Desenvolupament d'una *Progressive Web App* (PWA) de gestió d'una AMPA

Memòria de Projecte Final de Màster

Màster Universitari en Desenvolupament d'Aplicacions i Llocs Web.

Autor: *Josep V. Monjo Agut*

Professors: *César Pablo Córcoles Briongos* i *Carles Arnal Castello*

07/06/2021

Resum

La present **Memòria de Projecte Final de Màster** planteja el desenvolupament d'una PWA que facilitarà la gestió de les Associacions de Mares i Pares d'Alumnes (AMPA) a més a més de fomentar la participació dels propis pares, propiciar la sol · licitud de serveis de manera online, així com facilitar la comunicació al sí de l'associació. Per a això usarem *Quasar framework* (VueJS) per al *frontend* i *Hasura* (GraphQL) per al backend.

Abstract

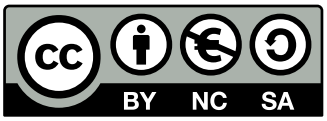
*This **Final Master's Thesis** proposes the development of a PWA that will facilitate the management of Parents Associations in addition to encouraging the participation of parents themselves, promote the request for services online, as well as facilitate communication within the association. To accomplish that we are going to use Quasar framework (VueJS) for the frontend and Hasura (GraphQL) for the backend.*

keywords: PWA, AMPA, GraphQL, VueJS, NodeJS, Hasura, Quasar

Índex

Índex de taules

Índex de figures



Codi font del document: https://github.com/fampa/tfm/

1 Introducció

1.1 Context

No cal recordar que el context actual de pandèmia mundial propícia que cada vegada més les persones romanguen a sa casa i realitzen totes les seues gestions de manera telemàtica, posant de relleu la importància de les eines digitals i de telegestió com ara videoconferències, aprenentatge online, administració electrònica, teletreball, etc...

És en aquest context que sorgeix la necessitat de modernitzar la gestió de l'Associació de Mares i Pares d'Alumnes (AMPA).

1.1.1 Estudi de mercat

Fent un estudi de mercat, en un primer moment es va plantejar la possibilitat d'elaborar un model *SaaS*[1]. En aquest cas l'aplicació podria contenir la gestió de diferents AMPA i cadascuna d'elles podria tindre el seu propi subdomini. Aquest és el model més freqüent al mercat per a quest tipus d'aplicacions web.

No obstant això, aquesta possibilitat es va descartar perquè la finalitat inicial no era muntar un negoci *SaaS* si no desenvolupar internament una eina per a l'autogestió de la

[1] De les segles en anglès *Software as a Service*.

AMPA de la que forme part com a membre de la Junta, fent-la el més oberta i transparent possible per a que puga ser implementada a altres AMPA o associacions que no vulguen un model de subscripció si no un d'autogestionat.

No obstant això, aquesta aplicació és relativament fàcil de reconvertir a un model de SaaS si en el futur es volgués usar per aquesta finalitat, ja que no totes les AMPA disposen de personal qualificat per posar en marxa una aplicació web a partir del seu repositori públic.

A continuació detallaré un **estudi de la competència** en aquest sector.

1.1.1.1 Mi ampa

- Url: https://miampa.com/
- Model de negoci: SaaS
- Descripció: ofereix inscripció online de socis, comptabilitat, multiidioma, gestió d'esdeveniments, extraescolars, tenda, galeria d'imatges, menus, carnet virtual, enquestes, web, personalització.
- Preu: s/d

1.1.1.2 playoff gestión de asociaciones

- Url: https://playoffinformatica.com/gestion-de-asociaciones
- Model de negoci: SaaS
- Descripció: Versàtil per a qualsevol tipus d'associació. Gestió de pagaments online i rebuts SEPA, gestió d'activitats, comunicacions per correu i mòbil, carnet virtual.
- Preu: entre 25€ i 65€ mensuals

1.1.1.3 AmpaSoft

- Url: https://ampasoft.es/
- Model de negoci: SaaS
- Descripció: gestió d'alumnes/pares, gestió econòmica (generació de remeses bancàries), missatgeria.
- Preu: s/d

1.1.1.4 AmpaNet

- Url: https://www.ampanet.es/
- Model de negoci: SaaS
- Descripció: gestió d'alumnes/pares, extraescolars, menjador, acampades, activitats, rebuts SEPA, web, tenda.
- Preu: s/d

1.1.1.5 EduTeca

- Url: https://edutecaservicios.es/software-especializado/
- Model de negoci: SaaS
- Descripció: Gestió altes/baixes, tallers/activitats, comunicacions, actes de junta i assemblea, web, memòria d'activitat i control de pressupost, matinera, menjador, comptabilitat, remeses bancàries.

1.1.1.6 GesAmpa

- Url: https://gesampa.com/joomla/

- Model de negoci: Llicència d'ús de Software + suport.
- Descripció: Gestió de membres i domiciliacions bancàries.
- Preu: des de 15€/mes o 100€/any

1.2 Justificació del Treball

La gestió de les Associacions de Mares i Pares d'Alumnes inclou bases de dades amb dades de pares, d'alumnes, de pagament de quotes, de serveis, etc... Aquesta tasca s'ha realitzat tradicionalment de manera offline (MS Excel, MS Access, formularis en paper,...) i amb poca o cap automatització. El canvi a una eina online facilitarà no només aquesta gestió si no que propiciarà la participació dels propis pares i mares en la gestió i actualització de les seues dades, així com en la sol · licitud de serveis que ofereix la seua AMPA de manera online, evitant l'ús de formularis en paper i minimitzant el contacte físic, aspecte aquest molt important en aquests temps de pandèmia mundial.

A més a més facilitarà la comunicació dels membres de la Junta de govern del AMPA amb els seus afiliats, ajudant a gestionar els continguts coma ara notícies, activitats, juntes, etc... així com l'enviament de notificacions push.

2 Descripció

S'ha decidit finalment aplicar un enfocament basat en el model d'aplicació web progressiva (PWA[1]) en el *frontend* en combinació amb una API GraphQL per al *backend*.

El model d'aplicació web progressiva està basat en estàndards web i no en cap *framework* o empresa. Per a que una aplicació web siga considerada **PWA**, ha de complir una sèrie de requisits que podríem dividir en tres grans àrees (Richard i LePage 2020):

- Capaç: Gràcies a les cada vegada més extenses API web, hi ha poques coses que una web app no siga capaç de realitzar avui en dia (xat en temps real, notificacions push, geolocalització, etc...).
- Confiable: una PWA s'ha de sentir ràpida, tant en la càrrega com en la interacció amb l'usuari. A més a més ha de permetre certa usabilitat amb una connexió a la xarxa dèbil o inexistent. Açò és possible gràcies als *service workers*.
- Instal · lable: una PWA s'executa en la seua pròpia finestra i ha de poder ser llançada des de l'escriptori de l'usuari.

Pel que fa a **GraphQL** és un llenguatge query per a APIs que proporciona de manera automàtica una documentació de les dades que poden ser extretes de la API, autocompletat de les *queries* i a més a més permet obtenir només aquells camps que ens interessen des d'un únic *endpoint*.

[1] De les segles en anglès *Progressive Web App*.

El projecte constarà de les fases que detallarem a continuació.

2.1 Investigació/Requeriments

Investigació de les característiques que haurà de tenir l'aplicació en base a les necessitats d'una AMPA, seguit de l'elaboració d'un document de requeriments que haurà de ser aprovat per part del client. Una vegada tancat aquest document iniciarem la següent fase.

2.2 Disseny

Elaboració del disseny de l'aplicació. S'elaborarà un *wireframe* de baixa fidelitat i/o un *mockup* d'alta fidelitat del disseny final.

Una vegada aprovat el disseny final passarem a la fase de desenvolupament.

2.3 Desenvolupament

2.3.1 Backend

Instal · larem l'eina Hasura (a un servidor VPS contractat pel client, o a un servei al núvol). Això ens facilitarà tindre en funcionament una base de dades postgreSQL i un servidor graphQL de manera ràpida.

Dissenyarem les taules i les relacions necessàries, segons el full de requeriments.

Configurarem el sistema d'autenticació amb Firebase i el d'autorització amb *Hasura*.

Addicionalment posarem en marxa un petit servidor *NodeJS* per gestionar transaccions amb dades sensibles com per exemple certes claus API que necessiten residir al costat del servidor, i els mecanismes per autoritzar a un administrador, així com d'altres com ara l'enviament de missatges per correu electrònic o de notificacions push.

2.3.2 Frontend

Instal · larem localment el framework Quasar, basat en VueJS.

Elaborarem les rutes necessàries per a la nostra app així com el sistema d'autenticació i autorització.

Adaptarem la nostra app per que complisca els requisits d'una PWA i ens assegurarem que hem acomplit tots els requeriments.

