts

```
src > app > TS app.component.ts > ...
import { Component, OnInit } from '@angular/core';
import { AuthService } from './auth/auth.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit{

  constructor(private _authService: AuthService) {}

  ngOnInit() {
    this._authService.autoAuthUser(); //autentificació automàtica
  }

}
```

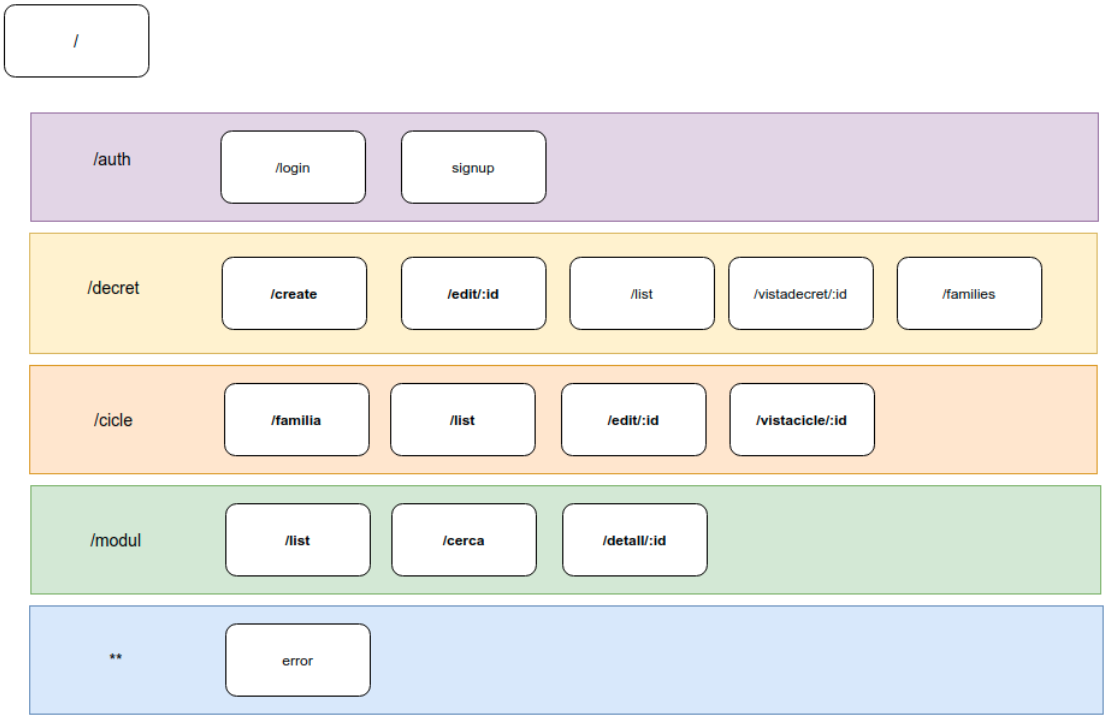
si mirem la plantilla que renderitza: app.component.html

```
src > app > <> app.component.html > ...
      Go to component
<div fxLayout="column" fxFlexFill fxLayoutAlign="center center">
  <app-header></app-header>
  <div fxFlex class="app" >
    <br/><br/><br/><br/>

      <router-outlet></router-outlet>
  </div>
  <app-footer></app-footer>
</div>
```

podem observar que es crida al component corresponent a la capçalera i al peu de pàgina, que són comuns en tot l'aplicatiu, i la crida al router-outlet, que crida a cadascun dels components definits en el router, segons l'adreça especificada.

Routing en el Frontend

Aquesta és l'estructura de rutes en el frontend.



que es defineix en l'arxiu de routing: app-routing.module.ts

src > app > TS app-routing.module.ts > ...

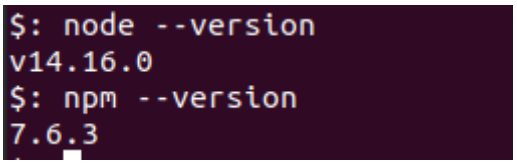
```
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { EditCicleComponent } from './cicles/editCicle/editcicle.component';
import { LlistaProgramacioComponent } from './modul/llista-modul/llista-programacio.component';
import { CercaModulComponent } from './modul/cerca-modul/cerca-modul.component';
import { HomeComponent } from './home/home-card/home.component';
import { DecretsFamiliaComponent } from './decret/decrets-familia/decrets-familia.component';
import { VistaDecretComponent } from './decret/vista-decret/vista-decret.component';
import { CiclesFamiliaComponent } from './cicles/cicles-familia/cicles-familia.component';
import { VistaCicleComponent } from './cicles/vista-cicle/vista-cicle.component';
import { DetallModulComponent } from './modul/detall-modul/detall-modul.component';




const routes: Routes = [
  { path: '', component: HomeComponent},

  { path: 'decret/create', component: CreaDecretComponent, canActivate:[AuthGuard]},
  { path: 'decret/list', component: LlistaDecretsComponent},
  { path: 'decret/vistadecret/:decretId', component: VistaDecretComponent},
  { path: 'decret/families', component: DecretsFamiliaComponent},
  { path: 'decret/edit/:decretId', component: CreaDecretComponent, canActivate:[AuthGuard]},

  { path: 'cicle/familia', component: CiclesFamiliaComponent,canActivate:[AuthGuard]},
  { path: 'cicle/list', component: CiclesFamiliaComponent,canActivate:[AuthGuard]},
  { path: 'cicle/edit/:cicleId', component: EditCicleComponent, canActivate:[AuthGuard]},
  { path: 'cicle/vistacicle/:cicleId', component: VistaCicleComponent,canActivate:[AuthGuard]},

  { path: 'modul/list', component: LlistaProgramacioComponent,canActivate:[AuthGuard]},
  { path: 'modul/cerca', component: CercaModulComponent,canActivate:[AuthGuard]},
  { path: 'modul/detall/:modulId', component: DetallModulComponent,canActivate:[AuthGuard]},

  { path: 'auth', loadChildren: () => import('./auth/auth.module').then(m => m.AuthModule)},
  { path: '**', component: PageNotFoundComponent },  // Wildcard route for a 404 page


];

@NgModule({
  imports:[RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: [AuthGuard]
})
export class AppRoutingModule {}
```

per tant l'espai que ocupa el router-outlet serà l'espai que ocuparan cadascun dels components aquí definits.

5. Implementació

5.1 Configuració de l'entorn de treball (linux Ubuntu 20.04)

Comprovem que tinguem node i el gestor de paquets de node instal·lats



sino estan instal·lats podem instal·lar-los amb la comanda

```
$ sudo apt update
```

```
$ sudo apt install nodejs
```

```
$ sudo apt install npm
```

per al backend cal instal·lar express, body-parser, cors, mongoose,

Express

```
$ npm install --save express
```

Express és un framework d'aplicacions web Node.js molt lleuger i flexible que proporciona un conjunt robust de funcions per a aplicacions web i mòbils. Amb una infinitat de mètodes d'utilitat HTTP i middleware a la nostra disposició, podem crear una API robusta de manera fàcil i ràpida. El rendiment d'Express proporciona una fina capa de funcions fonamentals d'aplicacions web, sense amagar les funcions de Node.js que ja coneixem.

body-parser

```
$ npm install --save body-parser
```

Per gestionar les sol·licituds HTTP a Express.js versió 4 i posteriors, hem d'instal·lar el mòdul de middleware anomenat body-parser.

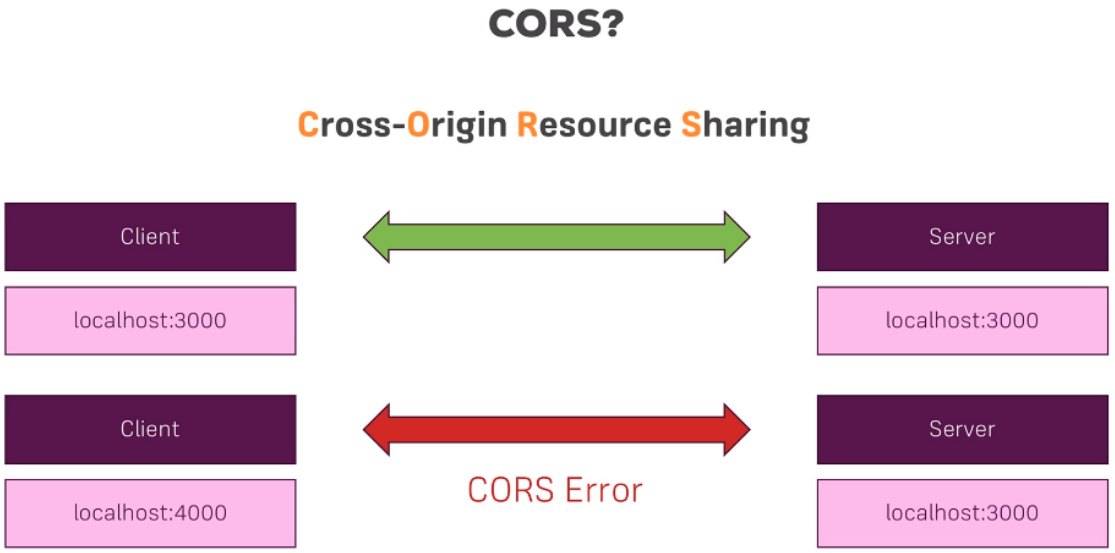
Body-parser extreu tota la part del body d'un flux de sol·licitud entrant i l'exposa a req.body, de manera que és és senzill interactuar amb la petició.

cors

```
$ npm install --save cors
```

Què són els CORS? Cross-Origin Resource Sharing

CORS: s'explica l'ús compartit de recursos entre origen. Si accedim a un lloc web, no podem d'obtenir dades de servidors tercers. Però hi pot haver excepcions. Si els dos propietaris de llocs web coincideixen en la cooperació, no hi ha res que els impedeixi arribar a un acord. L'intercanvi de recursos entre orígens (CORS) regula aquesta cooperació. Com funciona?



