UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

VoluntariApp

Disseny i desenvolupament d'una aplicació per a fer voluntariats amb gamificació.

Enginyeria del Software

Arnau Lamiel Sarasa

Facultat d'Informàtica de Barcelona

Director: Xavier Oriol Hilari

Codirectora: Alba Linares Griera

Lectura: 25 de gener 2022

Agraïments

Aquest treball no hauria estat possible sense l'ajuda i el suport de vàries persones a les quals vull agrair explícitament tot el que han fet per mi i pel projecte. Sobretot, vull aprofitar aquestes línies per donar-los el reconeixement que es mereixen.

A la primera persona que he d'esmentar és a la meva parella, la Clàudia. M'ha ajudat i m'ha fet costat durant tots aquests mesos tan intensos, no només en la part emocional, sinó també en la part escrita del treball. M'ha donat consells i assessorament lingüístic sempre que ho he requerit, a més, sempre m'ha ajudat en tot el que ha pogut i s'ha interessat en tot moment pel meu projecte.

A la meva mare, per ser-hi sempre, per educar-me sola des que tinc ús de raó i per confiar sempre en mi i en totes les meves decisions. Gràcies al seu suport he aconseguit fer l'últim pas de la meva etapa universitària amb ganes i confiança.

Al meu tutor del TFG, el Xavier Oriol Hilari. Primer per mostrar-me l'Enginyeria del Software com quelcom seductor i al qual voler-me dedicar. I segon per guiar-me, corregir-me, ajudar-me i aconsellar-me en l'actual projecte en el transcurs dels darrers cinc mesos.

Al Xavier Sierra, per oferir-me bons consells sempre que els hi he demanat, per les converses i debats sobre software, i per a estar sempre disposat a ajudar-me.

A la Alba Linares i a la Cristina Caña, per a fer de pont des de DAMM i oferir-se sempre a col·laborar amb tot el que estava a les seves mans, ajudant-nos en la presa de requisits i la validació de l'aplicació.

Al Gerard Gimeno, un molt bon amic que s'ha interessat per la informàtica a través del meu TFG,amb qui gaudeixo ensenyant-li tot allò que he après.

A tots aquells companys que m'han fet gaudir de l'etapa universitària, i han acabat essent amistats que difícilment podré oblidar. I a tots aquells professors que m'han fet aprendre de veritat, sobretot els de l'especialitat d'Enginyeria del Software.

I finalment, a tothom que s'ha involucrat i ha participat en aquest treball de forma directa, com el Nando, l'Umair, i l'Àlvaro. Però també a qui ho ha fet d'una forma més indirecta, com els enquestats.

Resum

Per desgràcia, les desigualtats socials i els problemes mediambientals en les societats capitalistes modernes no han parat d'augmentar, sobretot en els últims anys i en part per culpa de la pandèmia de la COVID-19. Per aquest motiu, i davant la inacció dels governs, molts individus amb consciència col·lectiva s'han interessat a ajudar voluntàriament als més desafavorits, al medi ambient, als animals…
Aquestes persones, anomenats voluntaris, necessiten una forma d'organitzar-se perquè l'ajuda sigui més efectiva, o tenir unes organitzacions que els facin aquesta feina. Per això, moltes ONG i empreses que volen ajudar activament a la societat, compleixen aquest paper.

Aquest document té com a principal objectiu dissenyar i desenvolupar una aplicació web que serveixi per fer de punt de trobada entre les organitzacions i les persones, centralitzar la informació sobre aquestes pràctiques de voluntariat, i oferir una incentivació extra per a fer-los a base de gamificació. A més, afegir funcionalitats extra que permetin als usuaris decidir triar aquesta plataforma davant la competència.

Aquest projecte s'ha dut a terme mitjançant metodologies àgils, específicament *Scrum*. S'han fet quatre iteracions referents als *Sprints* i en cada *Sprint* hem inclòs un gruix de tasques a dur a terme.

Com a resultat de la implementació d'aquesta metodologia, s'ha creat una aplicació web visualment atractiva, en la qual els voluntaris trobaran tota mena de voluntariats que podran realitzar i que han estat introduïts per les organitzacions, i pels quals obtindran una recompensa en forma de xapes (moneda virtual), que podran bescanviar per experiències ofertes per altres o la mateixa organització.
Aquesta aplicació s'ha creat seguint les especificacions de l'Enginyeria del Software, utilitzant una arquitectura en tres capes amb *React, Spring Boot i Spring JPA.* I una base de dades amb *PostgreSQL.*

Resumen

Por desgracia, las desigualdades sociales y los problemas medioambientales en las sociedades capitalistas modernas no han parado de aumentar, sobre todo en los últimos años y en parte por culpa de la pandemia del COVID-19. Por ese motivo, y ante la inacción de los gobiernos, muchos individuos con conciencia colectiva se han interesado en ayudar voluntariamente a los más desfavorecidos, al medio ambiente, a los animales…
Estas personas, llamadas voluntarios, necesitan una forma de organizarse para que la ayuda sea más efectiva, o bien tener unas organizaciones que hagan este trabajo. Por eso, muchas ONG y empresas que quieren ayudar activamente a la sociedad, cumplen esta función.

Este documento tiene como principal objetivo diseñar y desarrollar una aplicación web que sirva para hacer de punto de encuentro entre las organizaciones y las personas, centralizar la información sobre los voluntariados, y ofrecer una incentivación extra para hacerlos a base de gamificación. Además, añadir nuevas funcionalidades que permitan a los usuarios escoger esta aplicación por delante de la competencia.

Este proyecto se ha realizado mediante metodologías ágiles, específicamente *Scrum.*
Se han usado cuatro iteraciones referentes a los *Sprints* y en cada uno se han incluido varias tareas que se han tenido que hacer.

Como resultado de la implementación de esta metodología, se ha creado una aplicación web visualmente atractiva en la cual los voluntarios encontrarán todo tipo de voluntariados que podrán realizar y que han sido introducidos previamente por las organizaciones, y por los cuales obtendrán una recompensa en forma de chapas (moneda virtual), que podrán cambiar por experiencias ofrecidas por la misma u otras organizaciones.
Esta aplicación se ha creado siguiendo las especificaciones de la Ingeniería del Software, usando una arquitectura en tres capas con *React, Spring Boot y Spring JPA.* Y una base de datos con *PostgreSQL*.

Abstract

Unfortunately, social inequality and environmental problems in modern capitalist societies have suffered an important increase, especially in past years due to the COVID-19 global pandemic.
This is the reason why, on account of Governments' inactivity, collective responsibility has raised awareness in many people who have shown interest in helping deprived citizens, the environment, the animals…
These people are called volunteers, they need a way of organization to make their assistance more effective, or rely on organizations to get this job done. Therefore, plenty of NGOs and companies, which are open to helping our society actively, fulfill this role.

This paper aims to design and develop a web application that would be useful to connect organizations and people. It centralizes information about voluntary services and offers extra motivation due to its gamification. Furthermore, extra functionalities would be added which would allow users to choose this platform among the competition.

This project has been carried out through agile methodology, specifically *Scrum*. Four *Sprint* referred iterations have been done and a majority of tasks to complete has been included in every Sprint.

As a result of implementing this methodology, we have created a visually attractive web application in which volunteers will find any sort of voluntary service brought in by organizations. They will be able to complete every service and be rewarded with “badges”, a virtual currency that can be exchanged for any experience offered by other people or the organization.
The application has been created following Software Engineering specifications, using a three-layered architecture with *React, Spring Boot*, and *Spring JPA*. And a *PostgreSQL* database.

1. Introducció

1.1 Introducció

VoluntariApp és el meu treball fi de grau (TFG) per a la Facultat d'Informàtica de Barcelona (FIB), de la Universitat Politècnica de Catalunya, concretament per al Grau en Enginyeria Informàtica i per l'especialitat en Enginyeria del *Software*.
Aquest treball està fet en modalitat A, és a dir, és un projecte fet a la universitat, però amb una particularitat: la participació externa de l'empresa DAMM, una empresa cervesera creada el 1876 a Barcelona i que actualment té la intenció d'expandir la seva interacció social i el sentiment de marca.

Aquest treball és una continuació de la feina feta a l'assignatura de Projectes Aplicats d'Enginyeria (PAE) de la FIB, feta conjuntament amb els estudiants Fernando Marimon, Álvaro Antonio Gil i Umair Tehami.
A PAE l'objectiu principal era agafar un repte que ens oferís una empresa (en el nostre cas DAMM) i transformar-lo en una idea concisa per aplicar-la a l'enginyeria informàtica. En aquest projecte tractarem de dissenyar i desenvolupar la idea amb les modificacions pertinents per adaptar-ho a la complexitat d'un TFG.

Aquesta idea permet a qualsevol Voluntari que utilitzi *VoluntariApp*, poder buscar, trobar i fer els Voluntariats que li interessin, i obtenir Xapes per a poder bescanviar-les per les Experiències que li interessin. Alhora, fer Feedback per a puntuar ambdós Esdeveniments.

Per a tenir clar des d'on parteix aquest treball, hem d'esmentar que: tant l'idea general de l'aplicació, com la cerca inicial de l'estat de l'art, com la realització d'alguns *mock-up i com el* desenvolupament bàsic d'alguna pantalla (especificat a l'apartat 7.2.1)*,* es va decidir i desenvolupar a PAE.
Com en aquests fets anteriorment anomenats, hi ha petits detalls de l'aplicació com la decisió dels llenguatges de programació que també es van decidir a PAE.
Per altre banda, el que ens hem encarregat de fer en aquest treball és aplicar la idea primitiva de PAE, modificar-la conforme als requisits per reestructurar la solució, i dissenyar-la i desenvolupar-la com a aplicació web, duent aquesta solució a la pràctica .

1.2 Identificació del Problema

Com hem esmentat anteriorment, vam tenir un repte a solucionar. DAMM ens va proposar la creació d'una plataforma tecnològica que servís per fomentar el voluntariat i que fes de pont per establir una unió més propera entre les organitzacions i les persones.

Amb aquest repte, ens vam posar a investigar, i parlant amb diferents empreses, vam veure que existia un problema generalitzat en aquest aspecte.
Per tant, ens trobem que existeix la necessitat per part de moltes empreses, ja que la DAMM no era l'única, d'ampliar el sentiment de marca mitjançant obres socials. A PAE vam

poder observar que aquesta necessitat era quelcom primordial per l'àrea de Responsabilitat Social Corporativa (RSC) de totes les empreses.

Sumat a això, també vam fer unes enquestes[1] dirigides a companys d'universitat i de feina, amics i familiars, familiaritzats o no amb els voluntariats, per conèixer les preocupacions de l'altra part dels interessats. Amb això vam veure que per part seva, també volien una plataforma intuïtiva i fàcil que els permetés fer voluntariats de diferents tipus on la informació estigués centralitzada i no haguessin de buscar a diversos llocs depenent del tipus de voluntariat.

Tenint això en compte i centrant-nos en l'àrea social dels voluntariats, ens vam trobar que en una època totalment digitalitzada, i amb la facilitat que ens otorga internet, no hi havia opcions al mercat que solucionessin de manera adequada aquest problema (ho veurem amb més profunditat a l'apartat de l'Estat de l'Art).

Un altre problema que trobem en l'època actual, on acabem de passar per una pandèmia mundial (COVID-19) que sembla que ha canviat la forma de viure i treballar de les persones, és que les plataformes existents, a part del fet mencionat anteriorment, no afronten aquesta problemàtica amb visió pandèmica. No s'han adaptat a les noves formes de viure de la gent i, per tant, s'han estancat i han oblidat que mitjançant internet també s'hi poden fer voluntariats i altres esdeveniments.

Així doncs, vam idear una solució que incrementaria, per una part, el sentiment de marca, que tant els interessa a les organitzacions, en els voluntaris. I per una altra, la facilitat de cerca i realització de voluntariats, que interessa als possibles voluntaris.
Tot això, en una plataforma afegint i millorant els sistemes actuals, i mitjançant una idea puntera: la gamificació i compensació de les accions solidàries fetes pels voluntaris mitjançant experiències. Tot a partir de xapes virtuals obtingudes als voluntariats i que es poden bescanviar per experiències.

1.3 Motivació

Créixer en un món tan injust ens ha fet veure des de petits que les desigualtats socials del nostre entorn són notories i cal posar-hi solució.
Nosaltres, com a individus inclosos dins d'aquesta, se'ns ha assignat un rol passiu en el que, si volem millorar-la, hem de delegar les nostres idees als partits polítics, escollint la ideologia que més s'assimili als nostres pensaments.

D'aquesta manera no tenim un rol actiu participant directament en les preses de decisions, cosa que genera una frustració en moltes persones ja que les desigualtats no acaben desapareixent, per que les polítiques que es fan són, en molts casos, pactes entre ideologies per a fer petites reformes.

Per aquest motiu, i en aquest sistema social en el que vivim, ens neix una necessitat d'acabar amb aquesta desigualtat de forma en la que hi participem activament.

Com això és impossible sense atacar l'arrel del problema, existeix una forma d'ajudar a la societat des de les bases, ajudant per voluntat pròpia en tots aquells aspectes que no s'han resolt per problemes sistemàtics.
Aquesta forma d'ajudar són els voluntariats.

Per aquest motiu, i al veure la proposta que se'ns va oferir a PAE, em vaig decidir a desenvolupar i dissenyar a fons aquesta plataforma. Era una manera de tenir un impacte social positiu i servia per ajudar a tota la gent que volia ajudar els altres i millorar aquesta societat.
Aquesta era i segueix sent la meva màxima motivació per a aquest Treball Fi de Grau.

Alhora, aprendre noves tecnologies i dominar-les també m'interessava molt per al meu desenvolupament personal com a Enginyer Informàtic.
D'aquesta manera, podia tenir un *background* més extens per a poder treballar en diferents àmbits dins de l'Enginyeria del Software, des de programador coneixent tecnologies noves (ja sigui *full-stack, front-end* o *back-end developer*), fins a Arquitecte del Software o Enginyer de Requisits, perfeccionant i posant en pràctica tot allò après en les assignatures del grau que tant m'interessaven.

Aquesta barreja d'interessos m'ha fet decantar-me per a la realització d'aquest projecte.

2. Contextualització

2.1 Termes i conceptes relacionats amb el projecte

Per tal d'entendre els principals termes que s'utilitzaran en aquest treball, es passen a definir:

1. **Voluntariat:** Acció voluntària feta per una persona per ajudar els altres o la comunitat. Per exemple, anar a recollir brossa a la platja o quelcom semblant. No consideraríem voluntariats a gran escala com un viatge a un país subdesenvolupat per ajudar la població.