2.4 Proves i millores de rendiment

Farem proves de les diferents funcions i aplicarem les millores suggerides per l'eina de rendiment Lighthouse.

3 Objectius

3.1 Objectiu principal

- La modernització en la gestió de les AMPA, integrant el catàleg d'eines i aplicacions de gestió obsoletes en una única aplicació centralitzada i allotjada a un servidor web.

3.2 Objectius secundaris

- El plantejament de les necessitats de les AMPA que puguen ser assolides mitjançant una aplicació web i que plasmarem en forma de requeriments.
- Substituir les eines privatives que poguéssin estar usant en l'actualitat per una aplicació web de codi obert que puga ser usada i adaptada per totes les AMPA de manera lliure.
- Evitar l'ús de formularis en paper i minimitzar el contacte personal.
- Afavorir la immediatesa en les gestions i comunicacions amb l'AMPA.

4 Continguts

L'aplicació consisteix en les següents pàgines:

- Login: pàgina que permetrà fer login amb correu electrònic o compte de Google. Si l'usuari no existeix el crearà automàticament amb *Firebase*.
- Inici: apareixerà un feed amb les darreres notícies.
- Notícia (detall): detall de la notícia.
- Dades personals: inclou dades de l'usuari, de la família i del fill/s o filla/es, així com les dades de domiciliació.
- Extraescolars: pàgina on es llistaran les diferents activitats extraescolars ofertades
- Extraescolar (detall): pàgina de detall de l'activitat extraescolar o podem subscriure els nostres fills.
- Matinera: llistat amb l'oferta d'escola matinera.
- Matinera (detall): detall on contractar la matinera.
- Diverses pàgines dinàmiques: com ara Qui som, Que fem, ...
- Pàgina de contacte
- Pàgina de política de privacitat i pàgina de condicions del servei.

Els usuaris amb permisos d'administrador també podran veure els següents continguts:

- Blog: llistat d'articles del blog.

- Blog (detall): Pàgina on editar o afegir un article del blog.
- Membres: Llistat de membres del AMPA.
- Membre (detall): la mateixa pàgina de Dades personals però amb els continguts de l'usuari seleccionat.
- Pàgines: Llistat on editar i afegir pàgines.
- Pàgina (detall): edició del contingut de la pàgina.
- Etiquetes: llistat d'etiquetes on afegir i editar
- Etiqueta (detall): formulari d'edició d'etiqueta
- Serveis: llistat de serveis on afegir i editar
- Servei (detall): formulari d'edició del tipus de servei
- Famílies: llistat de famílies i IBAN

5 Metodologia

S'emprarà una metodologia de tipus *waterfall* amb retroalimentació. Per tant es presentarà el document de requeriments al client per a que el valide, i una vegada fet procedirem a executar cadascuna de les tasques necessàries de manera seqüencial, tornant enrere a alguna d'elles si fos necessari per reajustar algun requeriment.

6 Arquitectura de l'aplicació

L'aplicació segueix un model MVVM[1]. Es tracta d'un model de desenvolupament que separa el *business logic* de la capa de presentació, el que propícia la separació i la modularització del nostre codi, afavorint l'elaboració de tests unitaris i col · laborant a tindre un codi més net i més fàcil de mantenir (Britz 2017). A diferència del model MVC[2], el MVVM utilitza llenguatge de marcat per a la capa d'unió entre la capa lògica i la capa de presentació. Aquest és el cas del *framework VueJS* tal i com podem veure a la fig. 6.1. A més a més e el model MVC el controlador és el punt d'entrada de l'aplicació mentre que en el MVVM la vista és el punt d'entrada (Rungta s.d.).

[1] *Model View View-Model*
[2] *Model View Controller*

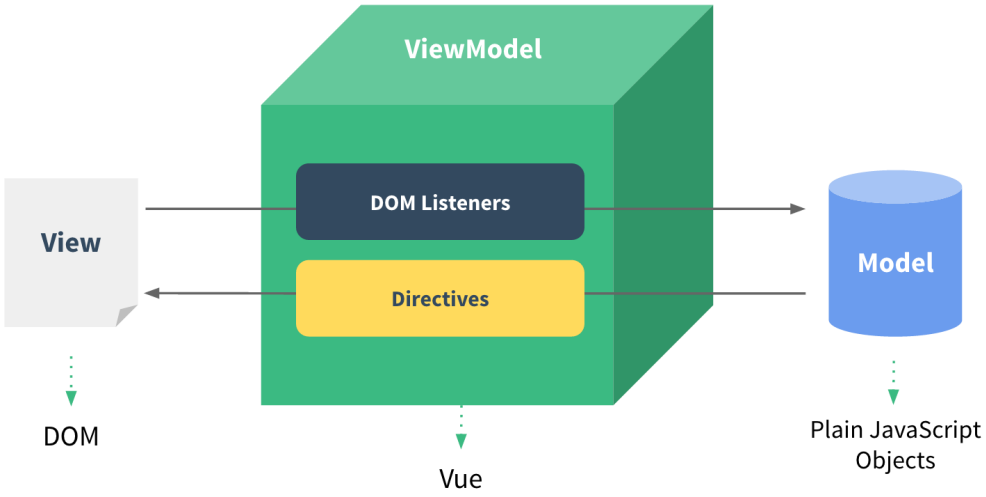


Figura 6.1: Model MVVM. Font: https://vuejs.org

6.1 Diagrames

A la fig. 6.2 se'ns mostra un esquema de l'arquitectura de la nostra app.

En resum la nostra app consta de 3 parts fonamentals:

- La base de dades, allotjada a un VPS amb sistema operatiu **Ubuntu**, amb **Docker** i exposada a l'exterior amb un proxy invers (**NGINX**), mitjançant el framework **Hasura**, la qual cosa ens proporciona una base de dades PostgreSql i una API graphql.
- El *frontend*, desenvolupat amb **Quasar Framework** (VueJS) i allotjat al CDN de Firebase, cosa que disminueix el temps de latència en la resposta.
- Un *backend* amb una arquitectura *serverless* de **Firebase Functions**, que és bàsicament un servidor amb NodeJs on poder programar el nostre codi del costat del servidor. En aquest cas aquesta part de la infraestructura la usem per a l'enviament de missatges i tasques automatitzades com ara la creació del perfil per

usuaris nous, l'atorgament de permisos d'usuari i administrador, etc... També fem ús de **Firebase Storage** per a l'allotjament d'imatges, i **Firebase Auth** per a la capa d'autenticació/autorització.

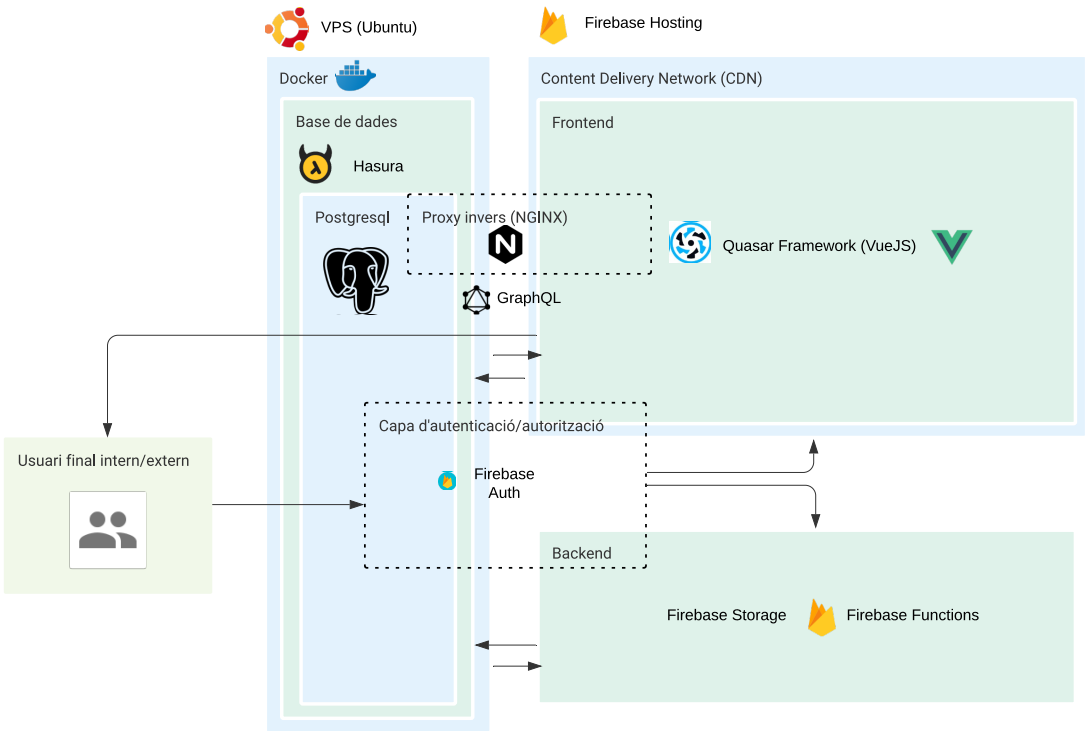


Figura 6.2: Diagrama Arquitectura app

A la fig. 6.3 figura el diagrama de casos d'ús de l'aplicació, posant èmfasi en els dos tipus d'actor, els pares/mares i els gestors de l'AMPA.