La same-origin policy (SOP) prohibeix que es carreguin dades de servidors aliens a l'accedir a una pàgina web. Totes les dades han de provenir de la mateixa font, és a dir, correspondre al mateix servidor. Es tracta d'una mesura de seguretat, ja que JavaScript i CSS podrien carregar, sense que l'usuari ho sabés, contingut d'altres servidors (i, amb aquest, també contingut maliciós). Tals intents són denominats "cross-origin requests". Si, per contra, els dos administradors web saben de l'intercanvi de contingut i l'aproven, no té sentit impedir aquest procés. El servidor sol·licitat (és a dir, aquell de què es vol carregar contingut) pot permetre llavors el accés mitjançant cross-origin resource sharing, és a dir, l'intercanvi de recursos d'origen creuat.

utilitzant aquesta llibreria l'usuari no s'adonarà en absolut de l'intercanvi entre tots dos servidors. Tots els navegadors actuals suporten el CORS, i l'enviament de sol·licituds i respostes succeeix ràpidament a l'sol·licitar una pàgina web sense que l'usuari ho noti.

mongoose

```
$ npm install --save mongoose
```

Mongoose és un Object Document Mapper (ODM). Això vol dir que Mongoose ens permet definir objectes amb un esquema que s'assigna a un document MongoDB. Mongoose proporciona una quantitat increïble de funcionalitats al voltant de la creació i del treball amb esquemes. A més d'aquestes opcions habituals, certs tipus de dades us permeten personalitzar encara més com s'emmagatzemen i es recuperen les dades de la base de dades.

Altres biblioteques necessàries de backend

- `bcrypt - encriptació del password`
- `jsonwebtoken -autentificació d'usuari`
- `DownloaderHelper - descàrrega d'arxiu`
- `pdfUtil - conversor de pdf a text`
- `uniqid - generardor d'ids únics per al nom d'arxiu`
- `path - per la ruta de l'arxiu, no cal instal·lar-la`

- fs - per al tractament d'arxius, no cal instal·lar-la

```
$ npm install --save nomBiblioteca
```

app.js

que és el que engega el backend queda com es mostra:

```
backend > JS app.js > ...
1   //  per permetre l'acces a una carpeta modificar el path
2   const path = require('path');
3
4   const express = require('express');
5   const bodyParser = require('body-parser');
6   const cors = require('cors');
7   const mongoose = require('mongoose');
8
9   const userRoutes = require("./routes/user");
10  const decretRoutes = require("./routes/decret");
11  const cicleRoutes = require("./routes/cicle");
12  const modulsRoutes = require("./routes/moduls");
13  const modulRoutes = require("./routes/modul");
14
15  const app = express(); // big chain middle ware
16  app.use(bodyParser.json());
17  app.use(bodyParser.urlencoded({ extended: false }));
18
19  // donem access a la carpeta (images)
20  app.use("/images", express.static(path.join("backend/images")));
21
22  // Arreglar els CORS
23  app.use(cors());
24
25  const port = 3700;
26
27  mongoose.set('useFindAndModify', false);
28
29  mongoose.Promise = global.Promise;
30  var options = {
31      useNewUrlParser:true,
32      useUnifiedTopology: true
33
34  }
35  // he creat un bd en local a mongo amb
36  // si la BBDD noexisteix la crea
37  // mongoose.connect('mongodb://localhost:27017/programam', options)
38  //     .then(()=> {
39  //         console.log("Connexio a la BD programam establerta correctament ");
40
41  //     })
42  //     .catch (err => console.log(err));
43
44  mongoose.connect('mongodb+srv://pilar:32pBAgd55ydkvwg@cluster0.5tug6.mongodb.net/myFirstDatabase?retryWrites=true&w=majority', options)
45      .then(()=> {
46          console.log("Connexio a la Mongo Atlas BD programam establerta correctament ");
47
48      })
49      .catch (err =>  {
50          console.log("No hi ha connexió a Mongo Atlas");
51          res.status(500).json({
52          message: ':-( BD no disponible'
53          });
54      });
55
56
57  // Backend Usuari
58  app.use("/api/user",userRoutes);
59  app.use("/api/decret",decretRoutes);
60  app.use("/api/cicle",cicleRoutes);
61  app.use("/api/moduls",modulsRoutes);
62  app.use("/api/modul",modulRoutes);
63
64  module.exports =  app;
65
```

Instal·lar angular i crear una app basada en Angular

Primer instal·lem el cli que és l'eina de comandes en línea per a crear, depurar i publicar aplicacions Angular https://cli.angular.io/

```
$sudo npm install -g @angular/cli
```

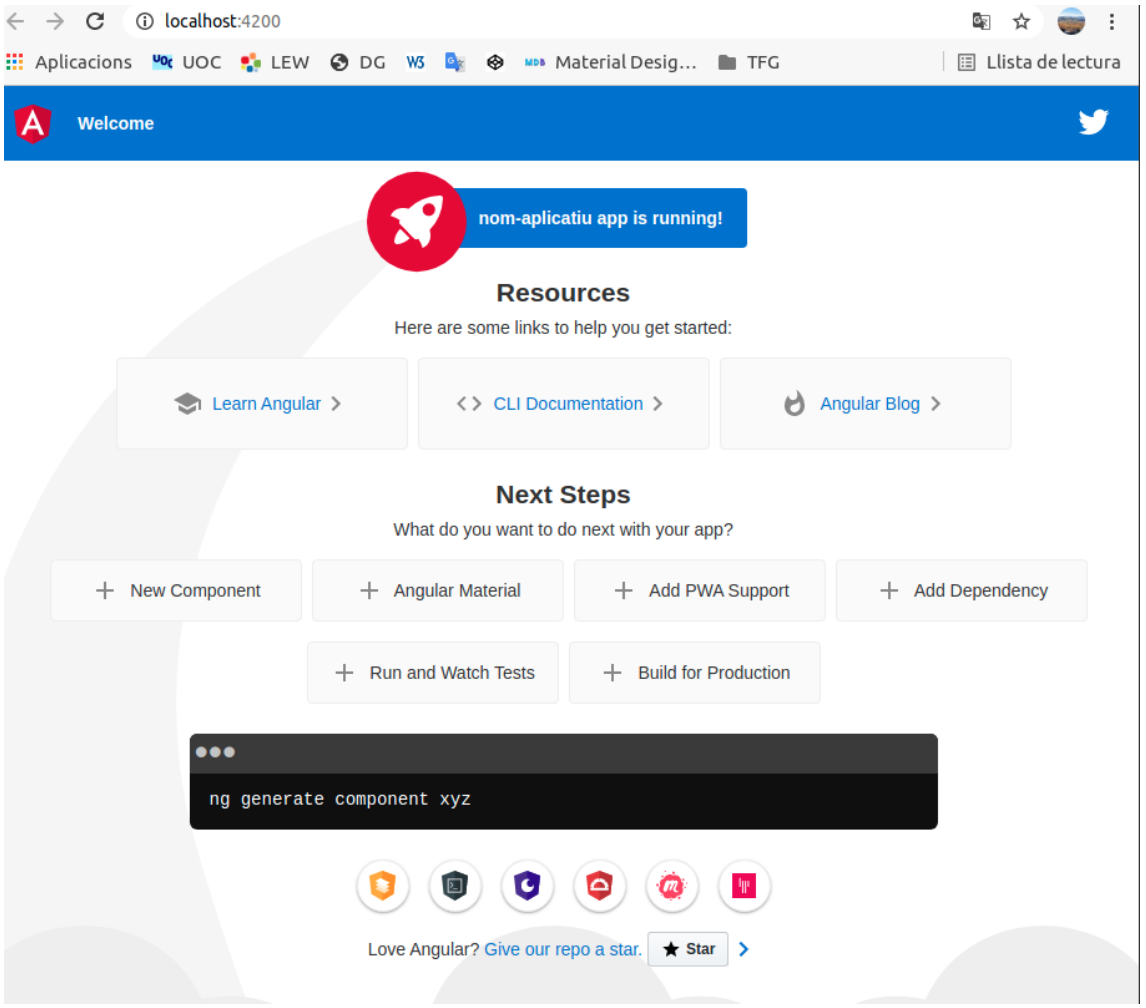
ens situem en la carpeta on volem desenvolupar l'app

```
$ng new nom-aplicatiu
```

sense routing i amb estils css: per exemple

```
$cd nom-aplicatiu
```

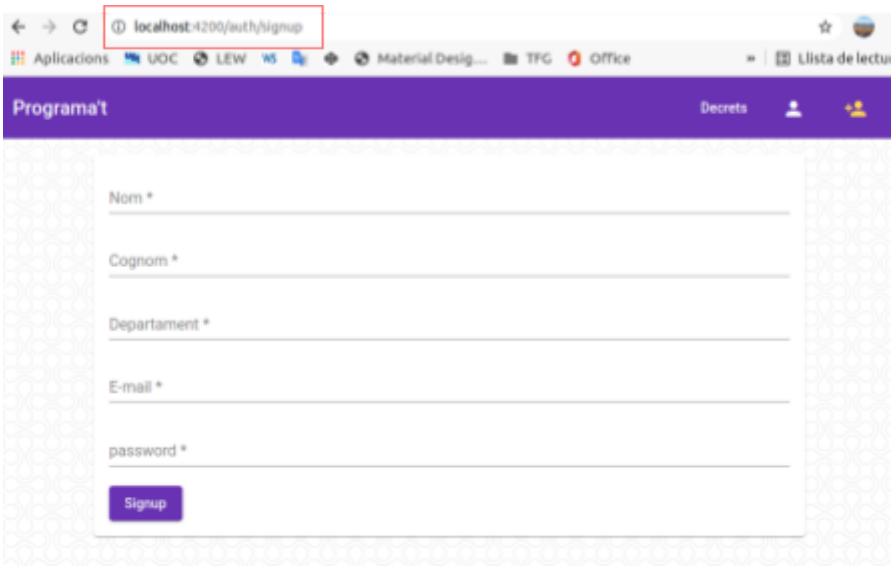
```
$ng serve
```



5.2 Usuaris

5.2.1 Registre i exemple de qualsevol petició al backend

Si anem a la ruta registre ens mostra la vista del formulari de registre



El formulari crida al mètode del component onSignup quan fem el submit amb totes les dades del formulari

```
src > app > auth > signup > TS signup.component.ts > ...
1   import { Component, OnDestroy, OnInit } from '@angular/core';
2   import { NgForm } from '@angular/forms';
3   import { Subscription } from 'rxjs';
4   import { AuthService} from '../auth.service';
5
6   @Component({
7     selector: 'app-signup',
8     templateUrl: './signup.component.html',
9     styleUrls: ['./signup.component.css']
10  })
11  export class SignupComponent implements OnInit, OnDestroy {
12
13    isLoading = false;
14    private authStatusSub: Subscription;
15
16    constructor(public _authService:AuthService) { }
17
18    ngOnInit(): void {
19      this.authStatusSub = this._authService.getAuthStatusListener().subscribe(
20        authStatus => {
21          this.isLoading =false;
22        }
23      );
24    }
25
26    onSignup(form: NgForm) {
27      //console.log(form.value);
28      if (form.invalid) {
29        return
30      }
31      this.isLoading=true;
32      this._authService.createUser(
33        form.value.email,
34        form.value.password,
35        form.value.nom,
36        form.value.cognom,
37        form.value.departament );
38
39    }
40
```

una vegada validat el formulari, es fa una crida al mètode createUser del servei AuthService que hem injectat en el constructor, passant-li les dades del formulari.