2. **Voluntari:** Persona que realitza un voluntariat.

3. **Xapes:** Moneda virtual de la plataforma que s'obté en fer voluntariats. Cada voluntariat dona un nombre de xapes, corresponent a la implicació que requereix.

4. **Experiència:** Recompensa en forma d'activitat acomplerta per una organització que es pot bescanviar per xapes obtingudes en un voluntariat. Per exemple, una entrada a un festival amb patrocini de l'organització que l'ofereix, una visita a una fàbrica, un esdeveniment esportiu, etc.

5. **Esdeveniment:** Un voluntariat o una experiència.

6. **Feedback**: Puntuació que se li otorga a un Esdeveniment per part d'un Voluntari.

2.2 Parts interessades o Stakeholders

Els *stakeholders* són els actors o parts implicades en el projecte. És a dir, les parts que estan afectades per la realització d'aquest projecte, ja sigui perquè es beneficien de la plataforma, hi estan interessades, afectades o hi participen.
Les llistarem a continuació:

- **Desenvolupadors:** Són els encarregats de desenvolupar la plataforma mitjançant codi. En aquest cas soc exclusivament jo.

- **Voluntaris:** Són una part dels *stakeholders* que se'n beneficiaria, ja que gràcies a aquesta plataforma podrien ajudar la comunitat i trobar voluntariats d'una manera més pràctica i fàcil, amb voluntariats de diferent índole i coordinar-se amb organitzacions i altres voluntaris per a fer grans coses. Alhora es beneficiarien de la gamificació i obtenció d'experiències en fer aquests voluntariats.

- **Organitzacions i Empreses:** Una altra part dels stakeholders que es beneficiaria de la plataforma, ja que són les que oferirien voluntariats i/o experiències per a, aprofitant la seva posició, ajudar a la comunitat, augmentar l'impacte social i el

sentiment de marca.

- **Ajuntaments i Administració:** Aquests stakeholders no són tan principals, però es beneficiarien de segons quins voluntariats, ja que, per exemple, en una recollida de brossa de la platja, s'està fent una labor social que correspondria a les administracions i que disminueix la càrrega de treball i recursos d'aquestes.

- **Veïns i comunitat:** Els veïns i la comunitat en general es beneficiaren de la plataforma, ja que molts voluntariats impacten directament en la vida d'aquestes persones, ja sigui perquè s'ha deixat l'espai públic més net, s'ajuda a persones en risc social, etc. i això fa que augmenti la seva qualitat de vida.

- **Plataformes similars:** Les plataformes similars que investiguem en el següent punt també es consideren stakeholders, ja que es veuen impactades (en aquest cas, negativament ja que augmenta la competència) per la nostra plataforma.

2.3 Estat de l'art

Pel que fa a l'estat de l'art, s'ha fet un estudi de mercat amb els productes relacionats que existeixen actualment que crec que podrien competir amb la idea de VoluntariApp.
La taula que es mostra a continuació es va realitzar durant PAE, en aquest apartat ens centrarem en analitzar-la:

	Objectiu	**Es pot fer feedback?**	**Inclou un sistema de gamificació?**
Welever [2]	Coordina voluntariats en empreses, universitats, ONG y ciutats.	Valoració de les ONG	**NO**
Hacesfalta.org [3]	Posa en contacte voluntaris amb ONG de diferent índole.	Valoració de les ONG	**NO**
Wildlife Volunteer [4]	Voluntariat en l'àmbit mediambiental (centres de rehabilitació, parcs…)	**NO**	**NO**
moviliza-T [5]	Posa en comú la demanda i l'oferta d'iniciatives de voluntariat que existeixen en diferents països.	**NO**	**NO**
GivingWay [6]	Ofereix voluntariat a diferents països segons preferències de l'usuari.	**NO**	**NO**

Figura 1: Taula dels productes relacionats amb el nostre amb dues funcionalitats remarcades.
Font: Elaboració pròpia (PAE).

La majoria de plataformes existents el que fan principalment és actuar de pont entre les organitzacions i els voluntaris, sense anar més enllà, sense que la mateixa plataforma controli la creació dels voluntariats i els seus requisits, i sense que els voluntaris tinguin contacte directe amb les organitzacions.
A més a més, en aquestes plataformes (totes menys Welever) només es tenen en compte els voluntariats d'Organitzacions no governamentals (ONG) i organitzacions supragubernamentals.

Com podem observar a la taula, hi ha dues funcionalitats principals per les quals crec que aquestes plataformes no són suficients per a captar voluntaris i organitzacions:

- La primera de totes és el *feedback*, és a dir, la valoració que un voluntari pot fer sobre un voluntariat que hagi fet. En les alternatives actuals, com a màxim es pot puntuar a les organitzacions que han fet el voluntariat, i res més.
  Això limita molt la informació que rep l'organització i hi impossibilita la possible millora, i no permet al voluntari puntuar més d'un voluntariat, ja que es puntua a l'organització.

- La segona, i pel que VoluntariApp destaca, és que en cap d'aquestes hi ha cap mecanisme per incentivar el voluntariat, com n'és la gamificació.

- Altres coses que no es tenen en compte a la taula però que són prou rellevants són que:

    - En aquestes plataformes no es pot notificar l'assistència a un voluntariat. Això és útil per a les organitzacions per treure mètriques, i en el nostre cas, també per poder certificar l'assistència i poder donar xapes.

    - No hi ha cap mecanisme de comunicació directa entre les organitzacions i els voluntaris per a anunciar els pròxims voluntariats o experiències.

    - No hi ha cap mecanisme de comunicació directe entre les organitzacions i els voluntaris mentre s'està fent el voluntariat per a notificar els canvis que es puguin produir en viu al voluntariat.

A més, aquestes plataformes són, en la seva majoria, aplicacions de mòbil pensant en la comoditat del voluntari.
Per la part de VoluntariApp, i donat el temps que tenim pel desenvolupament, s'ha cregut més convenient que aquesta fos una aplicació web.
Això és principalment per què les organitzacions prefereixen treballar en un ordinador d'escriptori/portàtil per a inscriure voluntariats i experiències, i alhora gestionar les altres funcionalitats d'una manera més fàcil.
A més a més, al voluntari no li comporta cap esforç ni incomoditat extra usar una aplicació web, al contrari, ja que els qui prefereixen accedir-hi des de el mòbil no tindran cap problema i hi podran accedir, però els que prefereixen fer-ho des de un ordinador, tindran llibertat per fer-ho.

3. Abast

3.1 Abast del projecte

Per considerar com a completat aquest treball hauríem de tenir finalitzada una plataforma web, com hem justificat anteriorment, on s'hi pugui fer tot un flux que ens permeti buscar, visualitzar, apuntar, confirmar assistència a voluntariats i experiències i crear-les si ets una organització, i que estigui gamificada per a incentivar la participació activa dels voluntaris i augmentar el sentiment de marca.
D'una manera concreta, ha de comptar amb els següents subobjectius:

- **Crear un sistema de comunicació en temps real entre organitzacions i voluntaris en els voluntariats:** A partir de les notificacions en temps real entre les organitzacions i els voluntaris que estan fent un voluntariat.

- **Poder fer experiències a partir de fer voluntariats (gamificació):** Amb les xapes obtingudes de fer un voluntariat, poder assistir a experiències ofertes per les organitzacions.

- **Poder obtenir mètriques i opinions dels usuaris que han assistit a esdeveniments de les organitzacions:** Mitjançant la confirmació d'assistència als esdeveniments i mitjançant el feedback que puguin haver donat aquests voluntaris .

- **Proporcionar un servei de voluntariats que sempre estigui actiu:** La plataforma ha de permetre la inscripció/realització de voluntariats en qualsevol moment del dia.

3.2 Obstacles i riscos

Per a poder-ho tenir tot sota control en el desenvolupament i disseny d'aquest projecte, hem de tenir en compte que poden haver-hi situacions que ens facin canviar la planificació que teníem en un inici. Aquestes possibles situacions les mostrarem a continuació.

- **Bugs:** Al desenvolupar una solució software és molt probable que vagin sorgint *bugs* que alenteixin el desenvolupament d'aquesta. Aquest obstacle és molt comú i pot arribar a ser molt delicat i retardar molt l'entrega del projecte.

- **Inexperiència en les tecnologies usades:** Aquesta situació també és important a tenir en compte, doncs a l'inici del projecte, si mai s'han utilitzat les tecnologies amb les quals es desenvolupa, hi ha una corba d'aprenentatge que pot fer que avanci molt lentament i endarrereixi l'entrega de la solució.

- **Caiguda del servidor o la Base de Dades (BBDD):** Tot i que no hauria de passar, sempre és possible que el servidor on està allotjada la plataforma o la mateixa

BBDD fallin, sorgint així errors que posen en risc la plataforma sencera.

- **Atac informàtic a la plataforma:** Un altra situació que hem de tenir en compte, encara que potser és la menys probable, és el risc a patir un atac informàtic que deixi sense servei la plataforma, i impossibiliti l'accés o el desenvolupament.

- **Calendari ajustat:** Com aquest projecte és un TFG, hi ha unes dates d'entrega força ajustades, i a la mínima que hi hagi un imprevist, això pot fer que no s'arribi a temps.

3.3 Gestió del risc: Plans alternatius i obstacles

En aquest apartat valorarem els riscos i obstacles observats anteriorment els quals ens poden fer endarrerir o cancel·lar la realització del projecte.
Per a evitar això, valorarem uns plans alternatius per a si ens trobem aquests obstacles. Els mostrarem a continuació:

- **Obstacle 1 - Bugs**
  Aquest obstacle és probable que ens el trobem, i pot arribar a endarrerir molt la data de finalització depenent de la seva magnitud.
  Per a poder evitar això, es faran tests unitaris[7] per a minimitzar l'impacte, i es crearan tasques extres per a solucionar els bugs que apareguin. A la nostra estimació tenim en compte hores extres per a problemes com aquest, per tant, no hauria d'afectar a la duració del projecte.

- **Obstacle 2 - Inexperiència en les tecnologies usades.**
  Aquest obstacle és molt probable que succeeixi, ja que les tecnologies que usarem són noves, i això tindrà un alt impacte en el desenvolupament. Fins que no es dominin aquestes tecnologies, s'han sobreestimat les tasques que les impliquen, estimant més temps del que requeririen si es dominés la tecnologia. Per aquest motiu, no hauria d'afectar a la duració del projecte.

- **Obstacle 3 - Mala estimació de les tasques**
  Podria ser que s'haguessin estimat malament les tasques, i que per aquest motiu, quelcom no s'acabés en el temps estimat. És una situació que és poc probable, però és probable, ja que no som experts en la gestió de projectes, i tindria un impacte mitjà/alt en el projecte.
  Per a poder minimitzar l'impacte d'aquest obstacle, utilitzem *agile* (explicació dels termes a l'apartat 8) i en les reunions de seguiment tals com els *sprint retrospective* podem veure quan això passa, i en el *sprint planning* podem retocar les estimacions de les tasques per a ajustar millor el temps que portarà aquell desenvolupament. A més, podem modificar el *Sprint* amb les tasques inacabades de l'anterior. Això, en el pitjor cas, pot arribar a afectar a la duració del projecte, pero no hauria d'afegir-li més d'una setmana.

- **Obstacle 4 - Calendari ajustat**
  Això és probable que ens succeeixi. Com aquest projecte és un TFG, s'ha d'ajustar a les dates especificades, el que fa que potser no es pugui acabar tota la plenitud del projecte. Tindria un impacte mitjà/alt.
  Per a minimitzar aquest risc, podem adaptar el projecte a les dates que ens venen donades, i descartar funcionalitats que no creguem imprescindibles, per tal de no afectar a la duració estimada inicial del projecte.

4. Requisits

4.1 Requisits funcionals

Abans d'entrar en matèria amb els requisits funcionals, cal dir que la presa de requisits s'ha fet de manera coordinada amb l'empresa DAMM. Han actuat, d'alguna manera, com a Client i Product Owner (si parlem amb nomenclatura Àgil, que descriurem amb més detall al punt 8).
Gràcies a la seva participació s'ha acabat de donar forma l'aplicació des d'un punt de negoci.
Per aquest motiu, hi ha hagut moviments de funcionalitats, algunes han caigut després de valorar la necessitat de cada una dins del negoci, i altres s'han afegit a les que inicialment es van presentar. Tota la presa de decisions en aquest aspecte sempre l'hem tingut nosaltres com a equip, pero valorant molt l'opinió de DAMM.

Un exemple clar va ser la funcionalitat del "rànquing" d'usuaris per xapes, la qual no es veia clara des d'un punt de negoci ja que no es buscaven ambients competitius, sinó col·laboratius, i aquest punt entrava en conflicte amb aquesta idea. Al final per decisió d'ambdues parts va caure.
Un altre exemple de requisit acordat amb DAMM és que, al bescanviar una experiència, el que s'hauria d'enviar per correu és un codi bescanviable a la web que s'indiqui a l'experiència, i no una entrada o qualsevol altre tipus de recompensa, ja que així és més personalitzable.

També s'ha tingut en compte, com hem explicat al principi, l'opinió dels propis voluntaris per a crear requisits, a partir d'una enquesta (explicat al punt 1.2). A partir d'aquí s'han creat requisits que responien a les necessitats dels possibles usuaris.
Per a definir aquests requisits, utilitzarem casos d'ús.

4.1.1 Diagrama de casos d'ús

Per a poder conèixer ràpidament la interacció d'un actor amb un sistema, podem recórrer als casos d'ús. Els casos d'ús vénen donats per les funcionalitats del sistema tal com les percep un actor, que és tota entitat externa al sistema que té relació amb aquest.
En el nostre cas, i per poder imaginar-ho d'una forma més directa, tenim un diagrama de casos d'ús, i seguidament les descripcions Brief Style d'aquests. A l'apartat 5 especificarem i desenvoluparem els més importants i rellevants.