El diagrama de flux de les peticions de dades entre el client i la base de dades de la nostra aplicació el podem observar a la fig. 6.4.

A la fig. 6.5 podem observar el flux quan fem login a l'aplicació. Hi ha que tindre en compte que la llibreria Firebase Ui usa la mateixa porta d'entrada per fer login que per fer sign up.

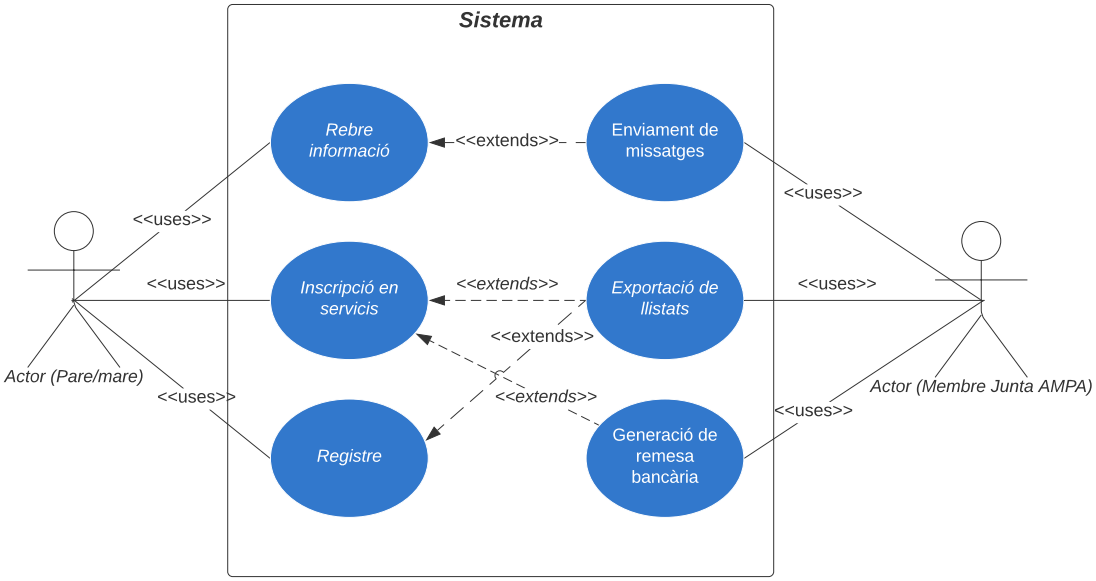


Figura 6.3: Diagrama de casos d'ús

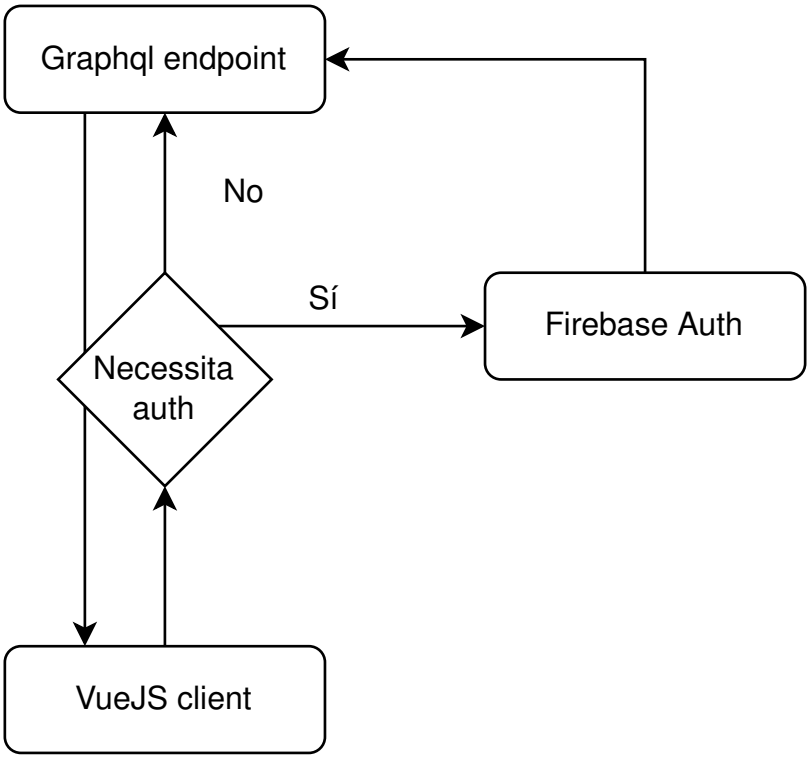


Figura 6.4: Diagrama de flux bàsic d'autorització

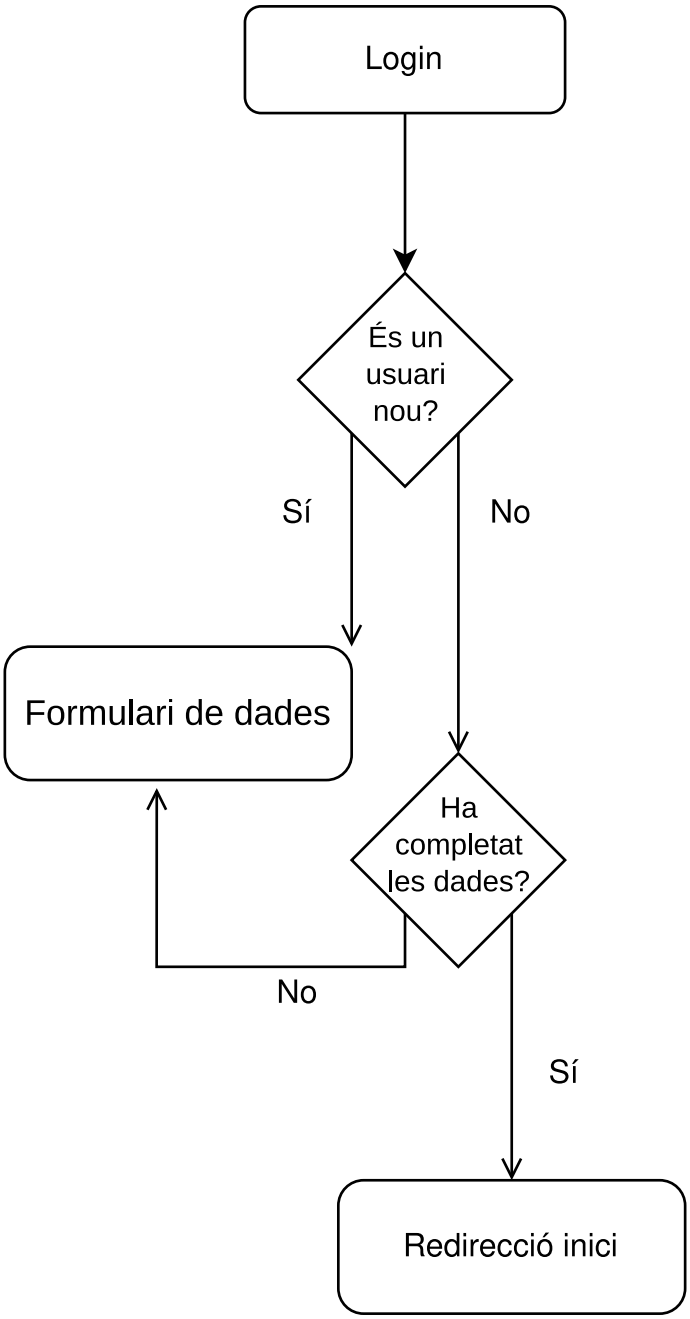


Figura 6.5: Diagrama de flux login

Un ús típic de l'aplicació serà la sol · licitud de serveis com ara les activitats extraescolars, tal i com es pot veure a la fig. 6.6.