En el servei fem una petició per post a la ruta del backend /api/user/signup passant-li les dades de l'usuari nou que es vol registrar

```
createUser(
  email: string,
  password: string,
  nom: string,
  cognom: string,
  departament: string,
) {
  const authData: AuthData = {
    email: email,
    password: password,
    nom: nom,
    cognom: cognom,
    departament: departament,
    role: ''
  }
  console.log(authData);
  return this._http
    // aquest és el format de la resposta que ens vindrà del backend
    .post(
      BACKEND_URL + 'signup',
      authData
    )
    .subscribe((responseData) => {
      //console.log("entrem");
      this._router.navigate(["/"]);
    }, error => {
      this.authStatusListener.next(false);
    });
}
```

en el backend hem definit la ruta:

```
// framework d backend
const express = require("express");
// el router
const router = express.Router();
const checkAuth = require('../middleware/check-auth');


// importem el controllador
const UserController = require("../controllers/user");

// assignem rutes a mètodes del controllador
router.post("/signup", UserController.createUser);
router.post("/login",UserController.userLogin);
router.get("/isadmin/:id",checkAuth,UserController.isAdmin);

module.exports = router;

```

que crida al mètode createUser del controlador UserController:

backend > controllers > JS user.js > ...

```
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const User = require("../models/user");

exports.createUser = (req, res, next) => {
  // fem hash al pw
  bcrypt.hash(req.body.password, 10).then((hash) => {
    //console.log(req.body);
    const user = new User({
      email: req.body.email,
      password: hash,
      nom: req.body.nom,
      cognom: req.body.cognom,
      departament: req.body.departament,
      role: "user",
    });

    user
      .save()
      .then((result) => {
        res.status(201).json({
          message: "Usuari creat correctament",
          result: result,
        });
      })
      // capturem l'error d'usuari existent
      .catch((err) => {
        console.log(err);

        res.status(500).json({
          message: "Dades incorrectes!!",
        });
      });
  });
};
```

en aquest mètode per una banda importem la llibreria bcrypt que ens permetrà desar les dades del password encriptades, i el model usuari:

backend > models > JS user.js > ...

```
1   const mongoose = require('mongoose');
2   const uniqueValidator = require('mongoose-unique-validator');
3
4   // plantilla que  li passem un obkj javascript
5   // hi hem de definir els camps i el q contenen
6
7   const userSchema = mongoose.Schema({
8     email: { type: String, required: true, unique:true},
9     password: { type: String, required: true},
10    nom: { type: String, required: true},
11    cognom: { type: String, required: true},
12    departament: { type: String, required: true},
13    role:{type:String,required:true}
14
15  });
16
17  userSchema.plugin(uniqueValidator);
18
19  // es crea el model
20
21  module.exports = mongoose.model('User',userSchema);
22
```

en el model importem mongoose i mongoose unique validator, i tenim definit l'esquema de la base de dades.

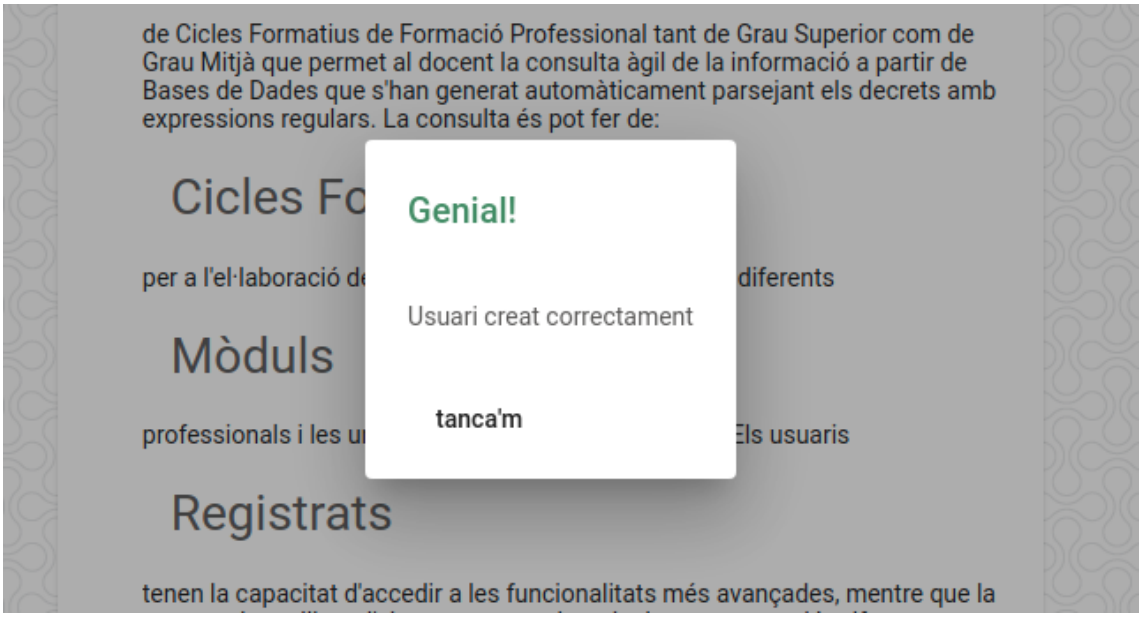
En el controlador fem un nou objecte user, en el qual s'encripta la contrasenya, i finalment cridem al mètode save que desa el document a la base de dades.

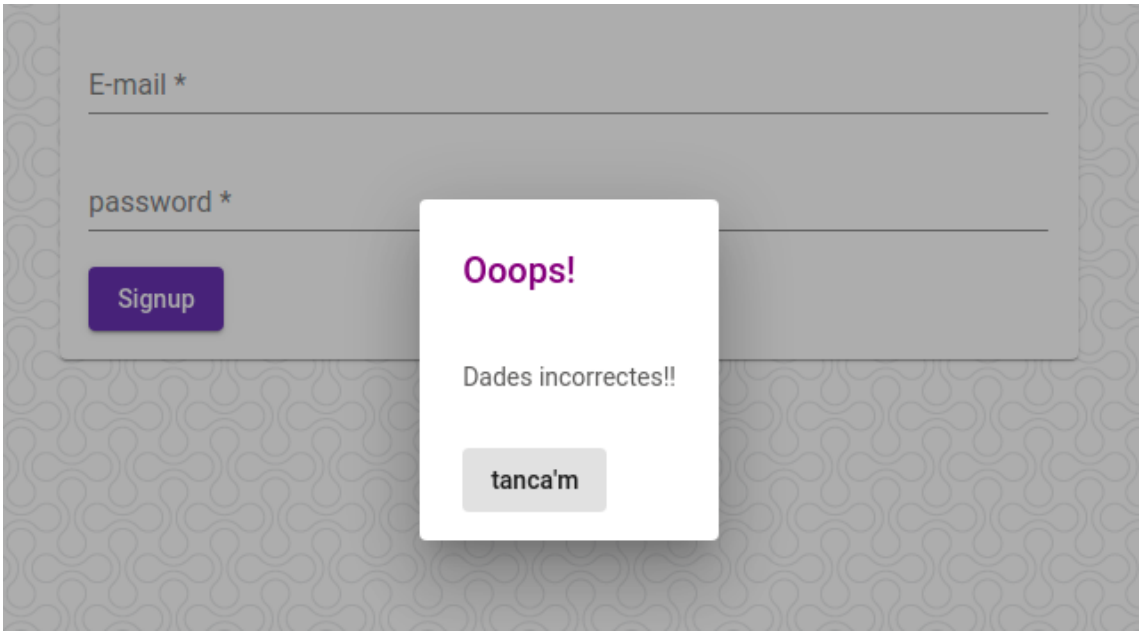
El mètode save és una promesa amb 2 possibles respostes, el resultat o l'error, en ambdós casos s'envia la informació amb una resposta 201 si tot ha anat bé o un error 500 amb format de JSON que té un missatge i la resposta.

Aquesta resposta es captura al frontend per un interceptor que està escoltant totes les respostes:

```
@Injectable()
export class ErrorInterceptor implements HttpInterceptor {

  constructor(private _dialog:MatDialog){}

  intercept(req: HttpRequest<any>, next:HttpHandler) {

    return next.handle(req).pipe(
      tap((event: HttpEvent<any>) => {
        if (event instanceof HttpResponse && event.status === 201) {
          const missatge = event.body.message;
          console.log("caçat");
          console.log(missatge);
          // console.log(event.body.message);
          this._dialog.open(SuccessComponent, {data: {message: missatge}});
        }

      }),
      catchError((error: HttpErrorResponse) => {
        console.log(error);
        let errorMessage = "Hi ha hagut un error";
        if (error.error.message) {
          errorMessage = error.error.message;

        }
        this._dialog.open(ErrorComponent, {data: {message: errorMessage}});
        //alert(error.error.error.message);
        return throwError(error);
      })
    );

  }
```

que mostra el component quadre de dialeg _dialog que hem injectat amb un missatge de tot ok, o bé d'error, segons la resposta.





aquest sistema de gestió de missatges d'èxit i errors basat en interceptor és molt eficient, net i que serveix per a qualsevol petició al backend.

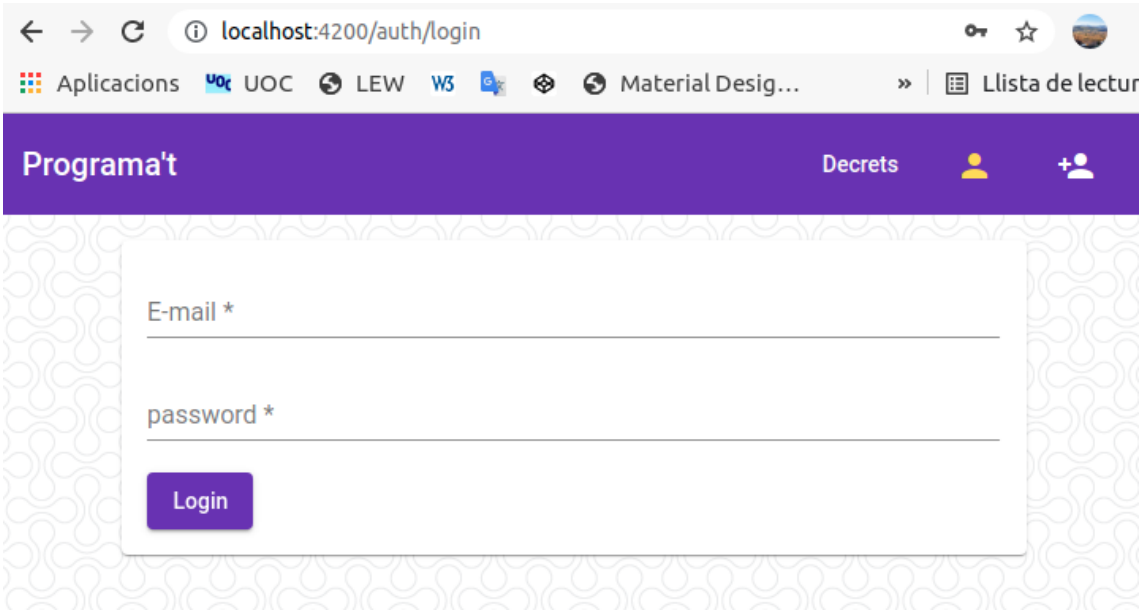
Finalment :

```
    .subscribe((responseData) => {
      //console.log("entrem");
      this._router.navigate(["/"]);
    }, error => {
      this.authStatusListener.next(false);
    });
```

amb el mètode subscribe en el mateix servei, redirigim a la ruta "/" o bé fem un next false, per aturar el subscribe.