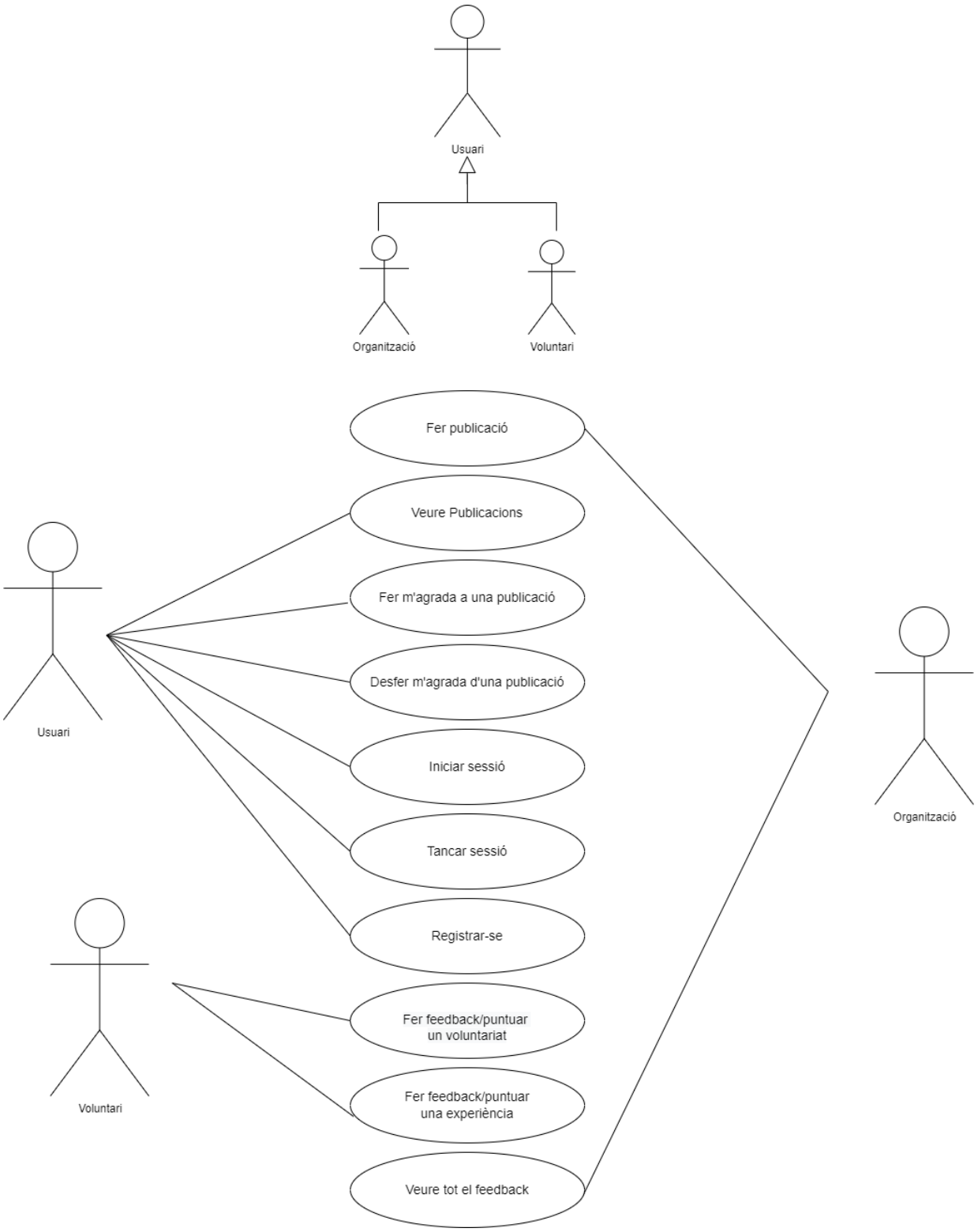


Figura 2: Diagrama de casos d'ús dels requisits funcionals. Part 1.
Font: Elaboració pròpia.

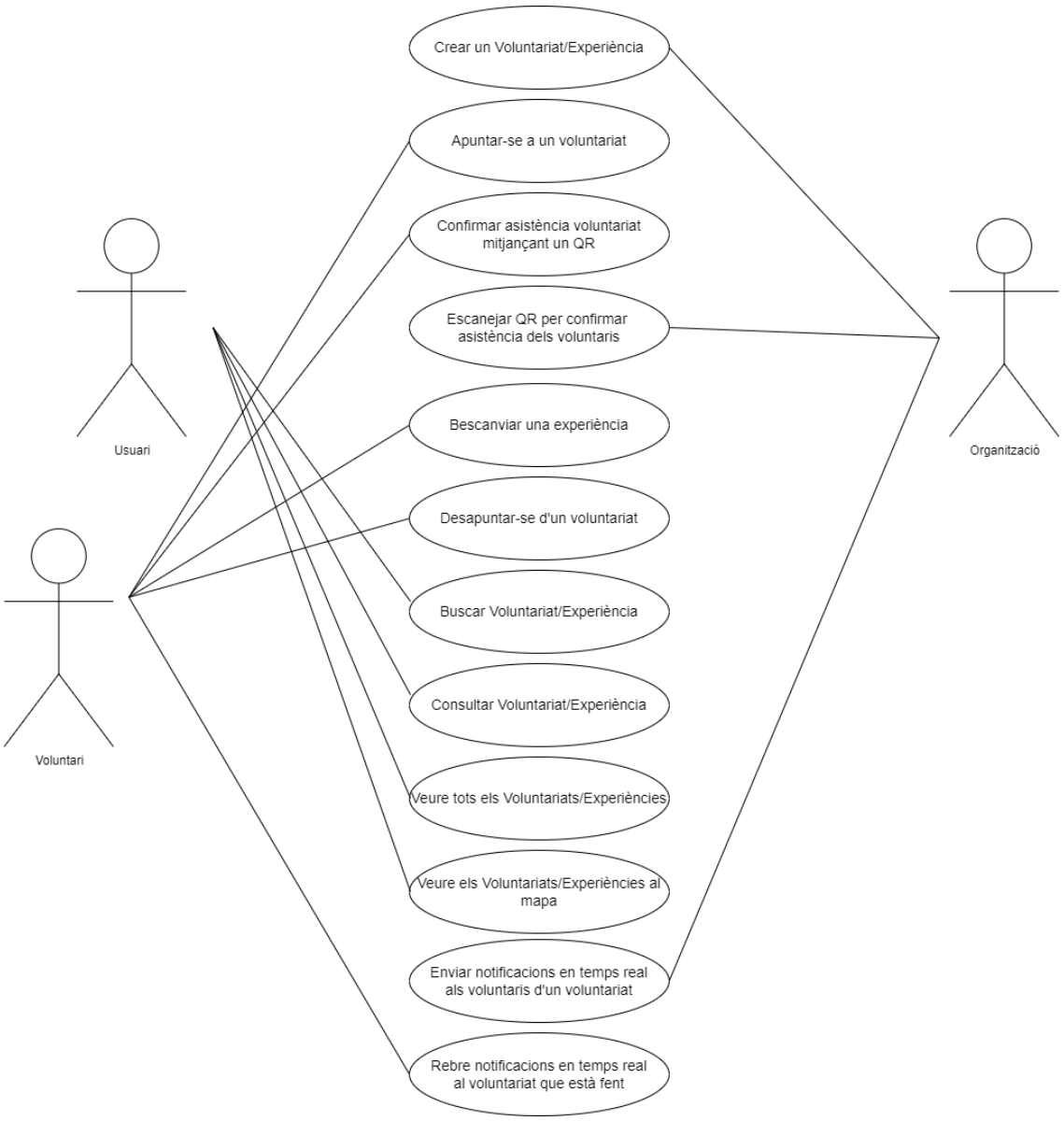


Figura 3: Diagrama de casos d'ús dels requisits funcionals. Part 2.
Font: Elaboració pròpia.

Com podem observar, tenim un sol actor, que és l'usuari, i dos sub-actors que hereden d'aquest Usuari; el Voluntari i la Organització.
Amb l'explicació anterior entenem que, totes les relacions que es facin amb l'actor Usuari, seran relacions alhora amb Voluntari i Organització, mentre que una relació es faci amb un Voluntari no implica que es faci amb una Organització, i viceversa.

4.1.2 Descripció de casos d'ús (Brief Style)

- **Fer publicació:** Una Organització pot fer una publicació que apareixerà a la pàgina principal.
- **Veure publicacions:** Un Usuari pot veure les publicacions de totes les organitzacions.

- **Fer m'agrada a una publicació:** Un Usuari pot fer m'agrada a una publicació d'una Organització que estigui al Sistema.
- **Desfer m'agrada a una publicació:** Un Usuari pot desfer el m'agrada a una publicació d'una Organització que li hagi donat m'agrada anteriorment que estigui al Sistema.
- **Iniciar sessió:** Un Usuari introdueix el seu nom d'usuari i contrasenya i accedeix al Sistema.
- **Tancar sessió:** Un Usuari es desconnecta del sistema.
- **Registrar-se:** Es pot crear un Usuari al sistema per a posteriorment poder Iniciar Sessió.
- **Fer feedback/puntuar un voluntariat:** Un Voluntari que hagi fet un voluntariat pot introduir un feedback i puntuació sobre aquell voluntariat al sistema.
- **Fer feedback/puntuar una experiència:** Un Voluntari que hagi fet una experiència pot introduir un feedback i puntuació sobre aquella experiència al sistema.
- **Crear un Voluntariat/Experiència:** Una Organització pot crear i introduir al sistema un Voluntariat o una Experiència per a que un Voluntari les pugui fer.
- **Apuntar-se a un Voluntariat:** Un Voluntari pot apuntar-se a un Voluntariat del sistema.
- **Confirmar assistència voluntariat mitjançant un QR:** Un Voluntari pot confirmar la seva assistència a un voluntariat ensenyant un QR i obtenir xapes.
- **Escanejar QR per confirmar assistència dels voluntaris:** Una Organització pot confirmar l'assistència a un voluntariat d'un Voluntari escanejant el seu QR.
- **Bescanviar una experiència:** Un Voluntari pot bescanviar una Experiència dins del Sistema si té les xapes suficients.
- **Desapuntar-se d'un Voluntariat:** Un Voluntari pot desapuntar-se d'un Voluntariat del sistema al qual estava apuntat.
- **Buscar Voluntariat/Experiència:** Un Usuari pot buscar entre els diferents Voluntariats/Experiències que existeixen al sistema.
- **Consultar Voluntariat/Experiència:** Un Usuari pot consultar qualsevol dels diferents Voluntariats/Experiències que existeixen al sistema.
- **Veure tots els Voluntariats/Experiències:** Un Usuari pot veure tots els Voluntariats/Experiències que existeixen al sistema.
- **Veure els Voluntariats/Experiències al mapa:** Un Usuari pot veure en un mapa tots els Voluntariats/Experiències que existeixen al sistema.
- **Enviar notificacions en temps real als voluntaris d'un voluntariat:** Una Organització pot enviar notificacions en temps real als voluntaris que hagin confirmat asistència a un voluntariat seu del sistema.
- **Rebre notificacions en temps real al voluntari que està fent:** Un Voluntari pot rebre notificacions en temps real de la Organització que organitza el voluntariat al qual està assistint.

4.2 Requisits no funcionals

Un cop definits els objectius del projecte i els requisits funcionals, passarem a definir els requisits no funcionals amb els que hauria de comptar el projecte per a assegurar-nos un funcionament correcte:

- La plataforma **ha de ser atractiva i agradable a la vista**, per tal de millorar l'experiència d'usuari.

- La plataforma **ha de tenir una resposta ràpida**. Per no impacientar l'usuari, la resposta de la plataforma no hauria de trigar més de 3 segons.

- La plataforma **ha de ser usable**, és a dir, que sigui fàcil d'usar per a tota mena d'usuari, tant si està experimentat amb les noves tecnologies com si no.

- La plataforma **ha d'estar sempre disponible**, per poder accedir als voluntariats en tot moment.

- La plataforma **ha de comptar amb fiabilitat de les dades**, és a dir, les dades han de ser correctes en tot moment.

- La plataforma **ha de ser segura**, és a dir, ha d'assegurar que tothom només pugui veure les seves instàncies, per no tenir conflicte de dades.

- La plataforma **ha de ser escalable**, és a dir, ha de poder adoptar funcionalitats extres en un futur.

- La plataforma **ha de ser compatible** amb qualsevol dispositiu amb accés a la web.

- La plataforma **ha de complir amb les lleis de protecció de dades**.

5. Especificació de Requisits

5.1 Business Process Model (BPM)

Per començar amb l'especificació de requisits, hem desenvolupat un BPM, que no és més que una seqüència ordenada d'activitats de negoci que s'han de dur a terme per a aconseguir un objectiu.
En el nostre cas, el flux de processos que durem a terme per a complir cada requisit.

Per tant, per a descriure aquest BPM, farem un diagrama amb BPMN (Business Process Model Notation), ja que és un estàndard global per modelar processos de negoci.
Analitzant els nostres fluxes, explicarem tres BPMN diferents, un que explica el flux dels voluntariats, desde la creació fins a que un voluntari el fa, un altre que fa el mateix amb les experiències i, finalment, un altre per a les notificacions en temps real.
Aquests tres fluxos són els tres més rellevants per a entendre les funcionalitats donades.

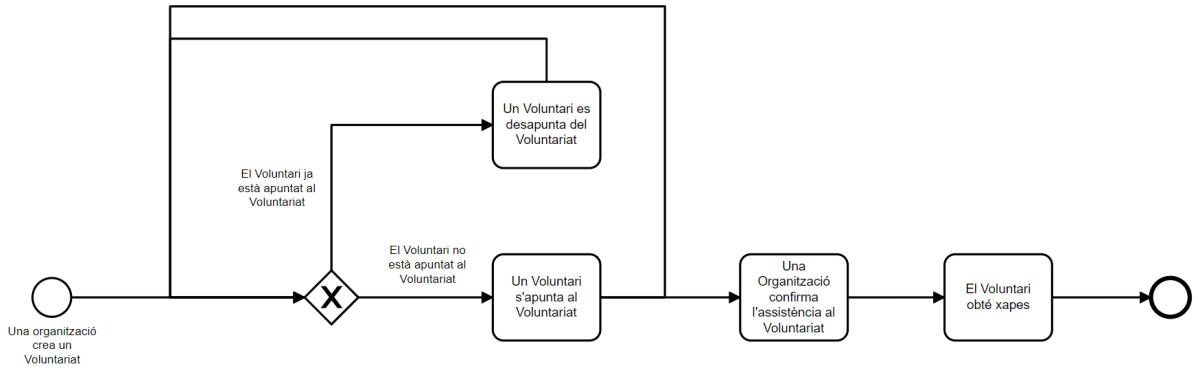


Figura 4: BPMN del flux d'un voluntariat.
Font: Elaboració pròpia.

En el primer flux, el que veiem és que el disparador és que una organització crea un voluntariat. Després d'això, un voluntari, si no està apuntat, es pot apuntar. A continuació, el voluntari pot escanejar el seu QR al lloc on es faci el voluntariat, i així confirma l'assistència al voluntariat i obté xapes.
Si aquest voluntari ja està apuntat al voluntariat, pot desapuntar-s'hi.