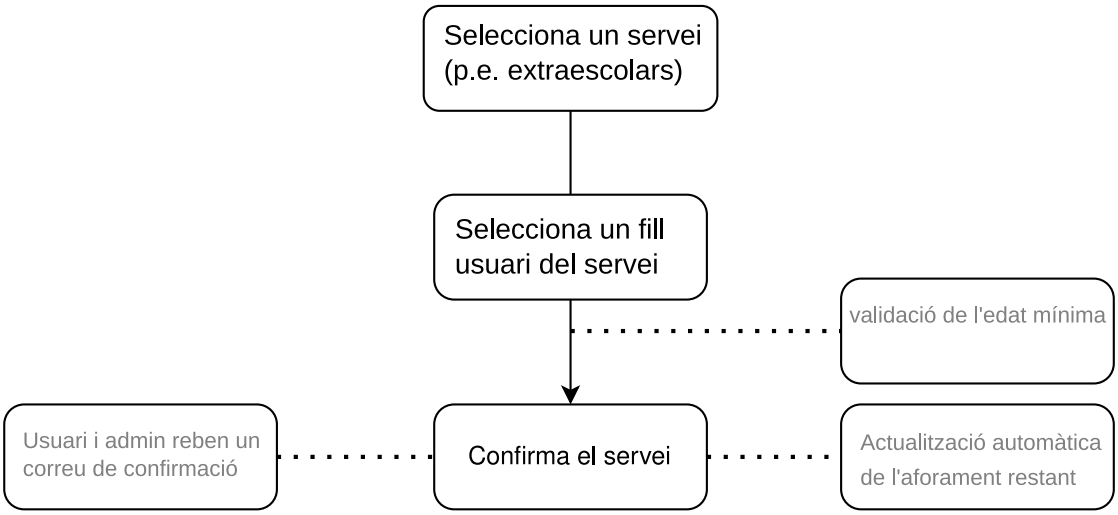


Figura 6.6: Diagrama de flux de sol · licitud de serveis

A la fig. 6.7 podeu observar el diagrama de classes en el que s'ha basat el disseny de la base de dades i dels models de TypeScript.

Aquest és el resultat d'una important refactorització. Podeu veure el diagrama anterior a la refactorització a l'annex 3.

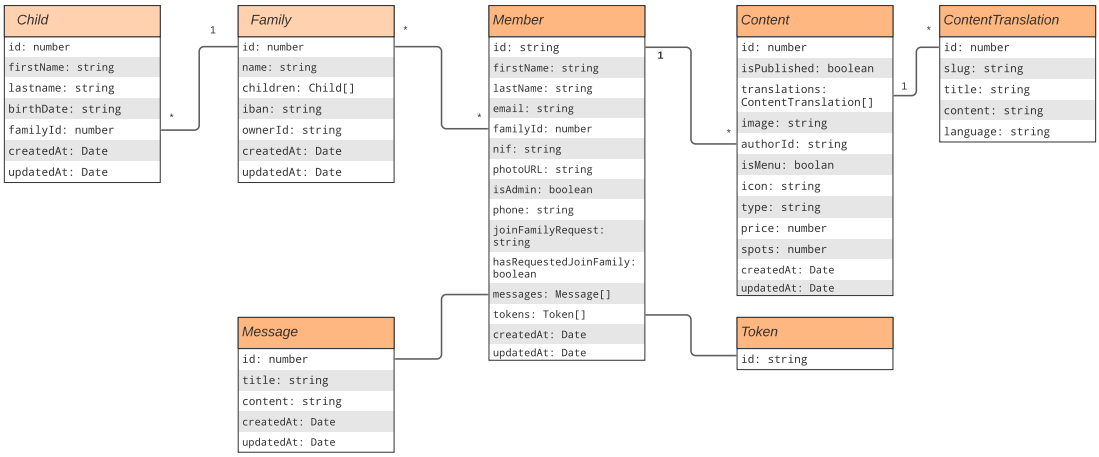


Figura 6.7: Diagrama de classes

6.2 Refactoring

Durant el desenvolupament de l'aplicació s'ha dut a terme una refactorització important tant de la base de dades com del *frontend*.

6.2.1 Base de dades

6.2.1.1 Problema a resoldre

Molts dels camps d'algunes taules de la base de dades es repetien, portant a duplicitats i complexitat innecessàries.

6.2.1.2 Solució

A la base de dades s'ha aplicat la refactorització *Extract interface*.[3] D'aquesta manera s'han pogut eliminar les taules Articles, Pages, Services, Tags i els corresponents Articles_translations, Pages_translations, Services_translations, Tags_translations, en favor de només dues taules: Contents i Contents_translations

6.2.2 Frontend

6.2.2.1 Problema a resoldre

Havíem detectat un *code smell*[4] que consistia a una repetició de codi en diferents components encarregats de renderitzar diferents tipus de contingut.

[3] https://refactoring.guru/extract-interface
[4] https://refactoring.guru/refactoring/smells

Gràcies a la refactorització de la base de dades hem pogut refactoritzar també el codi al *frontend*.

6.2.2.2 Solució

També hem aplicat el *Extract method* en aquest cas. Això ens ha permès eliminar els components específics que renderitzaven cadascún dels tipus de continguts per a passar a usar un únic component capaç de renderitzar tots ells.

El *commit* on hem dut a terme aquesta refactorització ha propiciat 2.263 eliminacions.

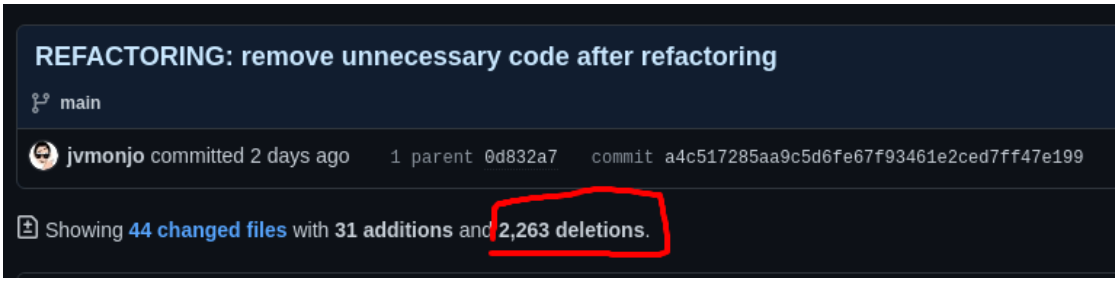


Figura 6.8: Resultat de la refactorització

6.3 Repositoris

- Frontend i Backend: https://github.com/fampa/ampa-pwa
- Base de dades i GraphQL API: https://github.com/fampa/ampa-graphql
- Landing page del projecte: https://github.com/fampa/ampa-landing
- Presentació: https://github.com/fampa/presentation
- Memòria del projecte: https://github.com/fampa/tfm

6.4 URLs

- APP: https://fampa-pwa.web.app/
- Landing page: https://ampa.netlify.app/
- Presentació: https://fampa.github.io/presentation/

7 Plataforma de desenvolupament

Necessitarem els següents recursos:

- Una base de dades postgreSQL gestionada per una API GraphQL, mitjançant el *framework Hasura*.
- Firebase functions, Firebase Hosting i Firebase Storage
- Un *frontend* VueJS mitjançant el framework *Quasar*.
- Un servei d'autenticació mitjançant *Firebase Auth*.

Per al desenvolupament s'han usat principalment dues màquines amb Sistema Operatiu Linux, i el IDE *Visual Studio Code*.

La present memòria està redactada en llenguatge *Markdown* i convertida a PDF amb Pandoc.

8 Planificació del treball

La memòria de Projecte Final de Màster consta d'una sèrie d'entregues parcials programades amb les dates que figuren a la taula 8.1.

Taula 8.1: Planificació del treball

PAC	Activitat	Inici	Fi
1	**Plantejament del projecte**	17/02/2021	02/03/2021
"	Document de requeriments	"	"
"	Context i Justificació	"	"
"	Descripció del projecte	"	"
"	Objectius	"	"
"	Metodologia	"	"
2	**Desenvolupament i documentació I**	03/03/2021	31/03/2021
"	Desenvolupament i docs Backend	"	"
"	Desenvolupament i docs Frontend	"	"
"	Continguts	"	"
"	Arquitectura de la app	"	"
3	**Desenvolupament i documentació II**	01/04/2021	09/05/2021
"	Desenvolupament i docs Backend	"	"
"	Desenvolupament i docs Frontend	"	"

PAC	Activitat	Inici	Fi
"	MVP i primeres proves (vídeo)	"	"
"	Procés de treball	"	"
"	CI/CD	"	"
Final	**Desenvolupament i documentació III**	10/05/2021	07/06/2021
"	Finalitzar Desenvolupament i docs	"	"
"	Optimitzacions (Lighthouse)	"	"
"	Finalitzar Memòria TFM	"	"
"	Elaborar presentació	"	"
"	Presentació en vídeo	"	"
"	Elaborar autoinforme d'avaluació	"	"

9 Procés de treball/desenvolupament

9.1 Investigació

Per avaluar les necessitats de l'AMPA, s'ha parlat amb membres de la junta directiva i conjuntament amb ells s'ha elaborat el document de requeriments que figura a l'annexe *Especificacions*.

També s'han estudiat els formularis en paper existents, així com la base de dades actual.