5.2.2 Autentificació

Formulari de Login, procedim de manera similar:



mètode del component login

```
30  onLogin(form: NgForm) {
31    if (!form.valid) {
32      return;
33    }
34    this.isLoading=true;
35    this._authService.login(form.value.email, form.value.password);
36    //console.log(form.value);
37  }
```

mètode del servei AuthService login:

```
99   login(
100    email: string,
101    password: string,
102
103  ) {
104    const authData: AuthData = {
105      email: email,
106      password: password,
107      nom: "",
108      cognom: "",
109      departament: "",
110      role: "",
111    };
112    this._http.post<{ token: string, expiresIn: number, userId: string, role: string }>
113      (BACKEND_URL + "login", authData)
114      .subscribe(response => {
115        //console.log(response);
116        const token = response.token;
117        if (token) {
118          this.token = token;
119          this.expiresInDuration = response.expiresIn;
120          this.setAuthTimer(this.expiresInDuration);
121          //console.log(expiresInDuration);
122
123          this.authStatusListener.next(true);
124          this.isAuthenticated = true;
125          this.userId = response.userId;
126          this.role = response.role;
127
128          const now = new Date();
129          const expirationDate = new Date(now.getTime() + this.expiresInDuration * 1000);
130          //console.log(expirationDate);
131          this.saveAuthData(token, expirationDate, this.userId, this.role);
132          this._router.navigate(['/']);
133
134        }
135
136      }, error => {
137        this.authStatusListener.next(false);
138      })
139  }
```

en aquest cas enviem les dades d'autorització (email, i password) per post a la ruta del backend /api/user/login i esperem rebre, si l'autentificació és correcta un token, que desarem al localStorage durant una hora, i que enviarem en els headers en qualsevol petició al backend, per fer-ho utilitzem de nou un interceptor.

src > app > auth > TS auth-interceptor.ts > ...

```
1  import { HttpInterceptor, HttpRequest, HttpHandler } from "@angular/common/http";
2  import { Injectable } from "@angular/core";
3  import { AuthService } from "./auth.service";
4
5  @Injectable()
6  export class AuthInterceptor implements HttpInterceptor {
7
8    constructor(private _authService: AuthService) {}
9    intercept(req: HttpRequest<any>, next:HttpHandler) {
10
11     const authToken = this._authService.getToken();
12     const authRequest = req.clone({
13       headers: req.headers.set('Authorization',"Bearer "+authToken)  // headers
14     });
15     return next.handle(authRequest);
16
17   }
18 }
19
```

que afegeix el token als headers de qualsevol petició HttpRequest.

En el backend, per una banda en el procés del login:

```
exports.userLogin = (req, res, next) => {
  let fetchedUser;
  User.findOne({ email: req.body.email })
    .then((user) => {
      if (!user) {
        return res.status(401).json({
          message: ":-( Dades d'autentificació incorrectes",
        });
      }

      fetchedUser = user;
      return bcrypt.compare(req.body.password, user.password);
    })
    .then((result) => {
      if (!result) {
        return res.status(401).json({
          message: ":-( Dades d'autentificació incorrectes",
        });
      }
      // JWT crear token per autentificació
      const token = jwt.sign(
        {
          email: fetchedUser.email,
          userId: fetchedUser._id,
        },
        process.env.JWT_KEY,
        //'secret_this_should_be_longer',
        { expiresIn: "1h" }
      ); //1h de validesa d token
      res.status(200).json({
        token: token,
        expiresIn: 3600, // caduca al cap d'una hora
        userId: fetchedUser._id,
        role: fetchedUser.role,
      });
    })
    .catch((err) => {
      return res.status(401).json({
        message: ":-( Dades d'autentificació incorrectes",
      });
    });
```

comprovem les dades, primer busquen en usuaris un usuari amb l'email que hem rebut del frontend, sinó en trobem cap retornem error, si hi ha l'email a la BBDD comprovem amb bcrypt que el password sigui el mateix que tenim encriptat a la BBDD, sinó és així retornem error, i si és correcte, generem el token utilitzant la biblioteca jsonwebtoken, que genera un token únic, per aquella combinació de email i userid, aquest token és el que es retorna al frontend per tal que sigui emmagatzemat en el localstorage i s'enviï en qualsevol petició al backend.

Per altra banda al backend tenim un middleware que ens permet comprovar que l'usuari estigui autentificat per segons quines rutes, és a dir que el token que ha afegit l'interceptor a les peticions sigui correcte per aquella combinació d'usuariId email.

per una banda en les rutes del backend:

```
11  // assignem rutes a mètodes del controllador
12  router.post("/signup", UserController.createUser);
13  router.post("/login",UserController.userLogin);
14  router.get("/isadmin/:id",checkAuth,UserController.isAdmin);
15
```

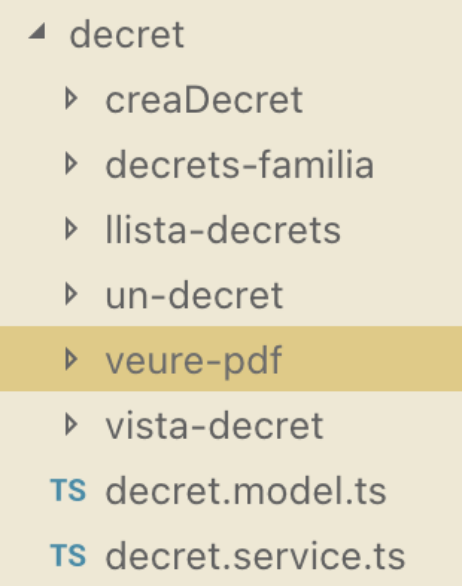
i el middleware:

```
backend > middleware > JS check-auth.js > ...
1   // comprovar q tenim un token
2   // q el token es valid
3
4   const jwt = require('jsonwebtoken');
5
6   module.exports = (req,res,next) => {
7     try {
8     //const token = req.query.auth
9       const token = req.headers.authorization.split(" ")[1];
10      const decodedToken = jwt.verify(token, "secret_this_should_be_longer");
11      req.userData = {email: decodedToken.email, userId: decodedToken.userId};
12      next();
13    //"Bearer akjsdhsjkadajk"
14    } catch (error) {
15      res.status(401).json({message:"No estàs autenticat!"});
16    }
17
18  }
```

que fa la verificació utilitzant la biblioteca jsonwebtoken, en cas que autentificació no sigui correcta s'envia una resposta d'error. Com abans és l'interceptor del frontend que mostrarà el missatge d'error a l'usuari.

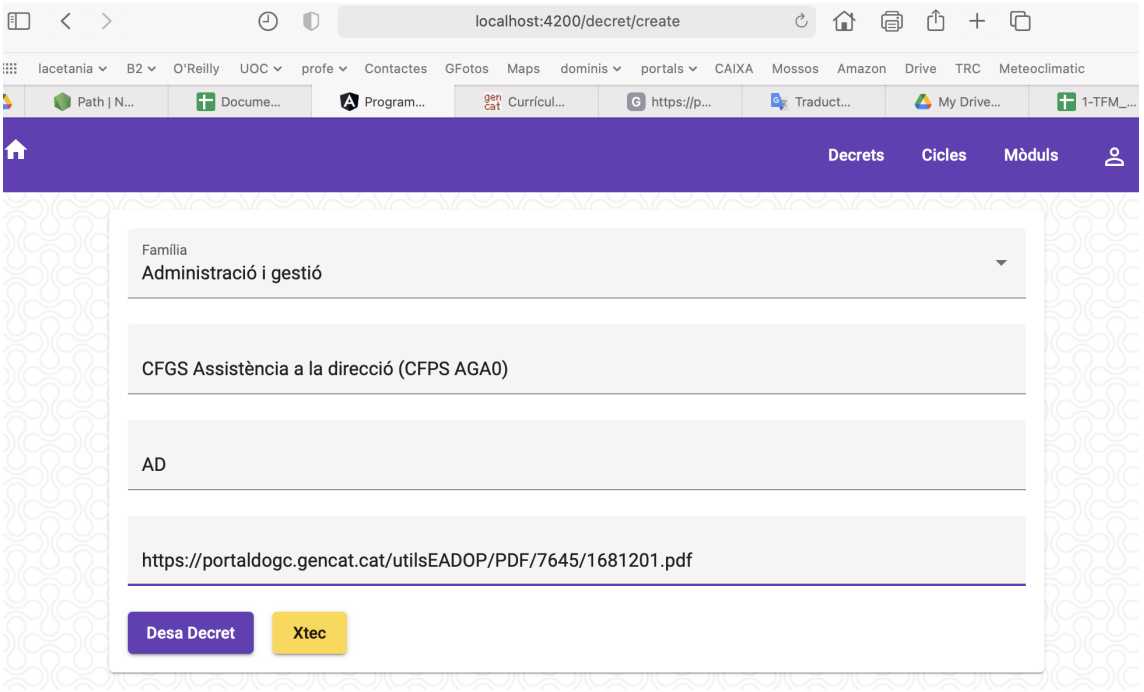
5.3 Decrets

Aquí tenim un llistat dels components Frontend, a part el model decret.js (equivalent al model dissenyat en mongoose), i decret.service.ts que és l'arxiu que farà la comunicació amb la base de dades.



Pujar decrets

Per pujar decrets a la base de dades cal tenir els permisos d'administrador.



En el frontend, a través d'un formulari pugem les dades del decret, en especial el nom complert i l'enllaç que com s'indica en l'apartat 4.5 d'aquesta memòria traiem de la web de l'xtec. El procediment és anàleg a donar d'alta un usuari.