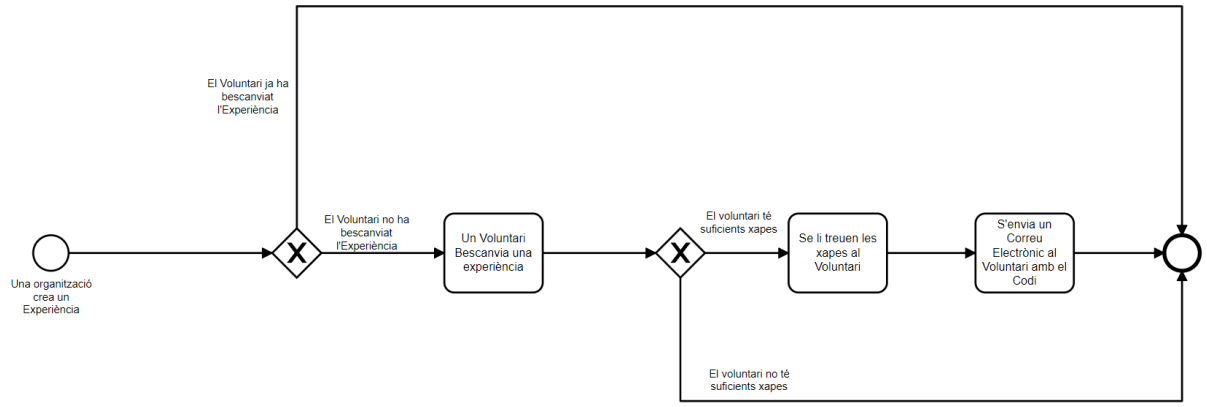


Figura 5: BPMN del flux d'una experiència..
Font: Elaboració pròpia.

En la figura anterior podem veure que, quan una organització crea una experiència, si el voluntari no l'ha bescanviat anteriorment, pot bescanviar-la si té suficients xapes. Si ho fa, se li treuran les xapes corresponents, i després rebrà un correu electrònic amb el codi que pot bescanviar al lloc indicat.
En cas contrari, acabarà el flux.

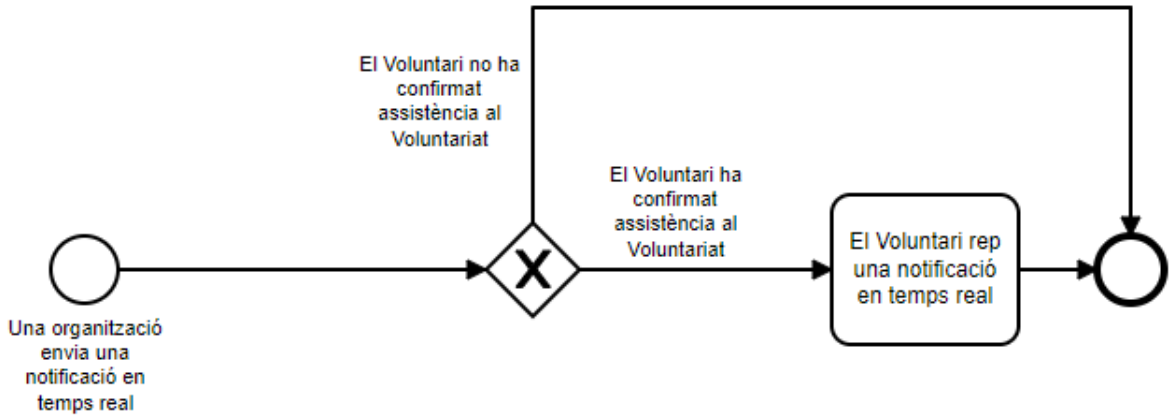


Figura 6: BPMN del flux d'una notificació en temps real..
Font: Elaboració pròpia.

Per acabar, veiem el flux d'enviar una notificació en temps real, i és tan senzill com, quan una organització té algun voluntari amb l'assistència confirmada a aquell voluntariat, pot enviar una notificació en temps real a aquests voluntaris, i la rebran als seus dispositius.

5.2 Especificació completa dels casos d'ús més rellevants

Ara passarem a especificar els casos d'ús que considerem més rellevants, per a conèixer-los més a fons.
Començarem amb la descripció *Brief Style* i seguirem amb l'especificació completa.

Veure els Voluntariats/Experiències al mapa: Un Usuari pot veure en un mapa tots els Voluntariats/Experiències que existeixen al sistema.

Cas d'ús	Veure els Voluntariats / Experiències al mapa	**Actor Principal**	Usuari
Precondició	- L'Usuari ha iniciat sessió al Sistema - El Sistema té almenys un Voluntariat / Experiència no online i amb data posterior o igual a l'actual.		
Disparador	L'Usuari vol veure els Voluntariats o Experiències a un mapa		
Escenari Principal d'Èxit			
1. El Sistema mostra a l'Usuari el llistat de tots els Voluntariats / Experiències registrats al sistema sobre un mapa.			
Extensions			
-			

Apuntar-se a un Voluntariat: Un Voluntari pot apuntar-se a un Voluntariat del sistema.

Cas d'ús	Apuntar-se a un Voluntariat	**Actor Principal**	Voluntari
Precondició	- El Voluntari ha iniciat sessió al Sistema - El Sistema té almenys un Voluntariat amb data i hora posteriors a l'actual. - El Voluntari ha accedit a la Informació del Voluntariat		
Disparador	El Voluntari vol apuntar-se a un Voluntariat per assistir-hi en un futur		
Escenari Principal d'Èxit			
1. El Voluntari selecciona un Voluntariat per apuntar-se. 2. El Sistema apunta al Voluntari al Voluntariat.			
Extensions			

Desapuntar-se d'un Voluntariat: Un Voluntari pot desapuntar-se d'un Voluntariat del sistema.

Cas d'ús	Desapuntar-se d'un Voluntariat	**Actor Principal**	Voluntari
Precondició	- El Voluntari ha iniciat sessió al Sistema - El Sistema té almenys un Voluntariat amb data i hora posteriors a l'actual. - El Voluntari ha accedit a la Informació del Voluntariat - El Voluntari està apuntat al Voluntariat		
Disparador	El Voluntari vol desapuntar-se a un Voluntariat.		
Escenari Principal d'Èxit			
1. El Voluntari selecciona un Voluntariat per desapuntar-se. 2. Abans d'executar l'ordre, el Sistema pregunta si està segur de voler fer l'acció. 3. Un cop acceptat, el Sistema desapunta al Voluntari del Voluntariat.			
Extensions			
1a - El Voluntari no accepta l'advertiment del sistema. 1a1 - Torna al punt 1			

Escanejar QR per confirmar assistència dels voluntaris: Una Organització pot confirmar l'assistència a un voluntariat d'un Voluntari escanejant el seu QR.

Cas d'ús	Escanejar QR per confirmar assistència dels voluntaris	**Actor Principal**	Organització
Precondició	- L'Organització ha iniciat sessió al Sistema. - El Voluntari ha iniciat sessió al Sistema. - El Sistema té almenys un Voluntariat amb data igual a l'actual i de la propietat de l'Organització. - El Voluntari està apuntat al Voluntariat de l'Organització.		
Disparador	Una Organització vol escanejar el QR d'un voluntari per confirmar l'assistència al voluntariat		
Escenari Principal d'Èxit			
1. El Voluntari selecciona el QR d'un Voluntariat en el que està apuntat. 2. L'Organització escaneja el QR del Voluntari. 3. El Sistema confirma l'assistència al Voluntariat del Voluntari			
Extensions			
1a - El QR no es llegeix correctament. 1a1 - S'informa a la Organització. 1a2 - S'acaba el cas d'ús. 2a - El Voluntari ja ha confirmat assistència a aquest Voluntariat 2a1 - S'informa a la Organització. 2a2 - S'acaba el cas d'ús.			

Bescanviar una experiència: Un Voluntari pot bescanviar una Experiència dins del Sistema si té les xapes suficients.

Cas d'ús	Bescanviar una experiència	**Actor Principal**	Voluntari
Precondició	- El Voluntari ha iniciat sessió al Sistema. - El Sistema té almenys una Experiència amb data igual o posterior a l'actual. - El Voluntari ha accedit a la informació de la Experiència.		
Disparador	Un Voluntari vol bescanviar una experiència		
Escenari Principal d'Èxit			
1. El Voluntari selecciona una Experiència per bescanviar-la. 2. Abans d'executar l'ordre, el Sistema pregunta si està segur de voler fer l'acció. 3. Un cop acceptat, si té prou xapes, el Sistema bescanvia l'Experiència. 4. El Sistema envia un correu al Voluntari amb el codi de l'Experiència.			
Extensions			
1a - El Voluntari no accepta l'advertiment del sistema. 1a1 - Torna al punt 1 2a - No hi ha xapes suficients. 2a1 - El Sistema informa al Voluntari que no té xapes suficients. 2a2 - S'acaba el cas d'ús. 3a - El voluntari ja ha bescanviat l'experiència 3a1 - S'informa al Voluntari. 3a2 - S'acaba el cas d'ús. 4a - El correu del Voluntari no existeix. 4a1 - S'acaba el cas d'ús.			

Enviar notificacions en temps real als voluntaris d'un voluntariat: Una Organització pot enviar notificacions en temps real als voluntaris que hagin confirmat asistència a un voluntariat seu del sistema.

Cas d'ús	Enviar notificacions en temps real als voluntaris d'un voluntariat	**Actor Principal**	Organització
Precondició	- L'Organització ha iniciat sessió al Sistema - Els Voluntaris han iniciat sessió al Sistema. - El Sistema té almenys un Voluntariat amb data igual a l'actual, de l'Organització. - Hi ha almenys un voluntari que ha confirmat assistència al Voluntariat.		
Disparador	Una Organització vol enviar notificacions en temps real als voluntaris del seu voluntariat		

Escenari Principal d'Èxit
1. L'Organització selecciona un Voluntariat de la seva propietat. 2. L'Organització escriu una notificació. 3. Abans d'executar l'ordre, el Sistema pregunta si està segur de voler fer l'acció. 4. Un cop acceptat, el Sistema enviarà la notificació a tots els voluntaris que hagin confirmat l'assistència a quell voluntariat.
Extensions
1a - L'Organització no accepta l'advertiment del sistema. 1a1 - Torna al punt 1 2a - El voluntariat ja ha acabat. 2a1 - S'informa a la Organització. 2a2 - S'acaba el cas d'ús.

Rebre notificacions en temps real al voluntari que està fent: Un Voluntari pot rebre notificacions en temps real de la Organització que organitza el voluntariat al qual està assistint.

Cas d'ús	Rebre notificacions en temps real al voluntari que està fent	**Actor Principal**	Voluntari
Precondició	- Els Voluntaris han iniciat sessió al Sistema. - El Sistema té almenys un Voluntariat amb data igual a l'actual. - El Voluntari ha confirmat assistència al Voluntariat.		
Disparador	Una Voluntari vol rebre notificacions en temps real del voluntari que està fent.		
Escenari Principal d'Èxit			
1. El sistema notifica al Voluntari en temps real un cop que hagi assistit al Voluntariat.			
Extensions			
1a - El Voluntariat ja ha acabat. 1a1 - S'acaba el cas d'ús.			

5.3 Diagrama d'especificació

5.3.1 Diagrama

Finalment, i per a poder fer-nos una idea sobre la forma que tindrà el nostre Sistema tenint en compte tots els requisits obtinguts, hem fet un diagrama d'especificació.

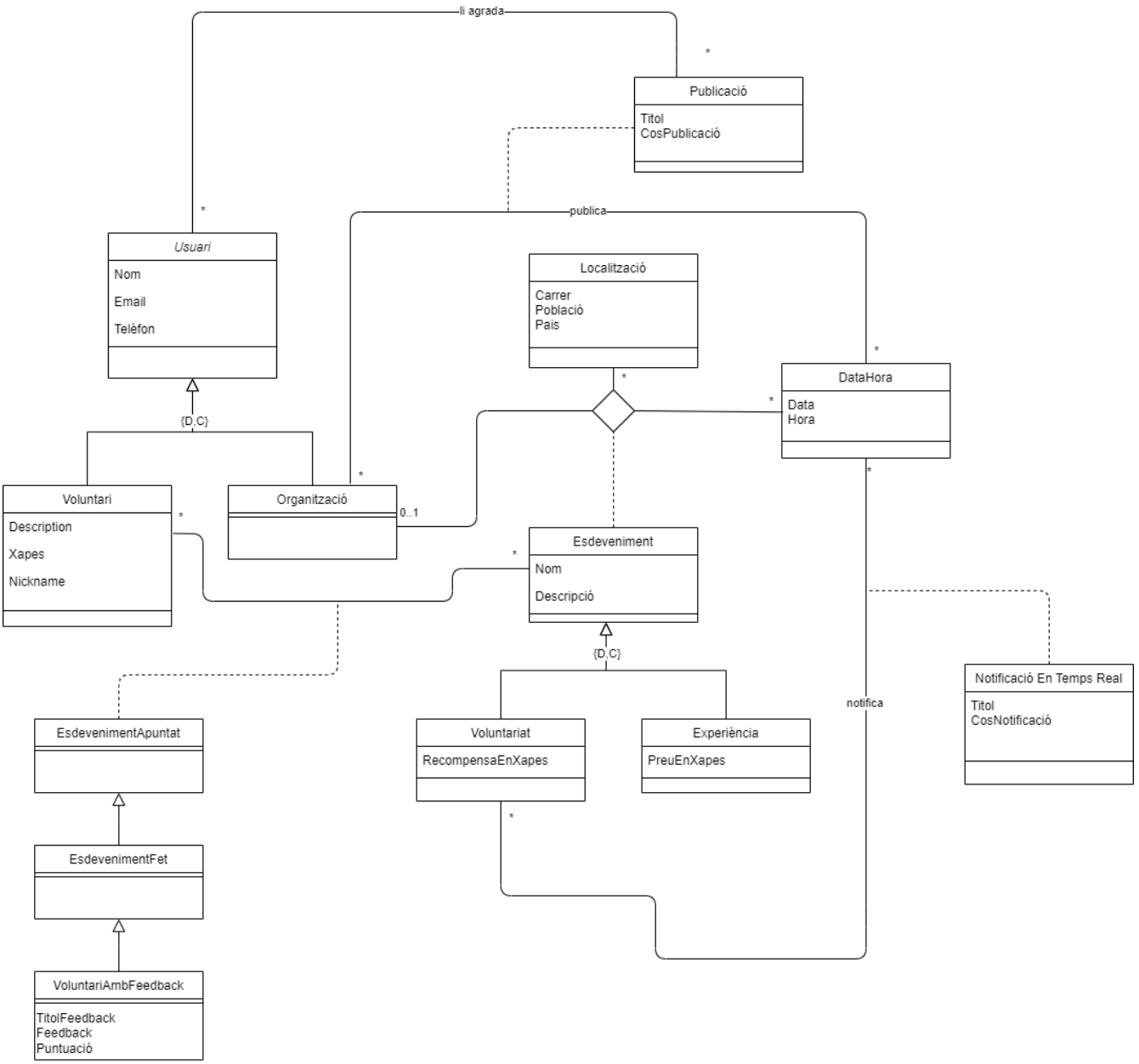


Figura 7: Diagrama d'especificació.
Font: Elaboració pròpia.