Per saber com enfocar el disseny hem tingut en compte les estadístiques d'ús de dispositius des dels quals la gent es connecta a internet de manera habitual. Segons el CIS -Centro de Investigaciones Sociológicas (2017), el 87% de la població es connecta a internet amb el mòbil mentre que només ho fa des d'un ordinador o portàtil entre el 45% i el 54%.

9.2 Disseny dels mockup

Hem dissenyat l'aplicació amb l'eina online Figma usant un acostament *mobile first*, ja que s'espera que la majoria de trànsit vinga des de dispositius mòbils.

S'ha optat per un disseny basat en el patró Material Design[1]. Un patró de disseny proposat per Google i que va tenir un grau auge gràcies a l'explosió d'Android com a sistema operatiu més usat en plataformes mòbils.

Per a la pantalla de login, tal i com es pot observar a la fig. 9.1, he optat per l'estil que proporciona per defecte la llibreria *Firebase UI*, personalitzant el text dels botons per a que concorde amb la localització de la app.

El menú (fig. 9.1) tindrà dues parts diferenciades, una visible per a tothom i l'altra de gestió només visible per a administradors.

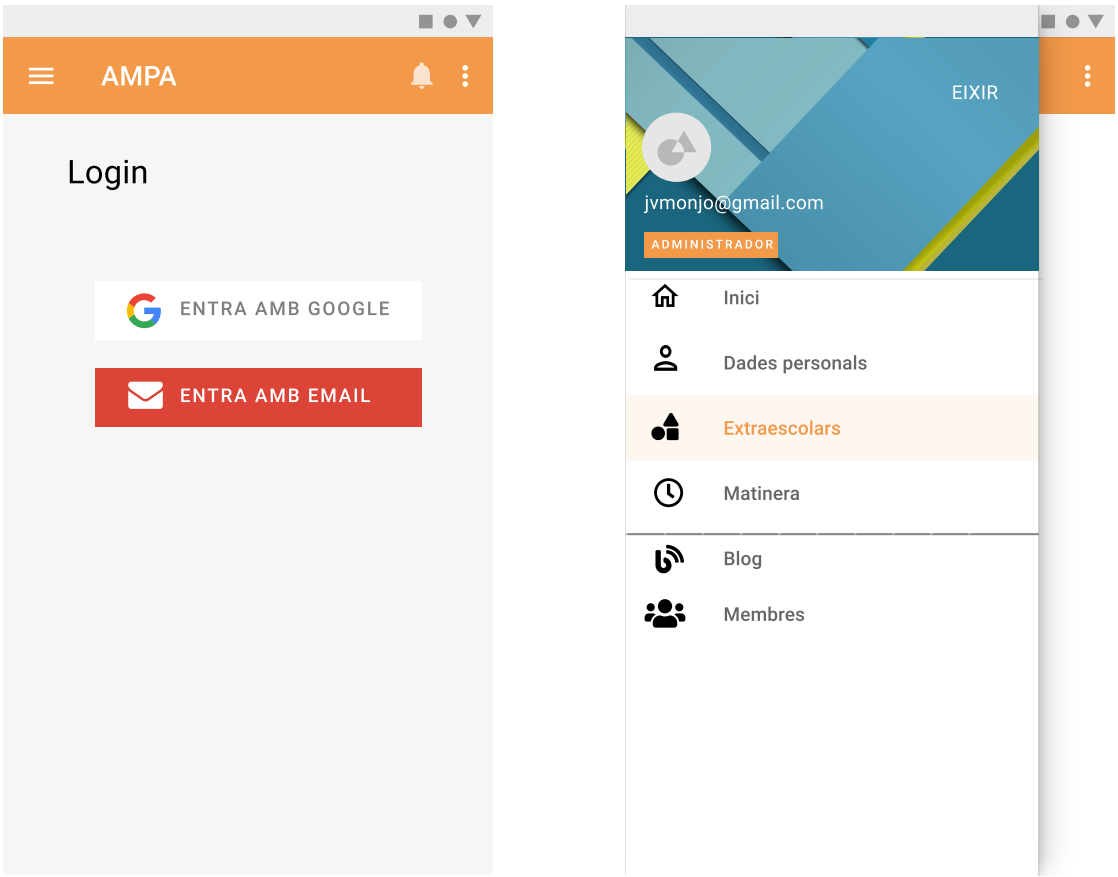


Figura 9.1: Mockup: Login i menu

[1] https://material.io/design

A la fig. 9.2 podem veure el formulari de dades personals i familiars. També observem el llistat que gestionaran només els administradors

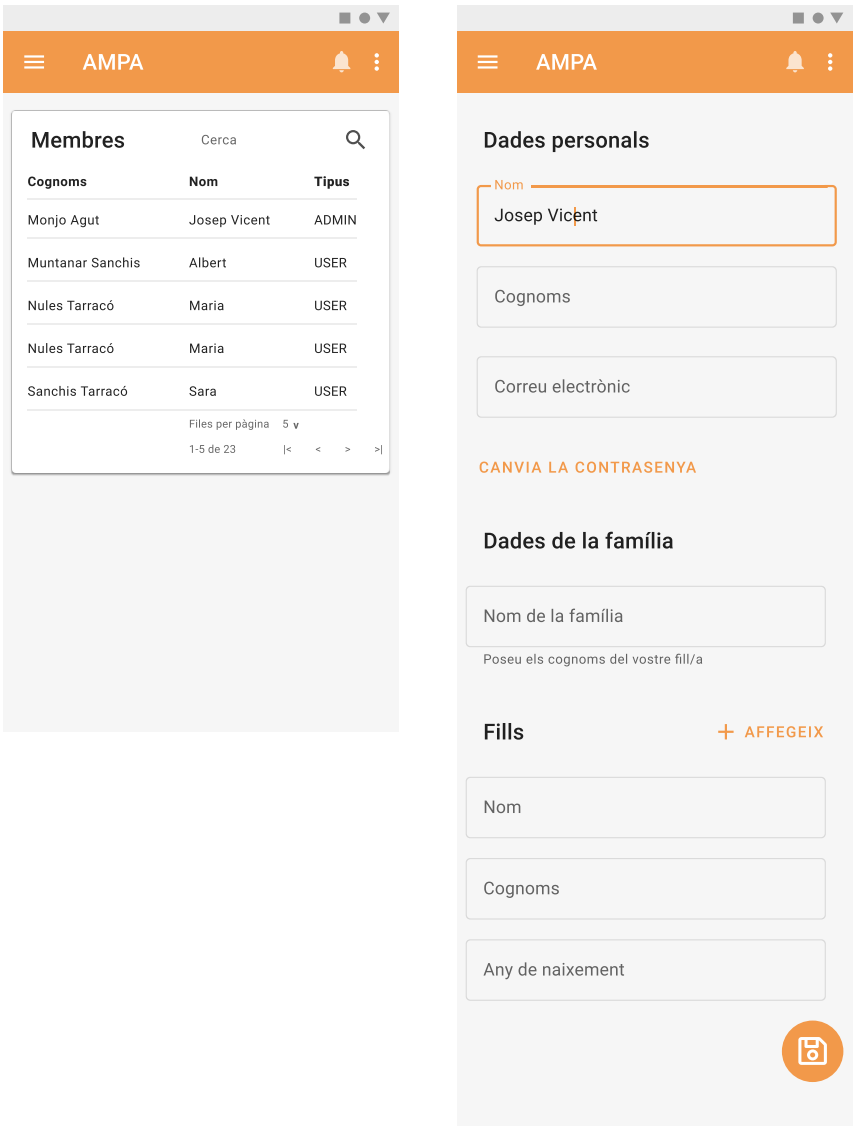


Figura 9.2: Mockup: Gestió d'usuaris

Per a la gestió de serveis com ara la matinera o les extraescolars (fig. 9.3) s'ha optat per presentar-ho en forma de taula i una vegada fem click que ens porte als detalls.