Un cop arriben les dades del formulari al backend, seran processades pel mètode addDecret del controlador decret.js. Primer comprovem que d'enllaç correspongui a un arxiu pdf utilitzant una expressió regular:

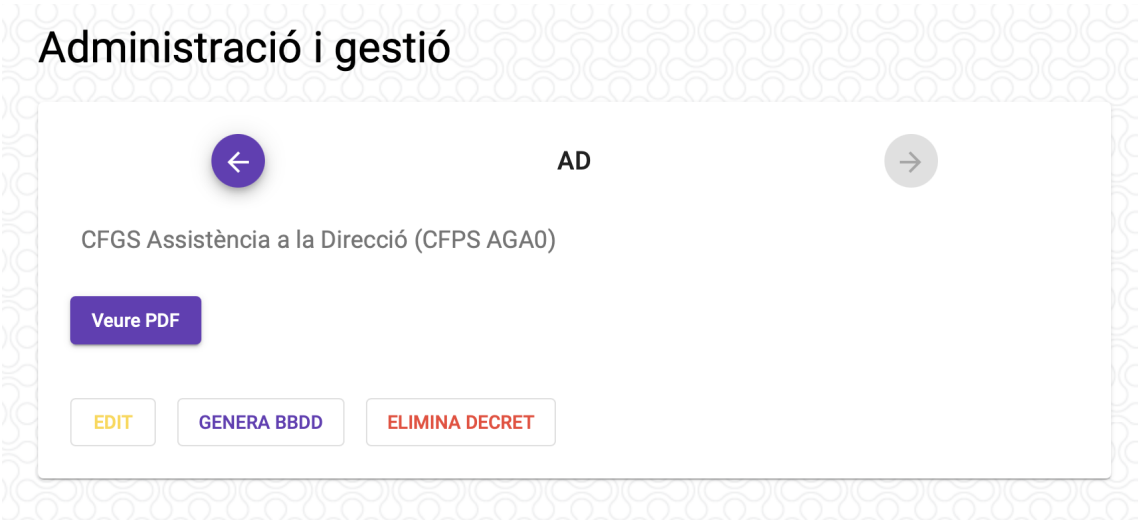
```
const regex = /http[s]*:\/\/.+\/(.+(\.pdf))/gm;
const match = regex.exec(linkPdf);
if (!match[1]) {
  res.status(401).json({ message: "L'enllaç no correspon a un pdf" });
}
```

```
const ID = uniqid();

const dl = new DownloaderHelper(linkPdf, __dirname + "/../decrets", {
  fileName: ID + ".pdf",
});

dl.on("end", () => {
  const url = req.protocol + "://" + req.get("host");
  const nomArxiu = url + "/api/decret/pdf/" + ID + ".pdf";

  // http://localhost:3000/api/decret/pdf/pdfsldkfjsld

  const decret = new Decret({
    familia: req.body.familia,
    nomCicle: req.body.nomCicle,
    enllasDecret: req.body.enllasDecret,
    nomLlarg: req.body.nomLlarg,
    nomArxiu: nomArxiu,
    bdCicle: false,
  });

  // console.log(decret);

  decret
    .save()
    .then((result) => {
      res
        .status(201)
        .json({ message: "Decret afegit correctament", result: result });
    })
    // capturem l'error d'usuari existent
    .catch((err) => {
      // console.log(err);
      res.status(500).json({ message: "Hi ha hagut un problema" });
    });
});
```

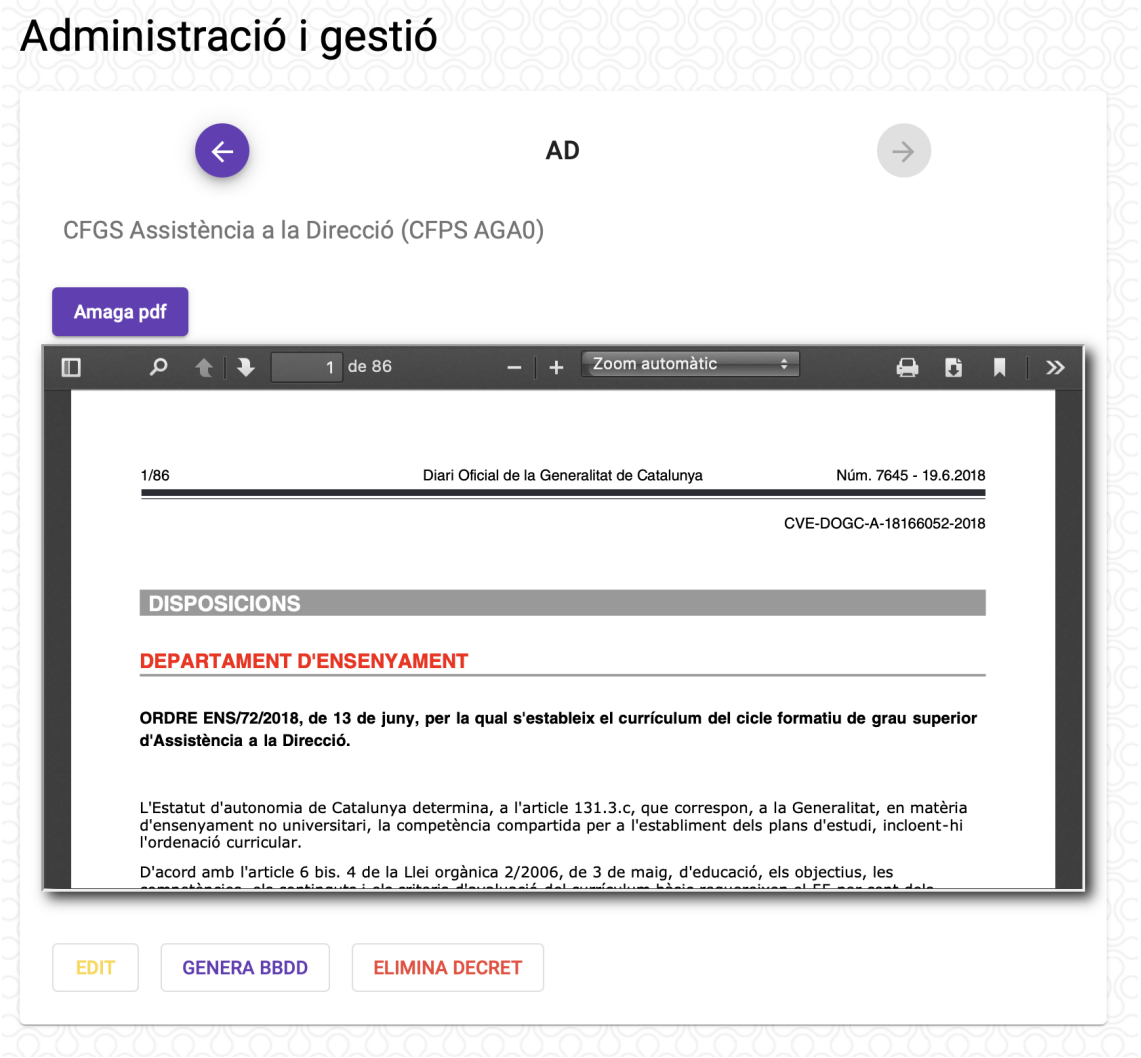
Generem un ID únic per desar la'rxiu al backend, utilitzant la biblioteca uniqId i utilitzem la biblioteca DownloaderHelper per baixar l'arxiu enllaçat, si aconseguim baixar l'arxiu generem un objecte nou Decret i el desem a la base de dades.

Visualitzar Decrets

En el frontEnd hem utilitzat la biblioteca ng2-pdfjs-viewer que hem hagut d'instal·lar amb el gestor de paquets npm tal i com hem vist en l'apartat anterior. Angular i el seu sistema de components ens permet molta modularitat i definir cada part amb un component diferent.

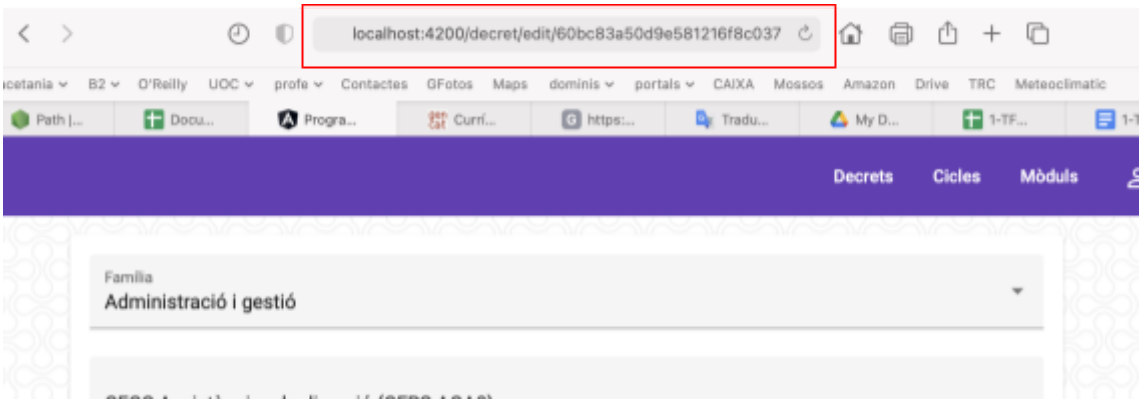


Les fletxes ens permeten la navegació pels diferents decrets de la mateixa família. L'usuari administrador pot editar el decret, i generar la base de dades associada. El botó veure pdf ens mostra el visualitzador de pdf



el visualitzador és bastant complert, permetent la descàrrega, impressió, el zoom, la navegació ràpida per les pàgines etc.

L'editor del decret funciona de la següent manera



fem una petició al backend buscant les dades del decret indicat a la barra d'adreces, i omple el formulari amb aquestes dades, si tot està correcte, el botó desa decret fa una petició put la backend per actualitzar el document de la base de dades.

En cas d'eliminar el decret es fa la petició delete al backend, que crida al mètodes corresponents

al servei decret.service.ts :

```
93    getDecret(_id: string) {
94      return  this._http.get<{
95        _id:string,
96        familia:string,
97        nomCicle: string,
98        enllasDecret: string,
99        nomLlarg:string,
00        nomArxiu:string,
01        bdCicle:boolean,
02      }>(BACKEND_URL+_id);
03
04    }
05    // actualitzar un post
06    updateDecret(decret: Decret) {
07      this._http
08        .put(BACKEND_URL + decret._id, decret)
09        .subscribe((response) => {
10          this._router.navigate(["/"]);
11        });
12    }
13    // eliminar decret
14    deleteDecret(_id: string) {
15      // en el servei fem la crida al backend i li passem l'iD
16      return this._http
17        .delete(BACKEND_URL + _id)
18
19    }
```

al backend:

```
15    router.get('/:id',DecretController.getOneDecret);
16    router.put("/:id", checkAuth, DecretController.updateDecret);
17    router.delete("/:id",checkAuth,DecretController.deleteDecret);
```

5.4 Generar la Base de Dades

Fins aquí l'aplicació no fa res de diferent de moltes aplicacions web, gestió d'usuaris, amb registre i autentificació, afegir/modificar/eliminar decrets a la base de dades, i consulta d'aquest decrets amb un visualitzador pdf. Però la idea de l'aplicació va sorgir perquè he observat que els documents pdf de cada decrets són bastant semblants els uns amb els altres. Segueixen una plantilla comuna, d'aquí ve la idea de parsejar aquests decrets per obtenir una consulta àgil de les dades necessàries per l'elaboració de programacions didàctiques.