5.2.2 Claus externes

Per aclarir els conceptes del diagrama d'especificació, concretarem les claus externes del diagrama:

-**Usuari**: {email}
-**Localització**: {carrer+població+pais}
-**DataHora**: {data+hora}

5.2.3 Restriccions textuals

Per acabar, cal definir les restriccions que no s'extreuen directament del diagrama, pero que són igual d'importants per a l'especificació de requisits, les restriccions textuals.

RT1: Cap Voluntari es pot apuntar a un Esdeveniment amb una data anterior a la data actual.
RT2: Cap Voluntari pot confirmar assistència a un Esdeveniment amb una data no igual a l'actual.
RT3: Cap Organització pot fer una Notificació en temps real a un Voluntariat amb una data no igual a l'actual.
RT4: L'atribut puntuació de VoluntariAmbFeedback només pot ser un número de l'1 al 5.
RT5: El PreuEnXapes i RecompensaEnXapes no pot ser mai inferior a 0.

5.2.4 Decisions preses per al disseny del diagrama

Per a poder fer aquest diagrama d'especificació hem hagut de pensar en els requisits i prendre decisions per a que aquests s'incloguin dins d'ell. Ara procedirem a explicar-les.

La primera i la decisió més rellevant que hem hagut de prendre és la de la classe associativa EsdevenimentApuntat, ja que hem decidit que aquesta classe lògica resultant de l'asociació entre Voluntari i Esdeveniment, tindrà una subclasse que hereda d'ella, la classe EsdevenimentFet, per a saber si s'ha assistit a aquest Esdeveniment.
Ho hem fet d'aquesta forma ja que creiem que és una bona forma de tenir diferenciats els que simplement s'han apuntat i els que ho han fet, pero d'una forma més organitzada que tenint una variable.
També hem creat una altre subclasse que hereda d'EsdevenimentFet, anomenada VoluntariAmbFeedback, que representa el Feedback fet a un Esdeveniment a partir d'un EsdevenimentFet.
Aprofitant EsdevenimentFet, podem utilitzar la mateixa estratègia que abans i tenir més organitzada la informació només per a que es pugui fer Feedback si el voluntariat s'ha fet.
Ho fem d'aquesta manera ja que queda molt més clara l'especificació i, a més, així ja es pot extreure del diagrama que aquest Feedback tindrà la informació del Voluntari, de l'Esdeveniment, i de si ha assistit a aquest.

A més d'aquesta, i per acabar, les altres decisions que hem pres i que són menys significatives però no trivials són que; la NotificacioEnTempsReal ha d'estar relacionada amb el seu Voluntariat en una DataHora determinats. La Publicació ha d'estar relacionada amb qui la crea (Organització) i a qui li pot agradar (Usuari). L'Esdeveniment és el resultat d'una Organització que el crea, en una Localització específica i en una DataHora específics.

6. Arquitectura del Software

L'arquitectura del Software és la descripció dels subsistemes i components que conformen un sistema software i les relacions que existeixen entre ells.
En els següents subapartats procedirem a parlar sobre l'arquitectura de la nostra aplicació, la seva descripció concreta, i el seu disseny.

6.1 Arquitectura de l'Aplicació

En aquest primer punt, definirem l'arquitectura necessària que hem dissenyat per a poder implementar el nostre projecte. Això ho farem mitjançant un patró de disseny arquitectònic.

Un patró de disseny arquitectònic s'usa per a resoldre problemes referents a l'arquitectura del software. Aquest patró no deixa de ser una solució general a un problema concret, pel que es pot reutilitzar i és aplicable a diferents problemes de disseny de software.

El patró de disseny triat és l'arquitectura en tres capes.
Aquesta manera de treballar ens facilitarà la canviabilitat, que alhora ens facilitarà l'extensibilitat, la portabilitat, la mantenibilitat i la reestructurabilitat de l'aplicació, i també ens permetrà provar-la amb més facilitat gràcies al seu aïllament entre capes.

Per aquest motiu, l'aïllament entre capes, escollim aquest patró. Per a que un canvi en qualsevol de les capes només afecti a aquella capa.
És a dir, per exemple, que un canvi al sistema de finestres de l'aplicació (Capa de Presentació) només afecti a aquesta capa, o que un canvi en el Gestor de BBDD, només afecti a la capa de Persistència. I finalment, que la capa de Domini, que s'encarrega de la gran majoria de la lògica, sigui independent dels canvis de plataforma, Sistema Operatiu (SO), o canvis en altres capes.

Per a poder complir aquest tipus d'arquitectura hem de mantenir tots els components agrupats en capes, i assegurar-nos que la comunicació entre aquests només es faci entre components de la mateixa capa o de les capes contigües.

La podem observar i entendre molt fàcilment amb la següent figura:

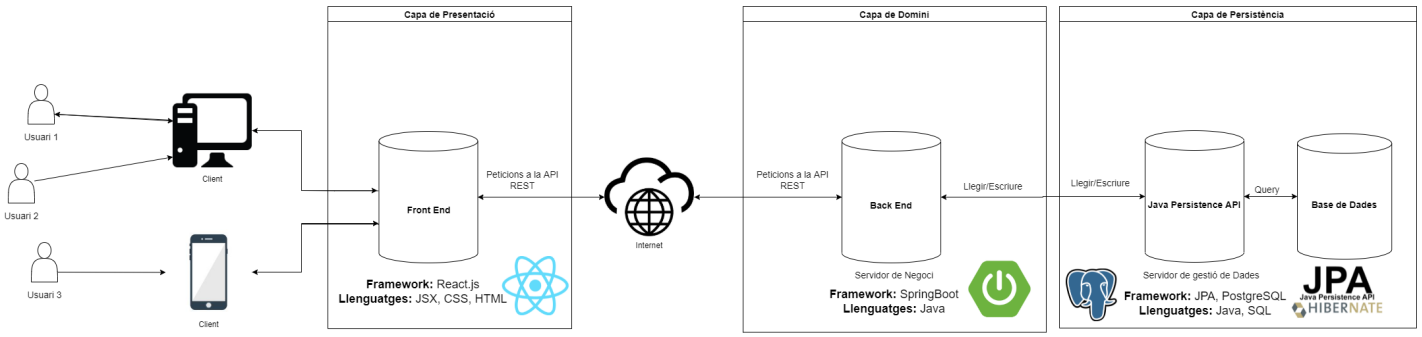


Figura 8: Esquema conceptual de les capes de l'aplicació, amb els seus frameworks.
Font: Elaboració pròpia.

Com veiem a la figura, queda molt clar que l'aplicació es divideix en tres capes. La capa de presentació, la capa de domini i la capa de persistència. Les passarem a explicar en detall al punt 6.2.

També podem observar que a cada capa s'utilitza un framework diferent, l'explicació completa de cada framework la veurem a l'apartat d'"Implementació", el punt 7.
Abans, però, passarem a explicar com seria el flux de l'arquitectura de l'aplicació.

En primer lloc, veiem que els usuaris es connecten a l'aplicació mitjançant un client, ja sigui un ordinador o qualsevol dispositiu amb accés a un navegador web.
A partir d'aquí, el que veuran serà la vista que ofereix la nostra Capa de Presentació, el Front End (creat amb el framework React.js), i interactuaran amb ella.
Si duen a terme una acció que requereix llegir o escriure certa informació, s'accedirà mitjançant una API REST a la nostra Capa de Domini, el servidor de Back End (creat amb el framework Spring Boot).
Finalment, i si aquesta capa ho creu necessari, accedirà també a la Capa de Persistència per a que la Java Persistence API (d'ara endavant JPA, i que inclou Hibernate com a generador de Persistència predeterminat) pugui llegir/escriure a la Base de Dades el que ens interessi.

Aquest recorregut, acabada l'operació que havia de realitzar, es recorre a la inversa quan la Capa de Domini rep una resposta exitosa o no de la Capa de Presentació.
Quan la Capa de Domini ja té la resposta, o en cas que no requereixi una crida a la Capa de Persistència, computarà el necessari i enviarà a la Capa de Presentació la informació necessària que aquesta última necessita.
Així l'usuari podrà rebre feedback sobre la seva acció i l'aplicació s'haurà actualitzat.

6.2 Descripció de les capes

Ara passarem a explicar en detall totes les capes anomenades anteriorment, per a entendre arquitectònicament quina és la seva finalitat i quina activitat desenvolupen.
En aquest apartat només s'explicaran els termes d'arquitectura, però no entrarem encara en parlar de frameworks, codi, interfície,etc. Això ho farem a l'apartat d'"Implementació", el 7.

6.2.1 Capa de presentació

La Capa de presentació és l'encarregada, bàsicament, de presentar o mostrar tota la informació de l'aplicació a l'Usuari, ja sigui Voluntari o Organització.

Aquesta capa coneix perfectament com ha de presentar les dades a l'usuari, i també com les ha de recollir per a enviar-les on calgui (en aquest cas només es pot comunicar amb la capa de domini).
En canvi, aquesta capa ignora quines transformacions se li han de fer a les dades per a poder donar resposta a l'usuari, i ho delega.
En aquesta capa, cada controlador està lligat a un cas d'ús i l'única validació que ha de fer es basa en la sintaxis de les dades que introdueixen els usuaris.

Per a resumir els conceptes tenim que, la capa de presentació:

- Es **relaciona** amb:
    - **Els Usuaris,** rebent els esdeveniments que aquests envien i presentant o mostrant-los els resultats que requereixen.
    - **La Capa de Domini,** comunicant-se per a passar-li els esdeveniments i accions que requereixen tractar la informació, ja que aquesta sap com fer-ho. O bé per a rebre les respostes que aquesta capa ens ofereix a l'enviar-li informació o esperar rebre-la.
- És **responsable** de:
    - **Tractar tota la interacció** amb la presentació (ja sigui per botons, menús, llistes,etc.)
    - **Delegar** els processos que no pugui fer a la capa corresponent (Domini).
    - **Comunicar els resultats** o respostes d'aquestes interaccions als usuaris.

6.2.1.1 Mapa Navegacional

Per a tenir clara l'estructura visual de la presentació, farem un diagrama que consisteix en un mapa navegacional de l'aplicació. Per a poder interpretar-lo d'una forma més concisa, hem decidit que aquest sigui en una notació adaptada del BPM.

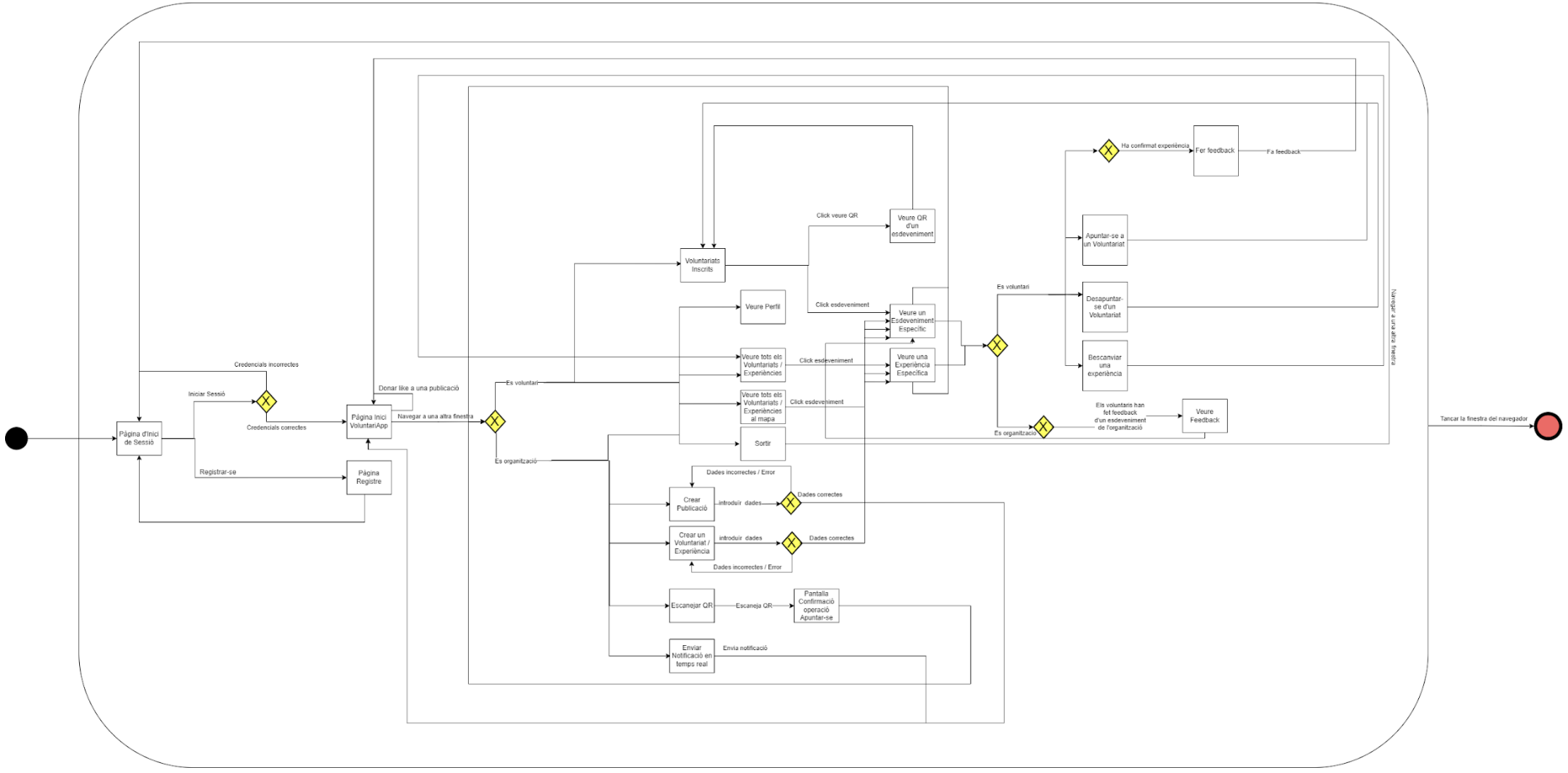


Figura 9: Mapa Navegacional de pantalles.
Font: Elaboració pròpia.