En la portada tindrem el blog on comptarem a una vista de targetes tal i com podem

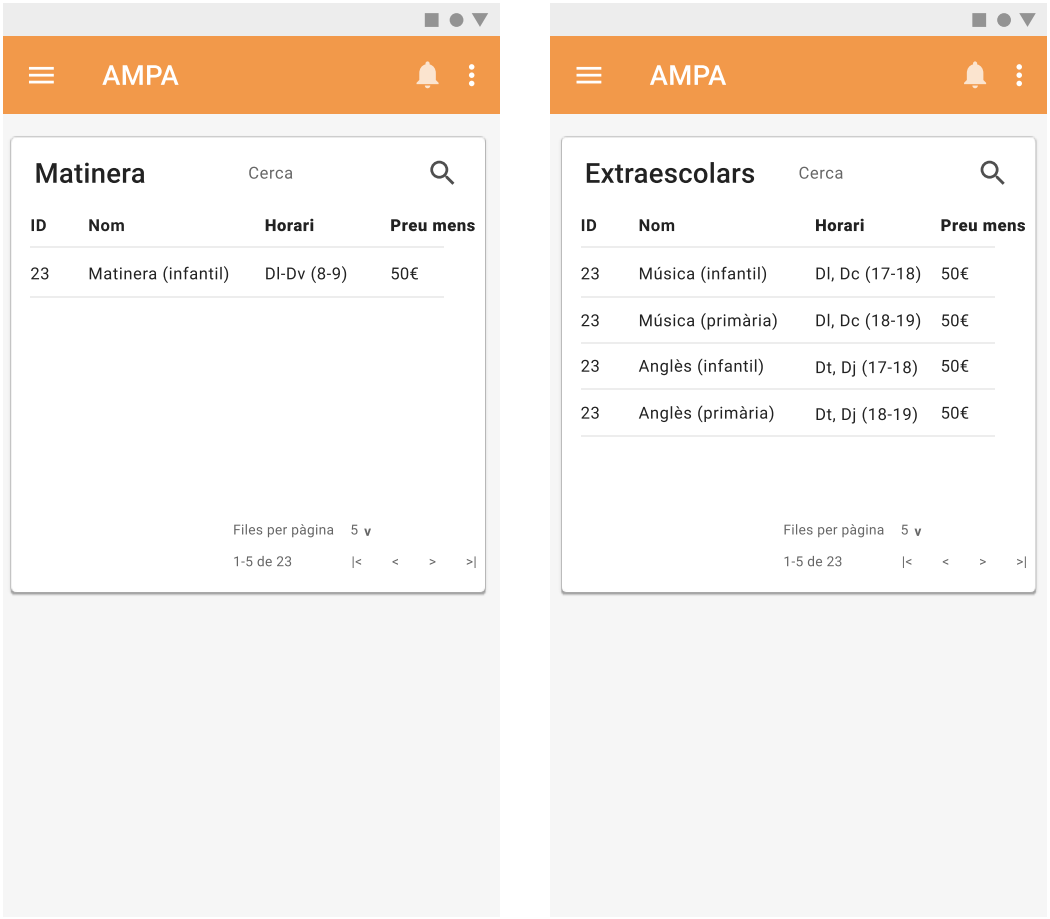


Figura 9.3: Mockup: Gestió de serveis

veure a la fig. 9.4 i al fer click a la targeta anirem als detalls de l'article. També podem veure el llistat d'articles tal i com ho veurà l'administrador.

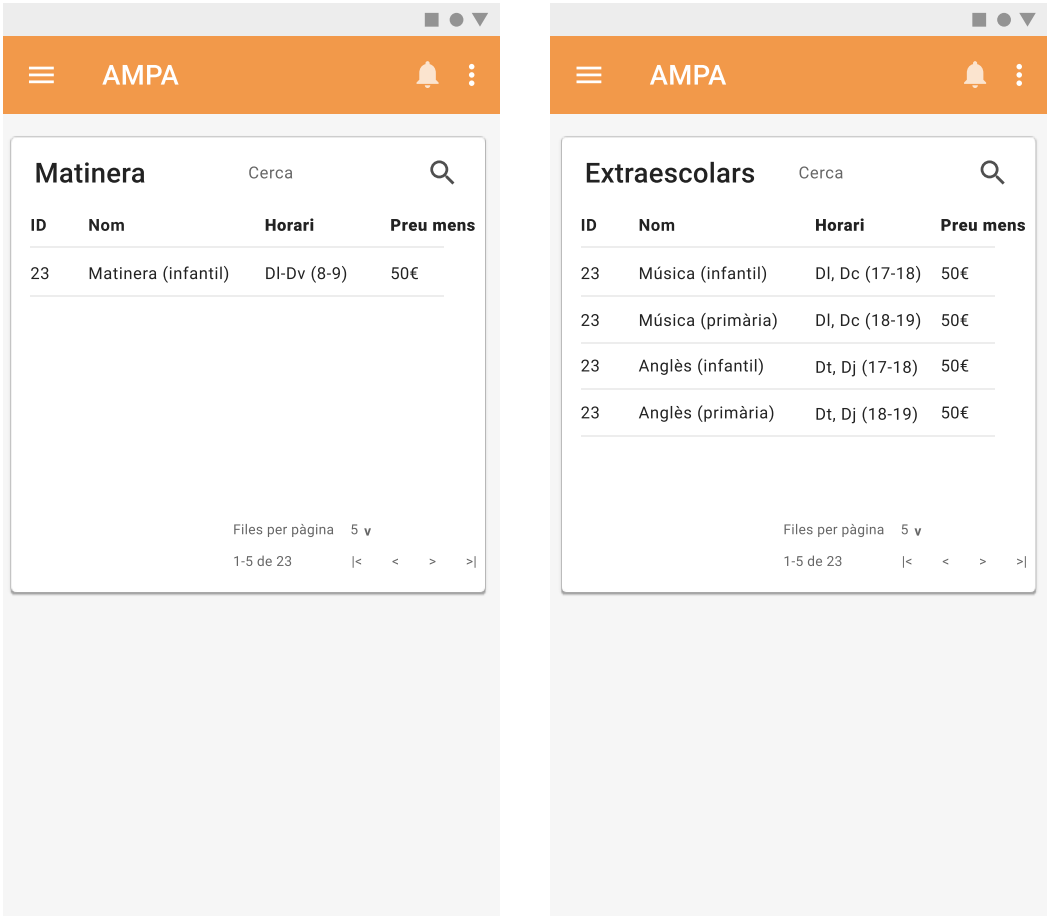


Figura 9.4: Mockup: Blog

9.3 Desenvolupament del *backend*

Per al control de versions usarem git. El codi font està allotjat a: https://github.com/fampa/ampa-graphql

Per gestionar el nostre backend usarem el *framework* Hasura[2]. En el nostre cas l'in-

[2]https://hasura.io/

stal · larem a un servidor VPS mitjançant Docker seguint les instruccions de la seua web.

Una vegada fet això tindrem un *endpoint* de GraphQl preparat per a funcionar.

Per poder fer un seguiment de les migracions i modificacions que li farem a la base de dades usarem Hasura CLI.

Iniciem el nostre projecte amb l'*endpoint* que ens ha donat la instal · lació:

```
hasura init ampa-graphql --endpoint https://db.monjo.xyz
```

Una vegada fet podem crear la nostra primera migració

```
hasura migrate create "init" --from-server --database-name default
```

```
hasura migrate apply --version "<version>" --skip-execution --database-name
default
```

I exportem les metadades:

```
hasura metadata export
```

Amb `hasura console` iniciarem una *GUI* on crearem les taules necessàries i gestionarem els permisos de cada taula/columna, així com les relacions entre elles. Automàticament se'ns crearà una migració amb cada canvi que fem.

Quan completem una fita a la nostra app, podem fer *squash* per unir les migracions:

```
hasura migrate squash --name "<feature-name>" --from <start-migration-version>
--database-name <database-name>
```

```
hasura migrate apply --version "<squash-migration-version>" --skip-execution
--database-name <database-name>
```

9.4 Desenvolupament del *frontend*

El codi font està disponible a https://github.com/fampa/ampa-pwa

En primer lloc hen instal · lat Quasar:

```
yarn global add @quasar/cli
```

I hem creat el projecte:

```
quasar create ampa-pwa
```

Si volem usar Quasar V2.0 necessitarem un pas extra ja que a la data d'escriure aquesta documentació aquesta versió no es troba a la branca principal.

```
yarn upgrade quasar@next
```

Finalment executem `yarn` per instal · lar totes les dependències.

Afegim suport per dotenv: `yarn add --dev dotenv`, i editem el fitxer `/quasar.conf.js`:

```
build: {
  env: require('dotenv').config().parsed
}
```

Seguint les recomanacions a la web oficial de Quasar, he instal · lat la versió 2, malgrat que encara es troba en beta. Això ha requerit certs *workarounds* com ara els proposats a aquest *issue* de github per fer funcionar la *store* de Vuex: [https://github.com/quasarframework/quasar/issues/8500]

9.4.1 Components

Sempre que un grup de codi es repeteix en més d'un lloc de la nostra aplicació, o si volem separar diferents funcions en diferents arxius, sol ser un bon indicador de que podríem fer ús d'un component.

Ací van alguns exemples

9.4.1.1 NewsCard

- src/components/NewsCard.vue

Com a curiositat, per a que totes les targetes de notícies tinguen la mateixa alçada independentment de la mida del titular he optat per una solució via css per fer elipsis del titular quan sobrepassa 2 línies:

```
.titular {
  display: inline-block;
  height: 65px;
  overflow: hidden;
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: 2;
  text-overflow: ellipsis;
}
```

9.4.1.2 TranslationEditor

- src/components/TranslationEditor

Aquest és un dels components (o component de components) del que em sent més orgullós, ja que ens permet tindre el nostre propi CMS[3], per editar no només els articles del blog sino tots els continguts dinàmics de la web (pàgines, serveis, etiquetes,...).