Com funciona aquest procés?

Per generar la base de dades cal tenir permisos d'administrador. En aquest cas apareix el botó Genera BBDD

Aquest botó

Crida al mètode onGeneraJSON, crida al mètode del servei generaBBD que alhora fa una petició al backend, concretament a la ruta /api/decret/genera, i li passem la id del decret que volem generar la BBDD.

```
onGeneraJSON(_id: string) {
  this._decretService.generaBBDD(_id).subscribe(res => {
    console.log(res);
    this.decret.bdCicle = true;
    this.isLoading = false;
    this._router.navigate(['/cicle/list']);

    // console.log(this.decret);

  });
}
```

```
69      generaBBDD(_id:string) {
70        // console.log("service");
71        // console.log(BACKEND_URL+'genera/'+_id);
72        return this._http.get<any>(BACKEND_URL+'genera/'+_id);
73
74      }
```

En el backend

```
router.get("/genera/:id",DecretController.generaBBDD);
```

cridem al mètode generaBBDD de DecretController, que rep la id del decret per paràmetre. Passos que segueix aquest mètode:

- Cerca el document decret amb aquesta id
- Comprova que el nom d'arxiu correspon a un pdf
- Converteix el pdf a text utilitzant la biblioteca pdfToText
- Parseja amb les funcions `parseDecret` i `creaCicleDocument` el resultat per crear el document cicle associat
- Si ho aconsegueix, parseja de nou el text (veure el punt següent `parseDecretModul`) per generar els documents mòdul associats al decret, i marca la propietat `bdCicle` a cert, indicant que ja s'ha generat la BBDD associada a aquest decret. Finalment retorna la resposta al frontend indicant l'èxit o fracàs de l'operació.

parseDecret

- Paràmetres: contingut de l'arxiu pdf en format txt
- sortida: contingut de l'arxiu en format txt net sense capçaleres, peus de pàgina i control de salts de línia

```
parseDecret = (data) => {
  console.log("parse");

  const regexCapsa = /\d+\/\d+.+Núm.+[+\d]/gm;
  data = data.replace(regexCapsa, "");

  const regexSaltsLinia = /\n{2,}/gm;
  data = data.replace(regexSaltsLinia, "\n");

  const regexCapsa2 = /\n\s+.+DOGC.+\n/gm;
  data = data.replace(regexCapsa2, "");

  const regexLiniesMalSaltades = /([a-zàèéíòóú,])(\n)([A-Za-z])/gm;
  data = data.replace(regexLiniesMalSaltades, "$1 $3");

  const regexPeu = /ISSN.+-2\d\d\d/gm;
  data = data.replace(regexPeu, "");

  //console.log(data.slice(0,10000)); //print all text

  return data;
};
```

creaCicleDocument

- paràmetres:
  - objecte decret relacionat
  - la sortida de parseDecret
- sortida
  - objecte cicle que convertirem en document a la BBDD
  - o bé false si hi ha algun problema en la generació de l'objecte

Com a exemple veiem com funciona la primer expressió regular:

```
const regexIdentificacioTitol =
  /Identificació.+\n?.*1.1\s?Denominació:\s?(.+).*\n?.+Nivell:\s?(.+).*\n?.+Durada:\s?(\d\.?\d+).*\n?.+professional:\s?(.+).*\n?.+europeu:\s?(.+)/gm;
```

Les dades s'extreuen d'aquesta part del document:

Annex

1. Identificació del títol

1.1 Denominació: Assistència a la Direcció

1.2 Nivell: formació professional de grau superior

1.3 Durada: 2.000 hores

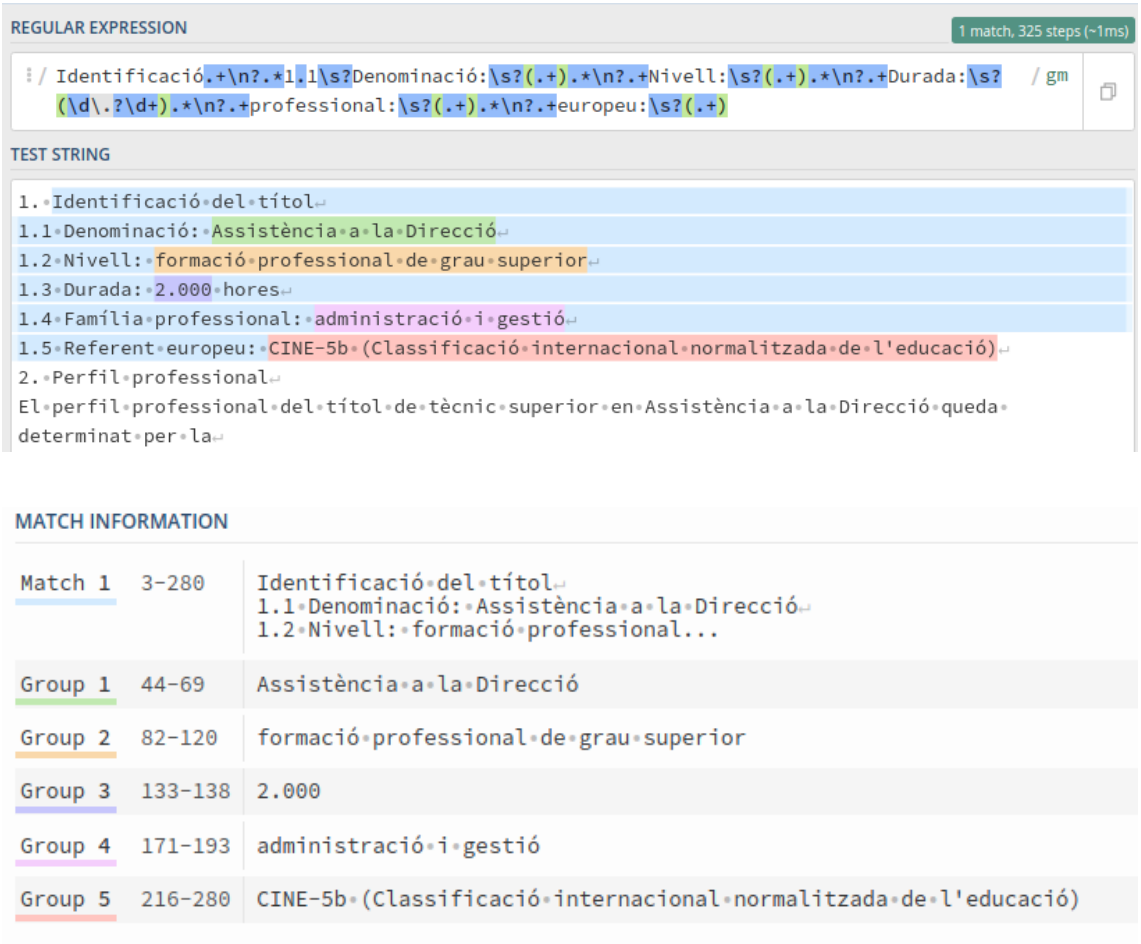
1.4 Família professional: administració i gestió

1.5 Referent europeu: CINE-5b (Classificació internacional normalitzada de l'educació)

El codi que ens ho proporciona

```
creaCicleDocument = (decret, data) => {
  // console.log(data.slice(0, 400));
  const regexIdentificacioTitol =
    /Identificació.+\n?.*1.1\s?Denominació:\s?(.+).*\n?.*Nivell:\s?(.+).*\n?.*Durada:\s?(\d\.?\d+).*\n?.+professional:\s?(.+).*\n?.+europeu:\s?(.+)/gm;
  const dadesIdentificacioTitol = regexIdentificacioTitol.exec(data);
  const codiCicleRegex = /\((.+)\)/gm;
  const codiCicle = codiCicleRegex.exec(decret.nomLlarg)[1];
  const regexCompetenciaGeneral = /Competència general[\s\S]+?\d+\.\d+/gm;
  const competenciaGeneralR = regexCompetenciaGeneral.exec(data);
  const competenciaGeneral = competenciaGeneralR[1];

  if (
    !dadesIdentificacioTitol ||
    dadesIdentificacioTitol.length < 4 ||
    !codiCicle ||
    !competenciaGeneralR
  ) {
    return false;
  }
  let durada = dadesIdentificacioTitol[3].replace(".", "");
  durada = parseInt(durada);
  const nomCicleCurt = decret.nomCicle;

  const cicle = new Cicle({
    _idDecret: decret._id,
    nomCicleCurt: nomCicleCurt,
    codiCicle: codiCicle,
    competenciaGeneral: competenciaGeneral,
    identificacioTitol: {
      nom: dadesIdentificacioTitol[1],
      nivell: dadesIdentificacioTitol[2],
      durada: durada,
      familia: decret.familia,
      referentEuropeu: dadesIdentificacioTitol[5]
        ? dadesIdentificacioTitol[5]
        : "-",
    },
  });
  return cicle;
};
```

si observem com treballa l'expressió regular, fa match de tota

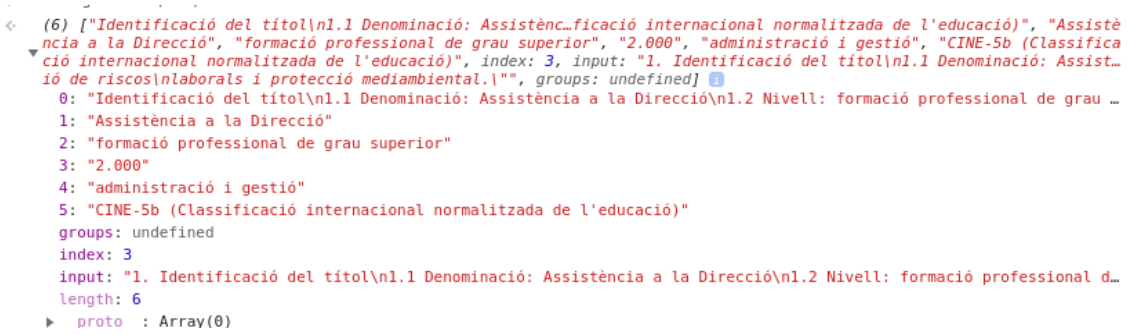


La idea és fer match de tot el punt 1, i extreure utilitzant grups cada part d'informació. En general sempre es treballa igual, buscar quelcom que fa que aquesta part sigui diferent de la resta, i analitzar-la, per tal de crear una expressió regular que extreu les dades que interessen per obtenir les dades de l'objecte cicle que finalment seran les dades del document de la base de dades corresponent. Examinem en detall la primera part de l'expressió per entendre el funcionament.

```
Identificació.+\n?.*1.1\s?Denominació:\s?(.+).*\n
```

La subcadena que busquem comença per Identificació, seguida d'almenys un caràcter, un salt de línia (opcional), després qualsevol nombre de caràcters, la cadena 1 qualsevol caràcter 1 i un espai opcional, més la cadena Denominació: un espai opcional, i trobem el primer grup (.+).*\n que són tots els caràcters fins el salt de línia (comportament greedy).