Com és un flux molt llarg i a simple vista pot no entendre's, l'explicarem part per part.

Per començar, tenim el procés d'Inici de sessió, en el que no entrarem molt en detall, ja que és el flux tradicional que se sol implementar a totes les pàgines web.

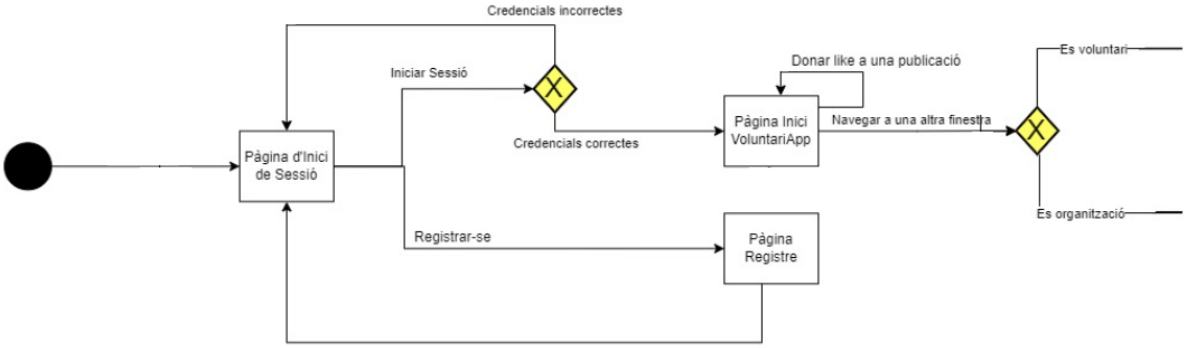


Figura 10: Mapa Navegacional inici sessió.
Font: Elaboració pròpia.

Podem veure que si tot va com s'espera, s'accedeix a la pàgina principal de VoluntariApp (Publicacions), i depenent de si és voluntari o organització, podrà fer unes funcionalitats o unes altres.
Per explicar-ho per parts, primer començarem amb el flux únic del Voluntariat, és a dir, totes aquelles funcionalitats que només fa l'Usuari Voluntari.

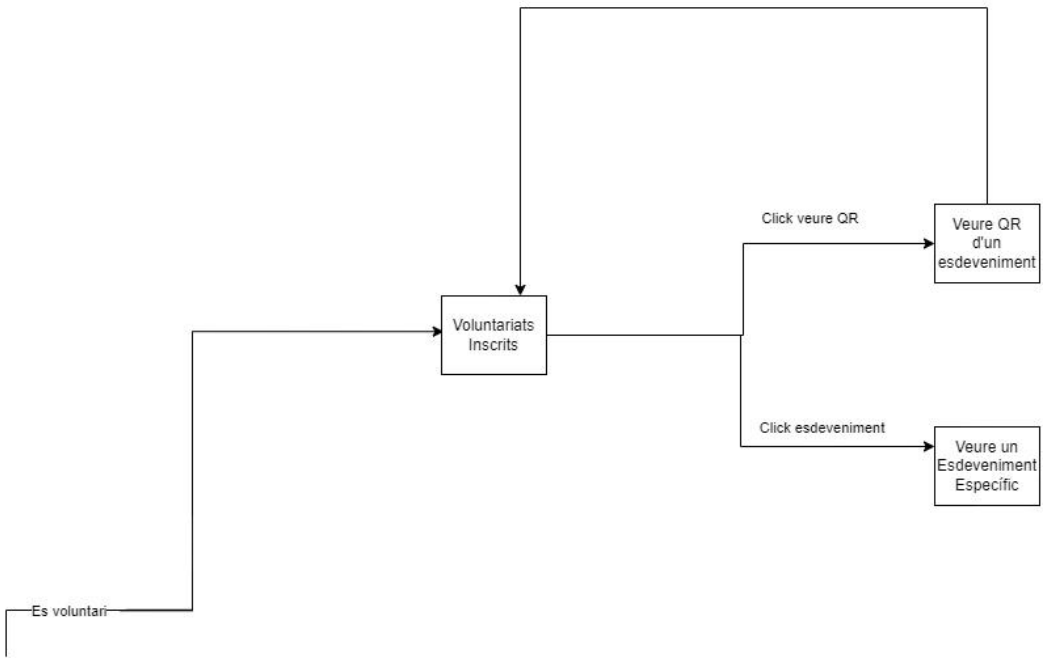


Figura 11: Mapa Navegacional exclusiu voluntaris.
Font: Elaboració pròpia.

En aquest cas, veiem que depenent de la condició, un Voluntari podrà veure els Voluntariats en els que està inscrit.
Després podrà veure el QR dels voluntariats o bé veure el detall de cada voluntariat en el que estigui inscrit. En el primer cas podrà tornar a decidir la funcionalitat que li interessi. El segon cas l'explicarem més endavant, ja que aquesta activitat, "Veure Esdeveniment Específic" també es veu en altres punts del flux que explicarem després.

En segon lloc, explicarem el flux únic de l'Organització, és a dir, totes aquelles funcionalitats que només fa l'Usuari Organització.

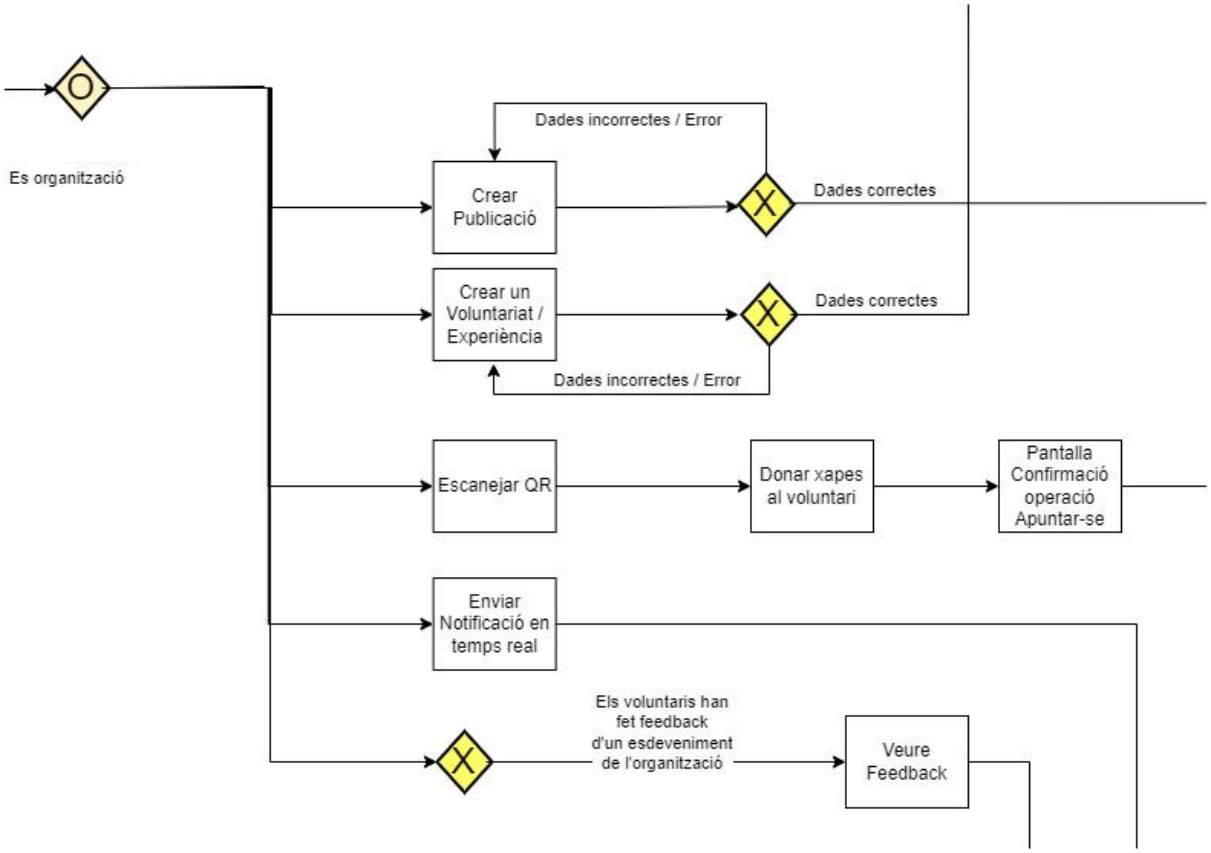


Figura 12: Mapa Navegacional exclusiu organitzacions.
Font: Elaboració pròpia.

Com podem observar, depenent de la condició l'Organització pot fer diferents activitats.
Pot Crear una Publicació, i quan ho faci satisfactòriament podrà tornar a decidir la funcionalitat que li interessi.
Pot Crear un Voluntariat o una Experiència, i quan ho faci satisfactòriament podrà veure l'Esdeveniment específic (com hem dit abans, ho explicarem més en detall més endavant).
Pot Escanejar el QR d'un Voluntari que vagi a fer el Voluntariat per confirmar assistència, i quan ho faci, el Voluntariat obtindrà les xapes corresponents, i alhora veurà un missatge confirmant que ha confirmat assistència. Després podrà tornar a decidir la funcionalitat que li interessi.
Pot Enviar una Notificació en temps real als Voluntaris que estan fent el seu voluntariat.
Si els Voluntaris han fet feedback d'un Voluntariat de l'Organització, podran veure aquest Feedback. Després podrà tornar a decidir la funcionalitat que li interessi.

Finalment passarem a explicar el flux de funcionalitats que poden fer tant els Voluntaris com les Organitzacions.

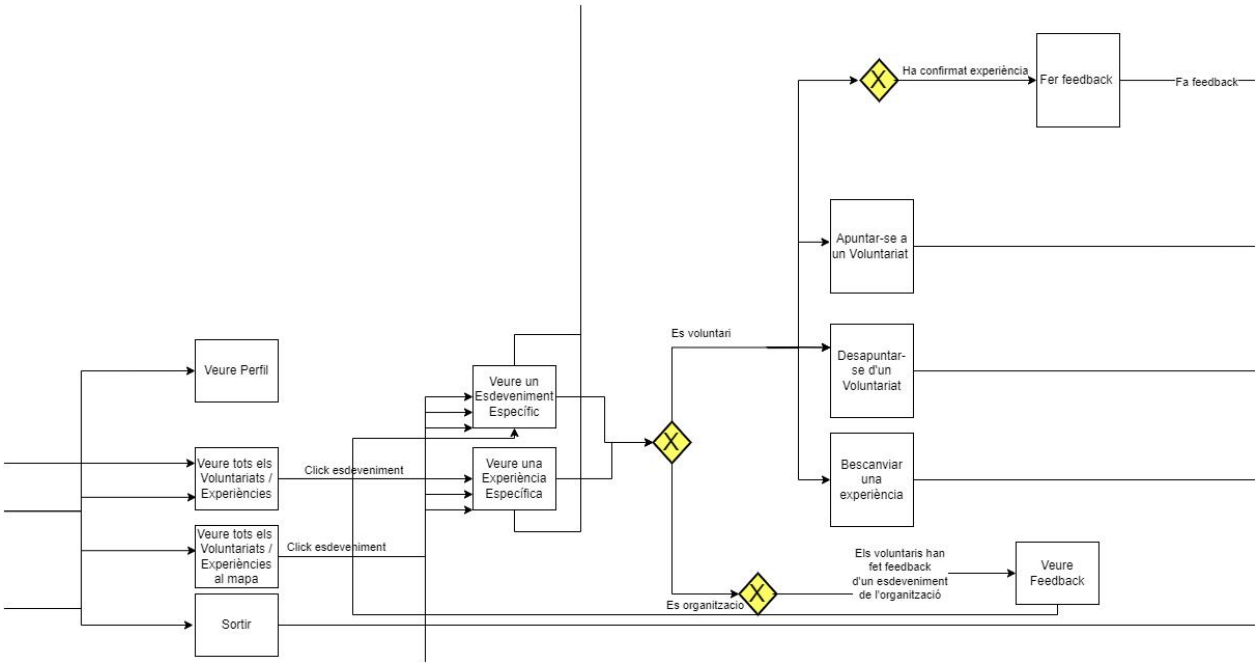


Figura 13: Mapa Navegacional exclusiu organitzacions.
Font: Elaboració pròpia.

Com veiem a la figura, tant un Voluntari com una Organització poden fer, segons la condició, el següent.
Poden veure el seu perfil, amb tota la informació.
Poden veure tots els Voluntariats i Experiències (en forma de llista o situats en un mapa) i alhora, seguint el flux, poden visualitzar un Esdeveniment específic. Si l'Usuari que està veient l'Esdeveniment específic és un Voluntari, podrà fer diverses accions més.
Aquestes accions realment estan dins de la finestra de Esdeveniment Específic, però les hem posat d'aquesta manera per a que quedin més clares, pero no son finestres de la presentació.
En el cas que aquest Esdeveniment sigui un Voluntariat, poden Apuntar-se i Desapuntar-se d'aquest Voluntariat. Després de fer l'acció, poden accedir a tots els voluntariats inscrits.
Si en canvi, aquest Esdeveniment és una Experiència, poden Bescanviar l'Experiència si tenen suficients xapes. Després podran tornar a decidir la funcionalitat que li interessi.
Finalment, tots els Usuaris podran Sortir de la sessió i tornar al primer procés d'Inici de Sessió.

Independentment del pas on estiguin, si tanquen la finestra del navegador, s'acabarà el procés.

Per si queden dubtes sobre el flux per als usuaris, inclourem un manual d'usuari després de l'apartat d'"Implementació", al punt 8.

6.2.2 Capa de domini

La Capa de domini és l'encarregada, bàsicament, de computar tot el necessari per a que la nostra aplicació funcioni tal com desitgem. És on resideix tota la lògica i la que ens fa possible que puguem satisfer els requisits més complexos.

Aquesta capa coneix perfectament com ha de satisfer les peticions de l'usuari que venen donades des de la capa de presentació, sap com treballar-les i com transformar-les en dades útils.
En canvi, aquesta capa ignora on es guarden les dades, com es guarden i, també, com es presenten o mostren a l'usuari.
En aquesta capa, cada controlador integrarà tantes classes de model com calgui per a satisfer un cas d'ús.

Per a resumir els conceptes tenim que, la capa de domini:

- Es **relaciona** amb:
  - **La Capa de Presentació,** comunicant-se per a rebre els esdeveniments o accions que requereixen tractar la informació. I també per a passar-li les respostes i resultats que el domini ha tractat i que li interessen a la presentació.
  - **La Capa de Persistència,** comunicant-se per a passar-li les operacions de consulta, modificació i escriptura i rebent les respostes i resultats que aquesta ens dona.
- És **responsable** de:
  - **Tractar tota la interacció** amb el domini, assabentar-se dels esdeveniments o accions que rep, i de les consultes que ha de fer i fa.
  - **Executar les accions** necessàries i **canviar l'estat del domini.**
  - **Delegar** els processos que no pugui fer a la capa corresponent (Presentació, Persistència).
  - **Obtenir els resultats i comunicar les respostes**.