Per a editar el contingut HTML he optat per un sistema WYSIWYG[4] amb la llibreria `tiptap`.

També he implementat un *uploader* d'imatges al servei en el núvol Firebase Storage.

9.4.1.3 PushToggle

- src/components/PushToggle.vue

Aquest és un altre component important ja que permet la subscripció del dispositiu de l'usuari per a rebre notificacions push.

Per a fer-ho usa la web API nativa per a obtenir el permís de l'usuari, crida a Firebase Messaging per obtenir el token de subscripció i l'envia a la nostra base de dades per poder-lo usar posteriorment.

```
const subscribe = () => {
      loading.value = true
      console.log('subscription initiated')
      const messaging = firebase.messaging()
```

[3] de les segles en anglés Content Management System
[4] de les segles en anglès What You See Is What You Get

```
    Notification.requestPermission()
      .then(async function (result) {
        if (result === 'granted') {
          await navigator.serviceWorker.ready
            .then(function (registration) {
              registration.showNotification('AMPA', {
                body: 'Molt be! Ja pots rebre notificacions push',
                icon: '/icons/icon-192x192.png',
                requireInteraction: true,
                badge: '/icons/icon-512x512.png',
                vibrate: [200, 100, 200, 100, 200, 100, 200],
                tag: 'ampa-tag'
              })
            })
        }
        return messaging.getToken()
          .then(async function (currentToken) {
            if (currentToken) {
              await sendTokenToServer(currentToken)
              $q.localStorage.set('pushToken', currentToken)
              emit('pushToken', currentToken)
              updateUIForPushEnabled()
              loading.value = false
            } else {
              // Show permission request.
              console.log('No Instance ID token available')
              // Show permission UI.
```

```
                updateUIForPushPermissionRequired()
                loading.value = false
              }
            })
            .catch(function (err) {
              console.error('Error retrieving token', err)
              loading.value = false
            })
        })
        .catch(function (err) {
          console.error('Unable to get permission', err)
        })
    }
```

9.4.2 Client de GraphQL

Afegim support per a *GraphQl* amb *Apollo client*.

```
yarn add @vue/apollo-composable
```

Per a poder usar queries en fitxers independents en format `.gql` necessitem afegir aquest codi a `src/shims-vue.d.ts`:

```
declare module '*.gql' {
  import { DocumentNode } from 'graphql'

  const content: DocumentNode
```

```
  export default content
}
```

I a `quasar.conf.js`:

```
chainWebpack (chain /** { isServer, isClient } **/) {
  // ...
  chain.module.rule('gql')
          .test(/\.(graphql|gql)$/)
          .use('graphql-tag/loader')
          .loader('graphql-tag/loader')
  // ...
}
```

Els articles del blog els carregarem paginant-los amb la tècnica del *infinite scroll*. Per a això al *frontend* usarem:

```
<div v-else-if="articles">
  <q-infinite-scroll :offset="0" @load="onLoad">
    <div class="row items-start">
      <div
        class="col-12 col-sm-6 col-md-4 q-pa-sm"
        v-for="article in articles"
        :key="article.id"
      >
        <news-card :article="article"></news-card>
      </div>
    </div>
```

```
    <template v-if="loading" v-slot:loading>
      <div class="row justify-center q-my-md">
        <q-spinner-dots color="primary" size="40px" />
      </div>
    </template>
  </q-infinite-scroll>
</div>
```

I a Apollo:

```
import { offsetLimitPagination } from '@apollo/client/utilities'


const cache = new InMemoryCache({
  typePolicies: {
    Query: {
      fields: {
        articles: offsetLimitPagination()
      }
    }
  }
})
```

```
query getArticles($offset: Int, $limit: Int) {
    articles(
      offset: $offset,
      limit: $limit,
      order_by: {created_at: desc}
    ) {
```

```
    id
    status
    image
    created_at
    updated_at
        translations {
            title
            slug
            language
        }
    }
}
```

9.4.3 Gestió d'estat (Vuex)

Per a controlar l'estat de la nostra aplicació usem Vuex, instal·lat durant la configuració inicial de Quasar. Per a que l'estat persisteixi usarem la llibreria `vuex-persistedstate`:

```
yarn add vuex-persistedstate
```

I afegirem ho configurem a Vuex al fitxer `src/store/index.ts`:

```
import createPersistedState from 'vuex-persistedstate'


const store = createStore({
  // ...
```

```
  plugins: [createPersistedState()]
});
```

9.4.4 Autenticació

Usarem Firebase per tasques d'autorització i autenticació.

El primer pas és crear un projecte a (https://console.firebase.google.com/u/0/?pli=1).

També usarem Firebase Hosting per a allotjar la nostra app posteriorment, així com Cloud Functions per a usar com a servidor nodejs, Firebase Storage per allotjar imatges pujades pels usuaris.

Afegim Firebase a la nostra app:

```
yarn add firebase
```

També necessitarem el cli de Firebase al nostre sistema:

```
npm install -g firebase-tools
```

Configurarem les credencials de Firebase amb variables ambientals, de manera que els desenvolupadors només hauran d'omplir les dades del fitxer `.env`.

Per al *flow* d'autenticació usarem la llibreria FirebaseUi:

```
yarn add firebaseui
```

9.4.5 Autorització

Assegurarem la nostra aplicació mitjançant mecanismes d'autorització tant en el *backend* com en el *frontend*

9.4.5.1 Backend

Hasura proporciona un sistema nadiu d'autorització granular que permet gestionar permisos d'accés tant a nivell de taules, com de columnes com de camps individuals, tal i com es pot veure a la fig. 9.5

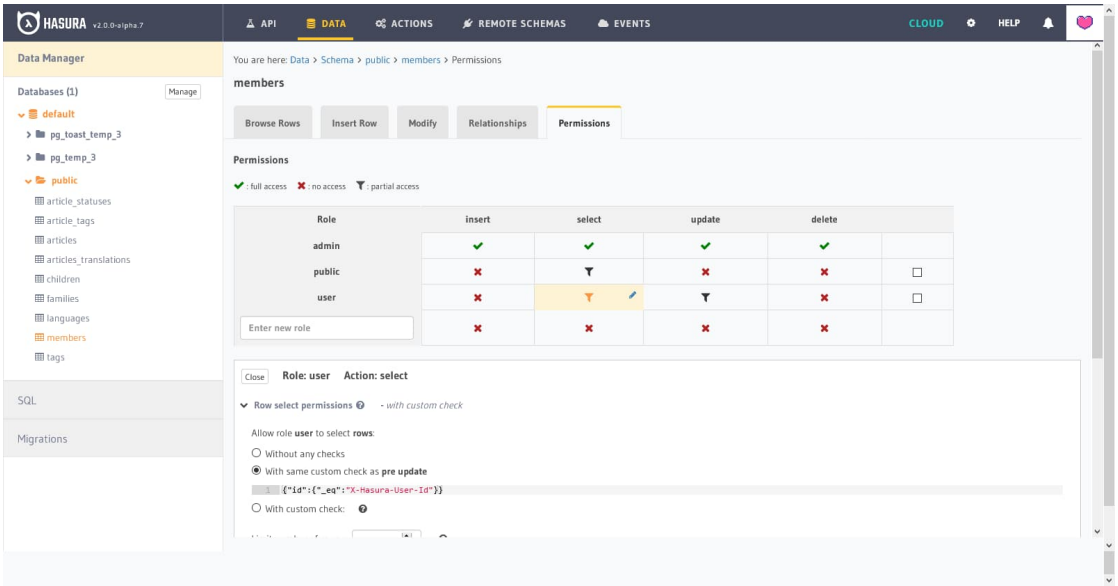


Figura 9.5: Sistema d'autorització de Hasura

9.4.5.2 Frontend

Per a controlar l'autorització al *frontend* usarem Vue Router. Concretament afegirem els requisits d'accés al paràmetre meta de la ruta dins de `src/router/routes.ts`

```
{
    path: '/user',
    meta: { requiresAuth: true },
    // ...
},
{
    path: '/admin',
    meta: { requiresScope: 'admin' },
    // ...
}
```

Això ens permetrà usar el mètode `router.beforeEach` per comprovar si eixa ruta està protegida i comparar amb el nivell d'autorització de l'usuari.

9.4.6 PWA

Afegim suport per a PWA.

```
quasar mode add pwa
```

Això crearà una carpeta `src-pwa` amb el codi font del *service worker*, i les icones necessàries a la carpeta `public`.