El resultat d'executar l'expressió sobre el text és un array format pels següents camps:



On l'element 0 és el match sencer, i a partir de l'element 1 fins el 5 el contingut de cadascun dels grups.

Un cop vist aquest exemple, les dades es van parsejant de manera similar. Comprovant en cada cas que el resultat és l'esperat, en aquest cas, la comprovació és la següent:

```
if (
  !dadesIdentificacioTitol ||
  dadesIdentificacioTitol.length < 4 ||
  !codiCicle ||
  !competenciaGeneralR
) {
  return false;
}
```

parseDecretModul

- Paràmetres:
  - la sortida de parseDecret
  - cicleId que és l'identificador del document cicle relacionat
- Sortida:
  - booleà indicant si ha tingut èxit o no en la generació de cadascun dels documents mòdul associats al cicle formatiu.

El funcionament és similar a creaCicle, però amb una mica més de complexitat, els passos que se segueixen són els següents:

- Per tal de facilitar la separació de dades entre mòduls, substitueixo la cadena `"Mòdul"` per `"#####Mòdul"`, de manera que podem delimitar on comença un mòdul i on acaba. Comença per “Mòdul” i acaba per “#####”.
- Extraiem totes les coincidències, hauria de ser una per cada mòdul i les posem en un array

```
const regexModul3 = /Mòdul([\s\S]+?)#####/gm;
const arrayModuls = regexModul3.exec(moduls);

while ((m = regexModul3.exec(moduls)) !== null) {
  if (m.index === regexModul3.lastIndex) {
    regexModul3.lastIndex++;
  }
  m.forEach((match, groupIndex) => {
    arrayModuls.push(match);
  });
}
```

- Ara parsejem cada mòdul per extreure totes les propietats que formaran part de l’objecte mòdul.
- Si ho aconsegueixo creo un objecte mòdul nou, i omplo amb les dades obtingudes fins el moment. Hi ha una variable control per veure si el nombre de mòduls possibles, és el mateix que que els s’han pogut parsejar. Només en el cas que coincideixin afegirem el mòdul a la BBDD

```
if (modul.startsWith("Mòdul prof")) {

  numModulsPossibles ++;

  const regexUFNumDurada =
    /Mòdul professional\s*(\d+)\:(.+)Durada\:\s?(\d+).+/gm;
  let dadesUF = regexUFNumDurada.exec(modul);
  if (!dadesUF) {
    ok = false;
  } else {
    const num = Number(dadesUF[1]);
    const nom = dadesUF[2];
    const durada = Number(dadesUF[3]);

    //  if (num==1) {
    //    console.log(modul);
    //  }

    const regexUFHoresLliureDisp = /Hores de lliure disposició:\s?(\d+)/gm;
    dadesUF = regexUFHoresLliureDisp.exec(modul);
    let horesLliureDisposicio = 0;
    if (dadesUF) {
      horesLliureDisposicio = Number(dadesUF[1]);
    }

    const regexUFECTS = /Equivalència en crèdits ECTS\:\s?(\d+)/gm;
    dadesUF = regexUFECTS.exec(modul);
    let equivalenciaECTS = 0;
    if (dadesUF) {
      equivalenciaECTS = Number(dadesUF[1]);
    }
    //console.log(num, nom, durada, horesLliureDisposicio, equivalenciaECTS),
    if (ok) {
      var modulBD = new Modul({
```

- Ara toca parsejar les UFs del mòdul, procedim de manera anàloga creant un separador entre UFs.
- I generem un array d'UFs. Per cada UF obtenim les dades generals, i finalment el Resultats d'aprenentatge, generant un array de resultats d'aprenentatge, cadascun amb el seu array de Criteris d'Avaluació i l'array de Continguts i cadascun amb el seu array de Subcontinguts.

Si veiem el pdf, per entendre les dades que se n'extreuen, primer les dades generals del mòdul:

Mòdul professional 1: Comunicació i Atenció al Client

Durada: 132 hores
Hores de lliure disposició: no se n'assignen
Equivalència en crèdits ECTS: 12

i després les dades de cada UF

UF 1: processos de comunicació oral a l'empresa

Durada: 55 hores

Resultats d'aprenentatge i criteris d'avaluació

1. Caracteritza les tècniques de comunicació institucional i promocional, distingint entre internes i externes.

Criteris d'avaluació

1.1 Identifica els tipus d'institucions i organitzacions empresarials descrivint-ne les característiques funcionals i organitzatives.

1.2 Relaciona les funcions tipus de l'organització: direcció, planificació, organització, execució i control.

1.3 Identifica l'estructura organitzativa per donar una assistència o la prestació d'un servei de qualitat.

1.4 Relaciona els diferents estils de comandament d'una organització amb el clima laboral que generen.

1.5 Defineix els canals formals de comunicació en l'organització a partir de l'organigrama.

1.6 Diferencia els processos de comunicació interns formals i informals i els relaciona amb la contribució a l'eficàcia i la cohesió de l'organització.

1.7 Reconeix la influència de la comunicació informal i les cadenes de rumors en les organitzacions i la seva repercussió en les actuacions del servei d'informació que es dona.

1.8 Relaciona el procés de demanda d'informació d'acord amb el tipus de client, intern i extern, que pot intervenir-hi.

1.9 Valora la importància de la comunicació externa en la transmissió de la imatge corporativa de l'organització en les comunicacions formals.

1.10 Identifica els aspectes més significatius que transmeten la imatge corporativa en les comunicacions institucionals i promocionals de l'organització.

… i pel que fa als continguts :

Continguts

1. Tècniques de comunicació institucional i promocional:

1.1 Les organitzacions empresarials. Característiques funcionals i organitzatives en funció de la dimensió i la forma jurídica.

1.2 Les funcions en l'organització: direcció, planificació, organització i control. Els departaments.

1.3 Tipologia de les organitzacions. Organigrames.

1.4 Direcció en l'empresa.

1.5 Processos i sistemes d'informació en les organitzacions.

1.6 Tractament de la informació. Fluxos interdepartamentals.

1.7 Elements i barreres de la comunicació.

1.8 Comunicació i informació i comportament.

1.9 Les relacions humanes i laborals en l'empresa.

1.10 La comunicació interna en l'empresa: comunicació formal i informal. Contribució a l'eficàcia i la motivació.

1.11 La comunicació externa en l'empresa. Contribució a la imatge corporativa.

1.12 Qualitat del servei i atenció de demandes. Normes de qualitat aplicables. Indicadors de qualitat. Anàlisi de les no conformitats.

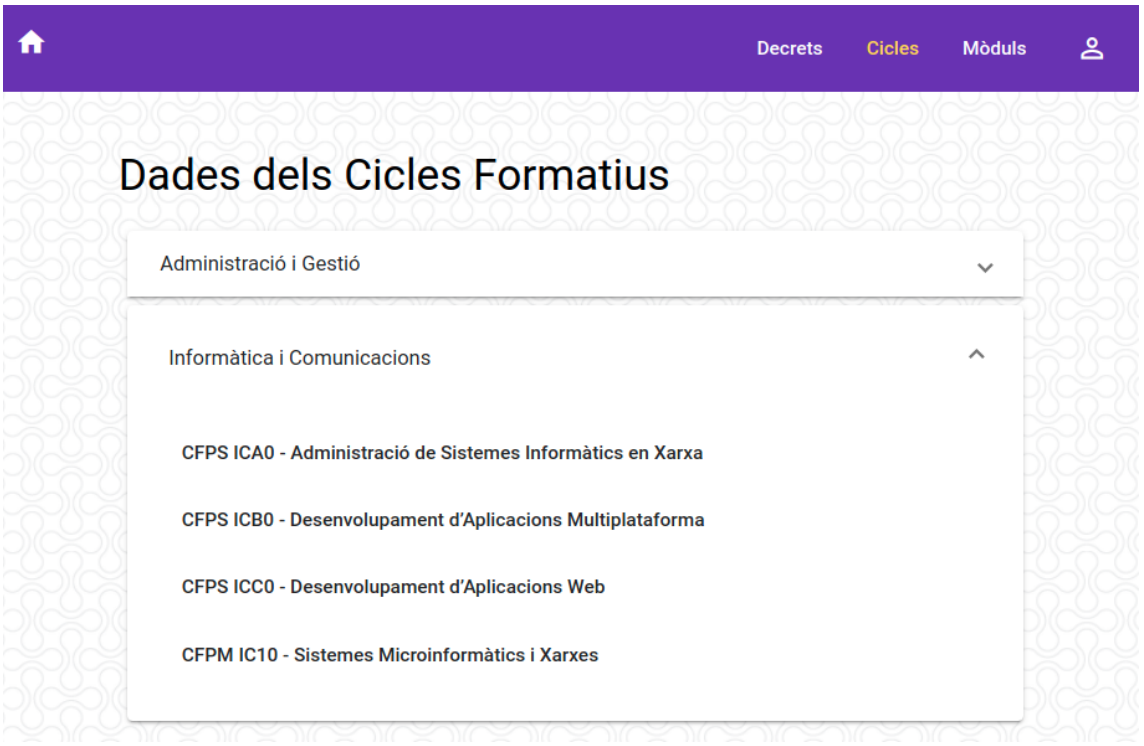
Les expressions regulars que permeten obtenir la informació són:

```
/Mòdul professional\s*(\d+)\:(.+)Durada\:\s?(\d+).+/gm

/Hores de lliure disposició:\s?(\d+)/gm;

/Equivalència en crèdits ECTS\:\s?(\d+)/gm;

/hores\s*(UF *1:[\s\S]+)/gm;

/(UF[\s\S]+?)#####/

/UF\s*(\d+):\s*(.+[\s\S]+?)Durada:\s*(\d+)/gm

/(\d\.[\s\S]+?)Continguts/gm

`(${dadesRAs[1]}\\.\\d+)\\s?(.+)`

/(\d)\.\s+(.+)/gm

(${dadesContinguts[1]}\\.\\d+)\\s?(.+)
```