6.2.2.1 Diagrama de disseny

Com ens trobem a l'apartat d'Arquitectura de la capa del domini, és molt interessant fer un diagrama de disseny d'aquesta capa per a conèixer com l'hem dissenyat.
En aquest apartat el farem i afegirem a continuació les seves restriccions textuals.

6.2.2.1.1 Diagrama

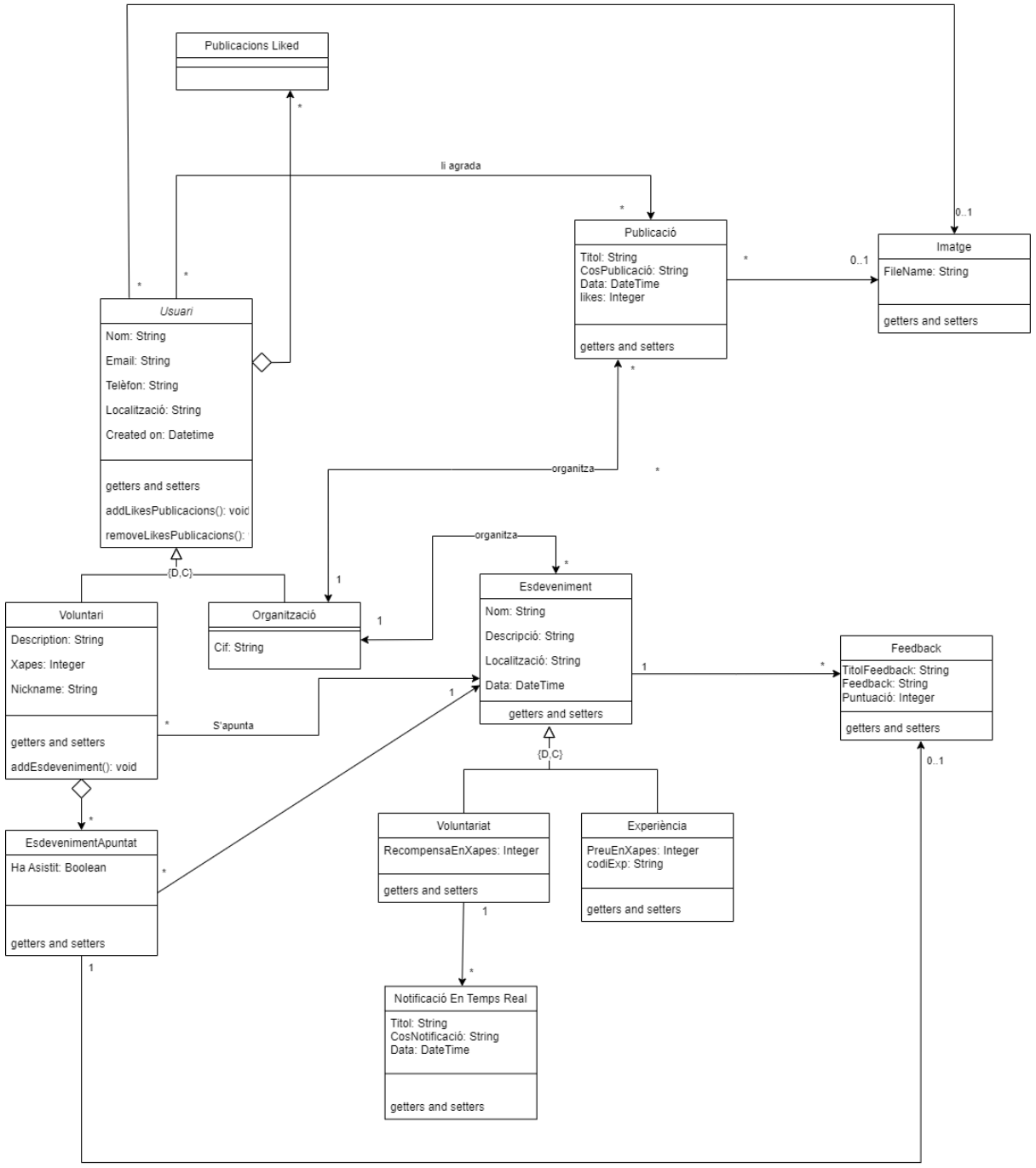


Figura 14:Diagrama de disseny.
Font: Elaboració pròpia.

6.2.2.1.2 Claus externes

-**Usuari**: {idUsuari}.
-**Publicació**: {idPublicacio}.
-**Esdeveniment**: {idEsdeveniment}.
-**NotificacióEnTempsReal**: {idNotificacio}.
-**EsdevenimentApuntat**: {idEsdeveniment, idVoluntari}.
-**Feedback**: {idFeedback}.

-**Imatge**: {idImatge}.
-**PublicacionsLiked**: {idUsuari, idPublicacio}.

6.2.2.1.3 Restriccions textuals

RT1: Cap Voluntari es pot apuntar a un Esdeveniment amb una data anterior a la data actual.
RT2: Cap Voluntari pot confirmar assistència a un Esdeveniment amb una data no igual a l'actual.
RT3: Cap Organització pot fer una Notificació en temps real a un Voluntariat amb una data no igual a l'actual.
RT4: L'atribut puntuació del Feedback només pot ser un número del 0 al 5.
RT5: El PreuEnXapes i RecompensaEnXapes no pot ser mai inferior o igual a 0.
RT6: Només pot haber un Esdeveniment alhora en una DataHora i una Localització determinades.
RT7: Un Feedback només podrà fer-se per un Voluntari si aquest té un EsdevenimentApuntat al qual ha assistit (haAsistit = true).

6.2.2.1.4 Decisions preses de l'especificació al disseny

Per a transformar el diagrama d'especificació al diagrama de disseny de classes hem hagut de prendre diverses decisions arquitectòniques. Les passarem a explicar a continuació.

Primer, hem decidit que la classe Publicació sigui una classe per si sola, relacionada amb l'Organització qui la crea. Així hi podrem navegar fàcilment.
Alhora, hem decidit afegir una classe per a guardar les publicacions a les que li ha donat "m'agrada" l'Usuari, que està estrictament relacionada amb l'Usuari.
Si ho fem d'aquesta manera, tindrem accés instantani i farem menys accessos innecessaris.

Després, hem decidit crear una classe EsdevenimentApuntat, igual que a la especificació, que ens indicarà els esdeveniments als que està apuntat un Voluntari. I hem insertat un camp de tipus booleà per a saber si l'Usuari ha confirmat assistència. D'aquesta manera ho tenim més accesible per a consultar-ho i no necessitem navegar.
Hem cregut que era la decisió més lògica i senzilla per a tenir accés en tot moment a la informació.

Una altra decisió rellevant és la traducció del Feedback. Hem decidit crear una classe nova Feedback, que ja no serà una subclasse. Aquesta classe estarà relacionada amb un Esdeveniment, ja que cada feedback es fa per un esdeveniment, i amb un EsdevenimentApuntat, ja que cada feedback es fa per un Voluntari en un esdeveniment en el que està apuntat i al qual ha assistit.
Hem triat aquest disseny per a poder obtenir els feedbacks de cada Esdeveniment de manera senzilla, i alhora poder obtenir els diferents feedbacks de cada Voluntari apuntat a un Esdeveniment.

Finalment, hem decidit crear la classe NotificacióEnTempsReal que ha passat a estar relacionada únicament amb la classe Voluntariat, ja que d'aquesta manera hi ha una navegabilitat lògica i clara. Ens vam plantejar que aquesta classe podriem traduir-la amb informació derivada materialitzada, pero al final hem cregut més oportú fer-ho sense, i calcular aquesta informació posteriorment navegant.

En lo referent a les claus externes, hem decidit fer surrogate keys per a que sigui més senzill i pràctic accedir a cada classe. No hem afegit al diagrama els id ja que només l'engrandiria i dificultaria la seva llegibilitat.

6.2.3 Capa de persistència

La Capa de persistència és l'encarregada, bàsicament, de fer les operacions corresponents amb la BBDD.

Aquesta capa coneix perfectament on i com estan emmagatzemades les dades, i com actualitzar la BBDD.
En canvi, aquesta capa ignora com es tracten les dades un cop estan disponibles.

Per a resumir els conceptes tenim que, la capa de persistència:

- Es **relaciona** amb:
    - **La Capa de Domini,** comunicant-se per a rebre les operacions de consulta, modificació i escriptura que s'hauran de fer i passant-li les respostes i resultats que es donen de les diferents operacions a BBDD.
    - **El Sistema de Gestió de Bases de Dades (SGBD),** comunicant-se per a passar-li les operacions a BBDD amb el llenguatge i formats adients. I rebent les respostes i resultats que s'hauran de passar a la capa de domini.
- És **responsable** de:
    - **Deixar que el domini no conegui on són les dades.**
    - **Delegar** els processos que no pugui fer a la capa corresponent (Domini).
    - **Obtenir els resultats i comunicar les respostes**.

6.2.3.1 Diagrama Entitat Relació (ERD)

Per aclarir com està formada la capa de persistència d'una manera més visual, hem generat un ERD en una eina de gestió i administració de bases de dades (pgAdmin).
El diagrama ens aporta informació sobre les taules a la base de dades, i les relacions entre aquestes entitats.

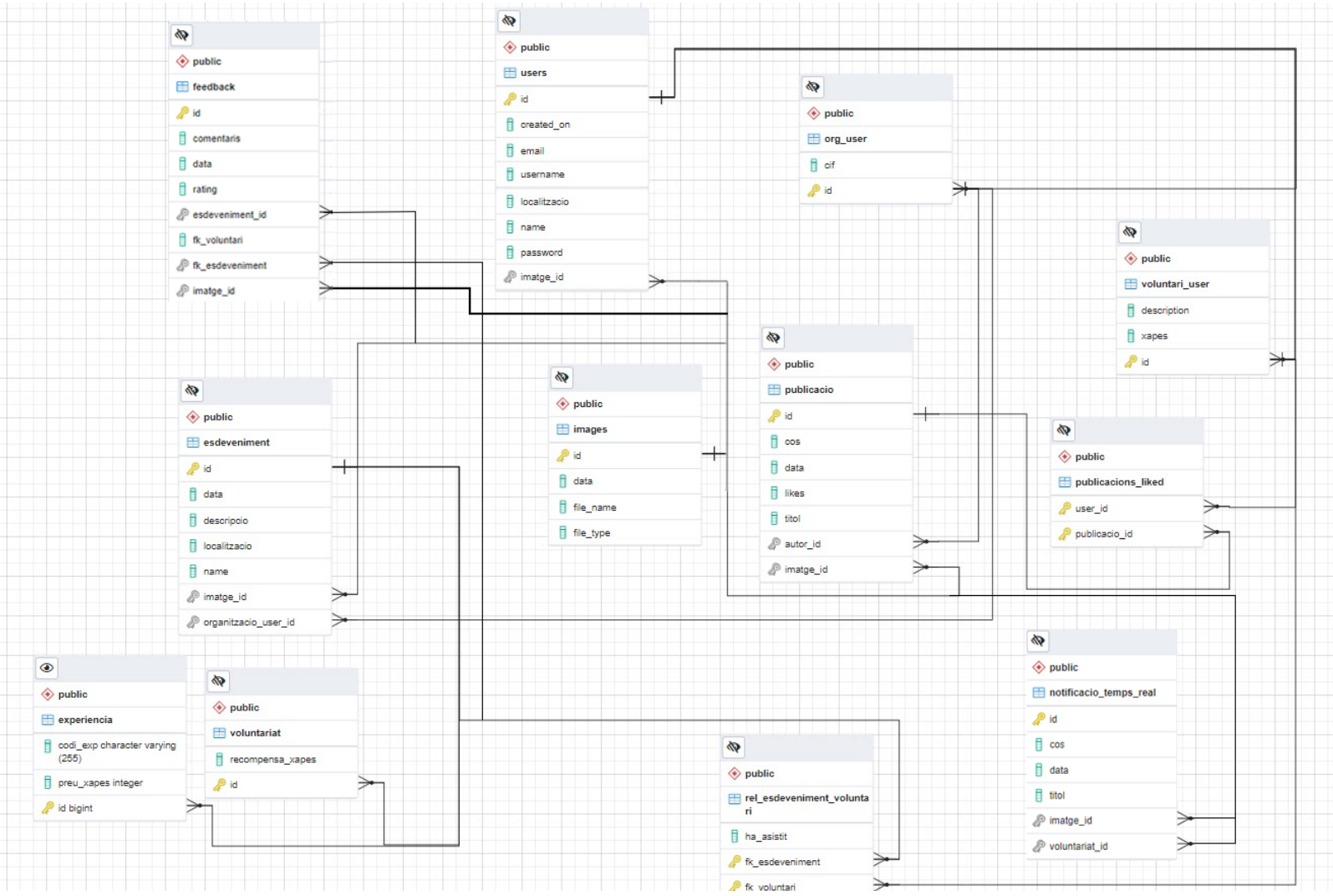


Figura 15: Diagrama ERD.
Font: Elaboració pròpia amb pgAdmin 4.

7. Implementació

7.1 Aprenentatge dels frameworks

Abans d'entrar en detall amb tot lo relacionat amb la implementació de les capes, parlarem breument sobre el que ha suposat una de les dificultats més grans del projecte, l'aprenentatge dels frameworks.

En aquest apartat no volem parlar sobre perquè hem triat cada framework, ni els explicarem en detall, ja que ho farem als punts següents. En aquest apartat volem parlar sobre la dificultat afegida d'aprendre tecnologies noves per a crear un producte des de zero.

Primer, i parlant del framework del Front-End, *React*, cal destacar que no és un framework al qual estiguéssim acostumats.
Si bé és veritat que aquest framework funciona amb les bases de qualsevol Front-End actual (com son un HTML + CSS + JS), també inclou conceptes i formes de treballar a les quals t'hi has d'acostumar amb el temps, com el nou llenguatge JSX (JS+XHML).
Malgrat això, el que podem concluir és que, encara que sigui un framework que al principi sembla que té una dificultat elevada, la corba de dificultat s'acaba aplanant cap al segon *Sprint*, i d'aquí en endavant, ens podem acostumar i treure resultats bons de forma ràpida.