Per a actualitzar el *service worker* cada vegada que es publique una nova versió de la app necessitem modificar `src-pwa/register-service-worker.js`:

```
import { Notify } from 'quasar'
// ...
updated (registration) {
  console.log('New content is available; please refresh.')
  return Notify.create({
    timeout: 0,
    message: 'Nova versió disponible. Actualitza polsant el botó',
    actions: [
      {
        icon: 'fal fa-sync-alt',
        label: 'ACTUALITZA',
        handler: () => {
          return registration.waiting.postMessage('skipWaiting')
        }
      }]
  })
},
// ...
```

Això ens mostrarà una notificació que ens donarà l'opció d'actualitzar enviant el missatge 'skipWaiting' al *service worker*.

També necessitarem escoltar a aquest missatge, per això editarem el fitxer `src-pwa/custom-service-worker.js`:

```
self.addEventListener('message', e => {
  if (e.data === 'skipWaiting') {
    console.log('skipWaiting called')
```

```
    self.skipWaiting()
  }
})
```

Per a generar les icones necessàries de manera automàtica usarem esta eina:

```
yarn global add @quasar/icongenie
```

Per generar les icones:

```
icongenie generate -i src/assets/icon.png
```

9.4.7 Llibreria d'elaboració pròpia

Per a la validació del dni faig ús d'una llibreria anomenada `spain-id` que vaig elaborar jo mateix i que està disponible a https://www.npmjs.com/package/spain-id.

9.4.8 Estudi d'usabilitat / Experiència d'usuari (UX)

9.4.8.1 Persona

A l'annexe 2 podem veure un estudi de *persona* on analitzem l'usuari arquetípic de la app, així com els usos que en farà d'aquesta.

9.4.8.2 Experiència onboarding

S'ha procurat que el fluxe de registre d'usuaris tinga el mínim de fricció possible. Per a això, a priori no apareix cap botó de crida a l'acció per a l'alta i s'ha substituït per tres senzills botons per fer login o *sign-up*.

La resta de dades necessàries es pregunten a una nova pantalla evitant així una reacció adversa que presentaria mostrar el formulari sencer de primeres (Enders 2016).

9.4.8.3 Velocitat de càrrega percebuda

És important que una aplicació carregue de manera ràpida, però això no sempre és possible. Per exemple, necessitem cridar a una api per obtenir les dades, o necessitem carregar imatges o altres fonts multimèdia.

Tradicionalment aquest aspecte s'ha solucionat amb indicadors de càrrega com ara spinners. No obstant això, aquest indicadors s'han anat substituint per altres elements que fan que la percepció de la velocitat de càrrega siga major, com ara el conegut *skeleton*. Una tècnica usada per exemple per pàgines com ara facebook o la cerca de Google (Wroblewski 2013).

9.4.8.4 Targetes (cards)

L'ús de targetes està destinat a resoldre el problema d'haver de presentar informació d'una manera resumida, ampliable al fer click sobre ella o sobre un element intern d'aquesta.[5]

En aquesta app fem ús de les targetes per presentar les entrades del blog, per exemple.

[5] http://ui-patterns.com/patterns/cards/

9.4.8.5 Scroll infinit

Quan tenim moltes dades que carregar necessitem algun mecanisme de paginació. Un d'ells és el conegut com a *infinite scroll* que proporciona una manera còmoda d'anar carregant nous continguts a l'usuari a mesura que fa *scroll* a la nostra app.[6]

Ho hem usat per carregar articles del blog a la portada.

9.4.9 Optimitzacions

S'han realitzat proves de rendiment amb Lighthouse[7], per avaluar el rendiment de l'aplicació així com les bones pràctiques pel que fa accessibilitat, SEO, PWA i d'altres.

La prova s'ha realitzat sobre l'opció mòbil (Moto G4 emulat), i els resultats han estat els següents:

9.4.9.1 Resultat preliminar

A la fig. 9.6 tenim els resultats preliminars de l'aplicació a Lighthouse abans de fer cap intervenció específica.

9.4.9.2 Resultat post intervenció

Com es pot observar a la fig. 9.7 s'ha intervingut en l'aspecte SEO corregint el fitxer `robots.txt`. També s'ha intervengut en aspectes d'accessibilitat canviant els colors principals del lloc per tal de facilitar el contrast i per tant la lectura a persones amb

[6] http://ui-patterns.com/patterns/ContinuousScrolling
[7] https://developers.google.com/web/tools/lighthouse

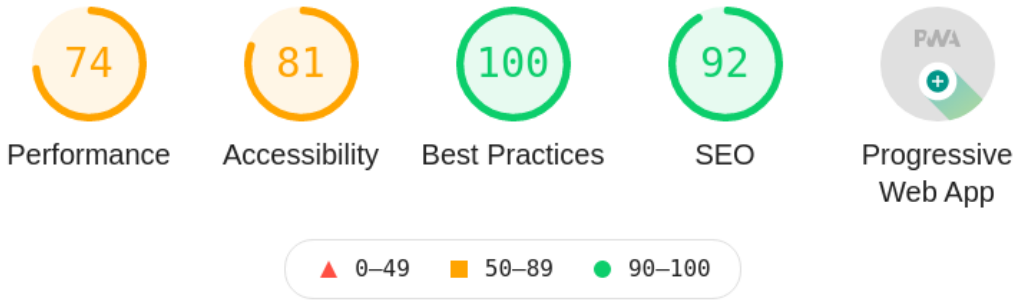


Figura 9.6: Resultat preliminar de les proves amb Lighthouse

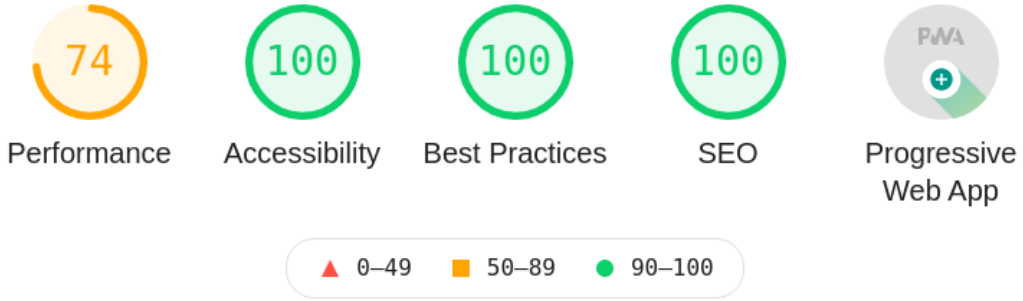


Figura 9.7: Resultat post intervenció de les proves amb Lighthouse

dificultats de visió. També s'han afegit propietats `aria-label` a alguns botons per facilitar la lectura de pantalla per part d'invidents.

Amb aquests canvis hem obtingut la màxima puntuació en accessibilitat, millors pràctiques, SEO i PWA.

L'apartat que queda pendent de millorar és el de rendiment ja que al moment de redactar aquest informe Quasar v2 no permet configurar `rel=preload` ni altres millores com eliminar el javascript i el css no usat sense trencar l'aplicació degut a que encara s'està treballant en la integració d'aquest framework amb Webpack v4.

10 Conclusions

Aquest màster m'ha fet millorar com a desenvolupador en diferents aspectes. D'una banda he refrescat coneixements que ja tenia adquirits com pot ser el desenvolupament en javascript, html i css.

Però també he aprofundit en d'altres que fins ara havia passat un poc de puntetes, com pot ser el TDD i el testing en general, l'ús d'Angular i de TypeScript.

En aquest projecte he pogut aplicar les metodologies apreses, sobretot pel que fa a la planificació del projecte, el disseny de mockups, la separació del projecte en unitats lògiques (ui, serveis, components, business logic, etc,...).

Per tant puc concloure que ha pagat la pena la realització del màster i considere que surto d'ell com a un millor desenvolupador del ho era abans de fer-lo.

Referències

Britz, David. 2017. *Enterprise Application Patterns using Xamarin.Forms eBook - Xamarin | Microsoft Docs*. Microsoft. https://raw.githubusercontent.com/dotnet-architecture/eBooks/master/current/xamarin-forms/Enterprise-Application-Patterns-using-XamarinForms.pdf.

Centro de Investigaciones Sociológicas. 2017. «Barómetro de febrero de 2017». Centro de Investigaciones Sociológicas. http://www.cis.es/cis/export/sites/default/-Archivos/Marginales/3160_3179/3168/es3168mar.pdf.

Enders, Jessica. 2016. *Designing UX: Forms: Create Forms That Don't Drive Your Users Crazy*. SitePoint.

Richard, Sam, i Pete LePage. 2020. «What Are Progressive Web Apps? Web.dev». 2020. https://web.dev/what-are-pwas/.

Rungta, Krishna. s.d. «MVC vs MVVM: Key Differences with Examples. Guru 99». Consulta 2 març 2021. https://www.guru99.com/mvc-vs-mvvm.html.

Wroblewski, Luke. 2013. «Mobile Design Details: Avoid The Spinner». 7 setembre 2013. http://www.lukew.com.