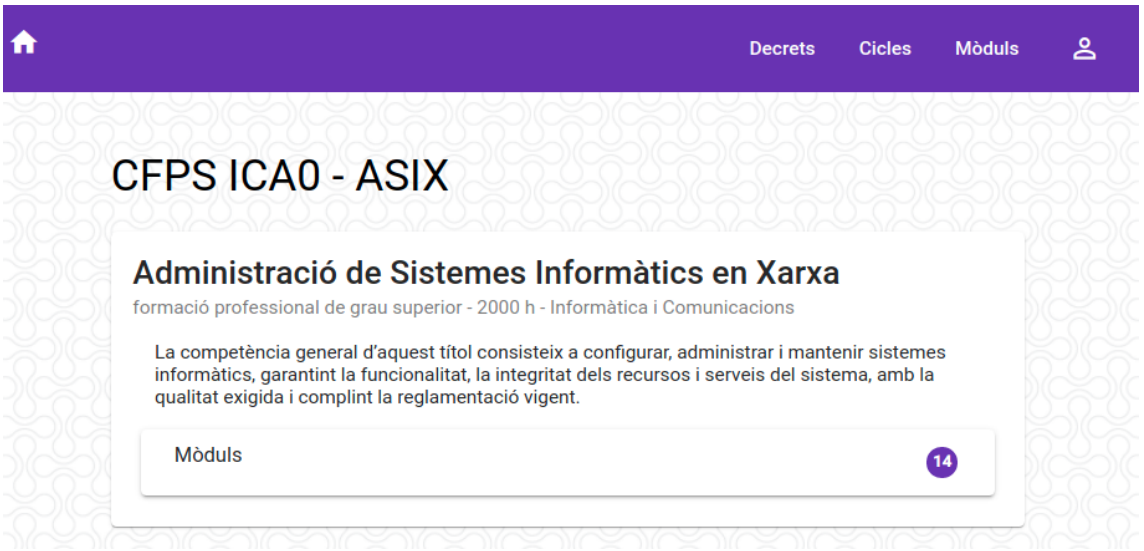
Si aconsegueixo parsejar tots els mòduls possibles, és que he tingut èxit, per tant genero un document Mòdul per cada mòdul possible.

```
if (numModulsPossibles == numModulsParsejats && arrayModulsBD.length>0) {
  arrayModulsBD.forEach ( modul => {
    modul
      .save()
      .then((result) => {
        console.log("Mòdul desat ", result._id);
        idModulsAfegits.push(result._id);
      })
      .catch((err) => {
        console.log(err);
        return err;
      });
  })
}
```

Per accedir a la vista d'aquestes dades al frontend cal estar autentificat com a professor o administrador i queda de la següent manera:



Apareixen 2 opcions més al menú,cicles i mòduls, i els cicles estan agrupats per famílies, si cliquem en un dels cicles (sempre que hi ha consultes a la BBDD s'exeuta un spinner, o similar per mostrar l'estat d'espera):



si despleguem, apareix la llista de tots els mòduls, si cliquem sobre un mòdul:

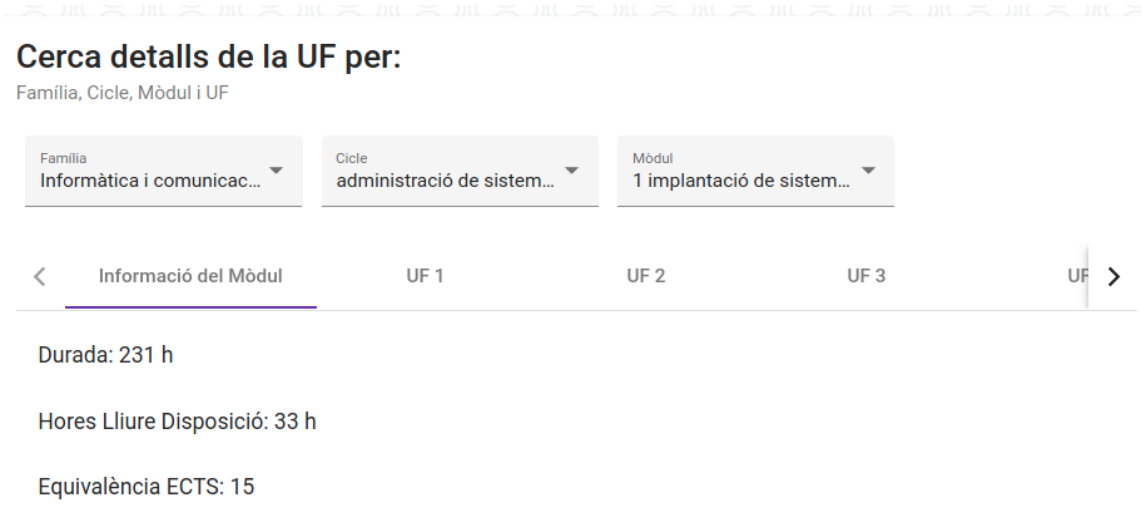
Ens apareix la informació general del mòdul, pestanyes per accedir a la informació de cada UF i un botó saber-ne més per anar al detall, si anem al detall

Tenim la mateixa informació que abans però amb 2 botons, resultats d'aprenentatge i continguts, si cliquem sobre algun d'aquests botons:

Ens apareixen els resultats d'aprenentatge d'aquella UF i si cliquem sobre un RA concret, n'apareixen els criteris d'avaluació relacionats

De manera anàloga amb els continguts.

A part tenim una darrera opció del menú que ens permet fer una cerca d'un mòdul concret, a partir d'un formulari, amb opcions select que es generen dinàmicament, a mesura que anem avançant en la tria.



i es pot accedir al mateix nivell de detall d'informació que hem vist anteriorment.

6. Conclusions

Què he après?

La realització d'aquest projecte ha comportat l'aprenentatge de les tecnologies necessàries per crear un aplicatiu web basat en MEAN stack. Donat que la programació web no forma part del currículum ha estat una bona oportunitat per aprendre i aprofundir en aquestes tecnologies.

A més s'ha intentat desenvolupar un aplicatiu que ara per ara no existeix. Facilitant la consulta de dades concretes del currículum d'un cicle formatiu, que són molt tedioses de consultar directament sobre el document.

Assoliment dels objectius plantejats inicialment:

No s'han assolit tots els objectius plantejats a l'inici. La idea inicial era poder crear programacions didàctiques a partir d'aquesta base de dades, però una vegada analitzat l'abast de l'aplicatiu, donaria per fer un altre TFG.

Tampoc s'ha pujat a producció, en aquest sentit és molt dificultós pujar un aplicatiu a producció sense recursos. Estaria bé que la UOC oferís un compte AWS amb un crèdit limitat, o similar, per facilitar-ho.

Seguiment de la planificació i metodologia al llarg del projecte:

En línies generals s'ha seguit la planificació inicial, tot i que ha calgut introduir canvis. La part d'aprenentatge de les tecnologies ha estat més costosa en hores de l'esperat, i ha fet que el desenvolupament hagi començat una setmana més tard de la planificació inicial. Tot i que he recuperat el temps, ja que ja tenia un coneixement de les tecnologies.

El desenvolupament del backend i frontend s'havia previst independentment al principi, i finalment s'ha fet en paral·lel.

L'ús de bootstrap ha estat substituït per Angular Material, així he tingut l'oportunitat d'explorar aquesta biblioteca de components d'Angular.

Línies de futur

Aquest projecte té una bona base per acabar desenvolupant una aplicació potent de generació de programacions didàctiques. A curt termini caldria trobar un entorn on poder posar a producció l'aplicatiu, aquestes serien les opcions per fer-ho:

- Mantenir MongoAtlas com a base de dades però agafar una versió de pagament, hi ha opcions des de 57€ al més
- Desenvolupar el backend i el frontEnd a una solució al núvol com AWS, o bé heroku.

Ara per ara ha quedat en una eina de consulta, però podria ser una eina de generació de programacions, la idea seria integrar-la amb la API de google document per poder fer una plantilla de programació on el docent crearia la programació en un entorn d'apliactiu web, i es generaria com a resultat un Google document amb el format esperat.

També caldria que l'usuari administrador pogués gestionar els usuaris, i una gestió amb més detall de les bases de dades generades, per tal de corregir alguna errada provinent de la generació automàtica.

7. Glossari

- MEAN MongoDB, Express, Angular, Node.js
- HTTP Hyper Text Transfer Protocol
- REST Representational State Transfer
- JSON JavaScript Object Notation
- BSON - Binary JSON
- API Application Interface
- SPA Single Page Application
- MVC Model – View – Controller
- I/O Input / Output
- CRUD Create, Read, Update, Delete
- CMS Content Management System
- SPA Single Page Application
- AJAX Asynchronous JavaScript And XML
- BBDD Base de Dades
- URL Uniform Resource Locator

8. Bibliografia, Webgrafia

1. SPA. https://en.wikipedia.org/wiki/Single-page_application
2. Express. https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction
3. Curs Udemy, javascript, angular i nodejs.

   https://www.udemy.com/course/master-en-javascript-aprender-js-jquery-angular-nodejs-y-mas/learn/lecture/10236306?start=0#overview

4. Curs Udemy, frameworks basats en javascript,

   https://www.udemy.com/course/master-en-frameworks-javascript-aprende-angular-react-vue-js/learn/lecture/15949402?start=120#overview

5. Curs Udemy MongoDB.

   https://www.udemy.com/course/mongodb-the-complete-developers-guide/learn/practice/1062344?start=start-page#overview

6. Curs Udemy Angular and NodeJs.

   https://www.udemy.com/course/angular-2-and-nodejs-the-practical-guide/learn/lecture/13914132?start=0#overview

7. Curs Udemy Regexp:

   https://www.udemy.com/course/regex-academy-an-introduction-to-text-parsing-sorcery/learn/lecture/8040106?start=75#overview

8. CRUD:

   https://es.wikipedia.org/wiki/CRUD

9. Stack Overflow

   https://stackoverflow.com/

9. Annexos

Codi en ZIP (accedir amb compte de la UOC)

https://drive.google.com/file/d/1SpST66KPkKnS9IMqcepXgS41labRC1k8/view?usp=sharing

Vídeo Demostratiu del funcionament

https://drive.google.com/file/d/1GLZwqh_lOuiVkujSqgU2tQeh2kDdDj9E/view?usp=sharing

Presentació

https://docs.google.com/presentation/d/17C18w9eD-IlcEYL8-EG2nIzoavWQfjAfd7RlvIk3J7g/edit?usp=sharing