En segon lloc, i ja parlant del framework del Back-End, *Spring Boot*, tampoc és un framework que haguéssim utilitzat tot i ser molt conegut i usat pels desenvolupadors de Java.
La part positiva d'aquest framework és exactament aquesta, que usa un llenguatge de programació Orientat a Objectes al qual estem acostumats, pero tota la seva infrastructura és completament nova.
Aquest framework és molt complet, té moltissimes possibilitats, i funciona amb anotacions. Aquestes han estat les contres més importants d'aprendre aquest framework, ja que abans de res havíem d'estudiar totes aquestes anotacions, i adaptar el pseudocodi que ens imaginàvem a aquest tipus d'implementació, i sent inexperts això és una feina pesada.
Tot i així, al final hem aconseguit dominar l'ecosistema i aprendre totes aquelles anotacions i funcionalitats que hi podem desenvolupar, creant així el nostre Back-End, però el domini del framework s'aconsegueix cap a l'*Sprint* 3.

Finalment, el framework utilitzat per a la capa de persistència, *SpringJPA*, també era completament nou per a nosaltres, ja que estàvem acostumats a generar la persistència de forma íntegra, i per estalviar-nos feina vam decidir utilitzar-lo.
Aquest framework té el mateix problema que l'anterior, ja que ambdos són part d'*Spring*, però alhora, quan acabes agafant pràctica i estudiant totes les anotacions, al final el desenvolupament s'agilitza, i el que al principi ens podia costar 3 hores, quan es domina es pot fer en 1, el que és molt útil a llarg plaç.

Un cop esmentat tot això, passarem a explicar la implementació de cada capa i del seu framework en detall.

7.2 Front-End

Per a l'explicació de la implementació de la capa de presentació, explicarem el Front-End. El framework escollit per a aquesta capa és *React* [8].
React és un framework basat en una extensió del llenguatge de programació *JavaScript* i *XML, JSX* [9], i en el que s'utilitzen també, JS natiu, HTML i CSS.

Funciona com una vista que renderitza components, per això és una de les opcions més adients per al nostre projecte, ja que per a la capa de presentació només necessitem mostrar informació que canvia durant el temps, fer la interacció amb l'usuari i rellevar gran part de la lògica (la de negoci) al Back-End.
Amb aquest framework aquesta tasca es se'ns facilita molt, ja que *React* incorpora moltes llibreries, nodes i frameworks que ens poden ajudar a l'hora del desenvolupament, traient-nos hores de feina.

7.2.1 Codi i funcionament

Per a entendre l'estructura i funcionament del codi, passarem a posar uns exemples i a explicar els trets més rellevants que fan que el nostre Front-End funcioni com ho fa.
Com hem esmentat anteriorment, hem utilitzat JSX per a programar aquesta capa.

Abans però, hem d'indicar que a PAE vam decidir que utilitzariem aquest framework per a la capa de presentació, i vam desenvolupar diferents arxius de codi amb un disseny simplificat i *hardcoded*, és a dir, amb dades falses i estàtiques. Aquests arxius eren l'Inici, els Voluntariats Generals, el Perfil i les Experiències Generals.
Per aquest motiu, hem volgut respectar la decisió i treballar sobre aquest frameworks i aprofitar la feina feta a PAE, mentre que hem hagut de crear molts arxius nous i reestructurar bastant els antics.

A la Figura 16 podem veure els directoris del nostre Front-End, que comentarem en aquest apartat.
El que hem de saber és que tot el directori "src" conté el codi. Els components encarregats de renderitzar les vistes estan situats a l'apartat de "components" i a la carpeta "utils" s'inclouen fitxers de codi que necessitem per a configurar APIs o llibreries externes.
Finalment, els arxius que estan situats a "src" sense estar dins de cap directori són les parts (components) fonamentals de la nostra presentació.

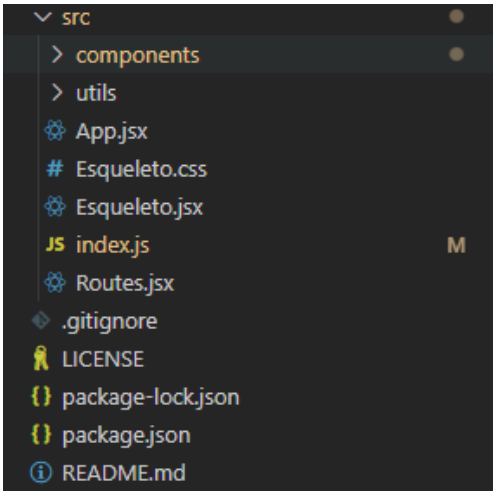


Figura 16: Directoris Front-End.
Font: Elaboració pròpia amb IntellIJ.

El primer que fa la nostra aplicació és executar l'arxiu *index.js*, que podem veure a la Figura 17.
Aquest arxiu el que fa és indicar quin *Document Object Model (DOM)* [10] executa. Un DOM és una interfície de programació per als documents HTML i XML, i que facilita una representació estructurada del document.

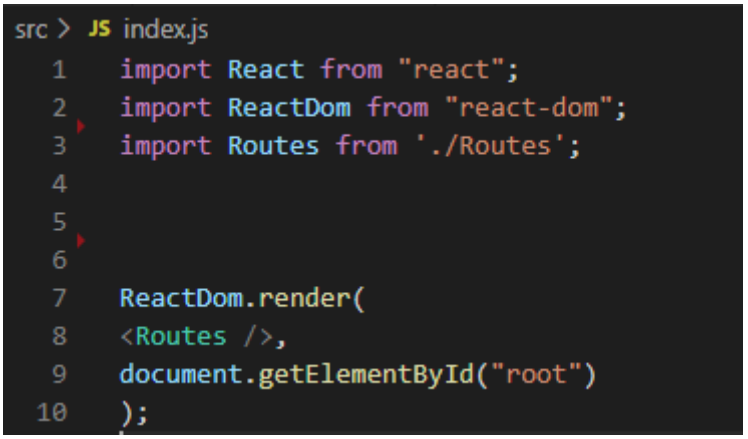


Figura 17: index.js.
Font: Elaboració pròpia amb IntellIJ.

Com podem veure, el que fa aquest arxiu és renderitzar l'arxiu "Routes.jsx", que passarem a explicar a continuació.

L'arxiu "Routes.jsx" (Figura 18) s'encarrega de crear totes les rutes de les vistes depenent de la direcció URL (el *path*).
Tal com està organitzada aquesta capa, només accedint a l'enllaç que ens interessi, el component ens enrutarà directament a la vista que ens interessa (en cas que ja tinguem la sessió iniciada en aquell navegador).
Hi ha dos nivells d'abstracció en aquesta capa, el primer nivell està comprès pel "Login", el "Registre" i l'"Esquelet".
En els dos primers, només es crida al component indicat, pero en el tercer, apareix el segon nivell d'abstracció.

src > Routes.jsx > ...

```jsx
import React from "react";
import {BrowserRouter, Switch, Route} from 'react-router-dom';
import Login from './components/Login';
import Esqueleto from './Esqueleto';
import Registre from "./components/Registre";

function Routes() {
  return (
    <BrowserRouter>
      <Switch>
        <Route exact path="/" component={Login}/>
        <Route exact path="/home" component={Esqueleto}/>
        <Route exact path="/general" component={Esqueleto}/>
        <Route exact path="/enrolled" component={Esqueleto}/>
        <Route exact path="/profile" component={Esqueleto}/>
        <Route exact path="/experiencies" component={Esqueleto}/>
        <Route exact path="/creaVoluntariat" component={Esqueleto}/>
        <Route exact path="/QrCode" component={Esqueleto}/>
        <Route exact path="/voluntariat/:id" component={Esqueleto}/>
        <Route exact path="/experiencia/:id" component={Esqueleto} />
        <Route exact path="/voluntariatsMapa" component={Esqueleto}/>
        <Route exact path="/experienciesMapa" component={Esqueleto}/>
        <Route exact path="/confirmaAsistVol/:idVoluntari/:idVoluntariat" component={Esqueleto}/>
        <Route exact path="/LlegeixQR" component={Esqueleto}/>
        <Route exact path="/creaPublicacio" component={Esqueleto}/>
        <Route exact path="/notificacio/:idVoluntariat" component={Esqueleto} />
        <Route exact path="/registre" component={Registre} />
        <Route path="/feedbacks" component={Esqueleto} />
        <Route path="/bescanviades" component={Esqueleto}/>
        <Route path="/myVoluntariats" component={Esqueleto} />
        <Route path="/myExperiencies" component={Esqueleto} />
      </Switch>
    </BrowserRouter>
  );
  }

  export default Routes;
```

Figura 18: Routes.jsx.
Font: Elaboració pròpia amb IntellIJ.

Al segon nivell, el que inclou l'"Esquelet", s'accedeix només quan l'usuari té la sessió iniciada.
Aquest també inclou un arxiu de rutes anomenat "Navigation.jsx", que és invocat quan a "Routes.jsx" es direcciona amb el component de l'Esquelet, ja que en aquest hi ha un fragment de codi que direcciona tots els enllaços, que es troben a la barra lateral de l'aplicació, amb el component de Navigation, tal com podem veure a la Figura 19.

```jsx
{/*<!-- Begin Page Content -->*/}
<div className="container-fluid">

    {/*<!-- Page Heading -->*/}
    <h1 className="h3 mb-4 text-gray-800">
        <Navigation />
    </h1>

</div>
```

Figura 19: Crida a Navigation desde Esqueleto.jsx.
Font: Elaboració pròpia amb IntellIJ.

I, tal com veiem a la Figura 20, el codi de "Navigation.jsx" redirecciona a cada component mitjançant les direccions URL.

```
return (
    <Fragment>
        <Route path="/home" exact component={Home} />
        <Route path="/general" >
            <VoluntariadosList voluntariados={getVoluntariados} apuntarseVoluntariado={apuntarseVoluntariado}/>
        </Route>
        <Route path="/enrolled">
            <Enrolled voluntariados={getVoluntariados} desapuntarse={apuntarseVoluntariado}/>
        </Route>
        <Route path="/bescanviades" component={Bescanviades}/>
        <Route path="/profile" component={UserProfile} />
        <Route path="/experiencies" component={Experiencies} />
        <Route path="/creaVoluntariat" component={CrearVoluntariat} />
        <Route path="/creaExperiencia" component={CrearExperiencia} />
        <Route path="/creaPublicacio" component={CrearPublicacio} />
        <Route path="/ranking" component={Ranking} />
        <Route path="/QrCode" component={QrCode} />
        <Route exact path="/voluntariat/:id" component={VoluntariatEspecific} />
        <Route exact path="/experiencia/:id" component={ExperienciaEspecifica} />
        <Route path="/voluntariatsMapa" component={VoluntariatsMaps} />
        <Route path="/experienciesMapa" component={ExperienciesMaps} />
        <Route exact path="/confirmaAsistVol/:idVoluntari/:idVoluntariat" component={ConfirmarAsistencia} />
        <Route exact path="/LlegeixQR" component={LlegeixQR}/>
        <Route exact path="/notificacio/:idVoluntariat" component={EnviarNotificacio} />
        <Route path="/feedbacks" component={FeedbacksVoluntari} />
        <Route path="/myVoluntariats" component={MyVoluntariats} />
        <Route path="/myExperiencies" component={MyExperiencies} />


    </Fragment>
);
```

Figura 20: Crida a Navigation desde Esqueleto.jsx.
Font: Elaboració pròpia amb IntellIJ.

A continuació, passarem a explicar peculiaritats de *React* que fan que aquest sigui una eina molt potent per al desenvolupament.
Aquest mètode es diu “useLayoutEffect()”. Amb aquesta declaració, com podem observar a la Figura 21, podem controlar tot el que succeeix a l’inici de la renderització.
En el cas de l’exemple, podem observar que al carregar, comprova si la cookie username existeix, en cas contrari t’envia a la pàgina d’Inici de sessió.
A més, si incloem dins dels símbols “[]” una variable, aquesta funció s’executarà cada cop que aquesta canviï.
Hi ha una funció molt similar anomenada “useEffect()” que té la mateixa funció pero s’executa quan tot l’ecosistema està estable, i la primera s’executa només renderitzar el DOM.

```
useLayoutEffect(() => {

    if(!cookies.get('username')){
        window.location.href="./";
    }

}, []);
```

Figura 21: useLayoutEffect() a Esqueleto.jsx.
Font: Elaboració pròpia amb IntellIJ.

Finalment, i per fer-nos una idea com es complementen el js i el XML/HTML en la nostra vista, a la Figura 22 podem observar com en el component “CrearVoluntariat.jsx” es retorna

un HTML (on hi ha elements com , <hr>) que inclou parts XML de mòduls i llibreries instal·lades (com el Form), i que inclou js per a controlar la lògica de interacció amb l'usuari (com els onSubmit o onChange).

```
return(
    <div>
        Crear Voluntariat
        { /*<!-- Divider -->*/}
        <hr className="sidebar-divider d-none d-md-block"/>
        <div className="Formulari">
        <Form onSubmit={handleSubmit}>

            {/*<input name = "email"  onChange={(e) => setEmail(e.target.value)}></input>*/}
            <Form.Group size="lg" controlId="nom" >
                <Form.Label style={sizeTitol}>Nom voluntariat:  <span>   </span></Form.Label>
                <Form.Control
                className ="formControl"
                    type="text"
                    value={name}
                    onChange={(e) => setName(e.target.value)}
                />
            </Form.Group>
```

Figura 22: Fragment de CrearVoluntariat.jsx.
Font: Elaboració pròpia amb IntellIJ.

7.2.2 Llibreries i APIs utilitzades

A continuació, definirem les llibreries i APIs utilitzades per la nostra capa de presentació.
Com utilitzem *React*, utilitzem moltes llibreries pel simple fet de què el propi framework ens incita a fer-ho. Passarem a descriure-les i a explicar-les, i després parlarem sobre les API que fem servir.

7.2.2.1 Llibreries

Com hem comentat abans, el propi framework ens empeny a utilitzar llibreries per la facilitat d'adaptació d'aquestes amb el propi codi.
Per a utilitzar una llibreria, l'únic que hem de fer és instalar-lo mitjançant *npm[X]* amb un "*npm install* [NomLlibreria]".
A partir d'aquí podrem utilitzar-les sense problema.
Aquestes llibreries es troben amb les seves versions al "package.json" (a la Figura 23 en podem veure un fragment).

```
"axios": "^0.21.4",
"cloudinary-core": "^2.12.0",
"cloudinary-react": "^1.7.0",
"cors": "^2.8.5",
"firebase": "^9.6.1",
"http-proxy-middleware": "^2.0.1",
"jQuery": "^1.7.4",
"leaflet": "^1.7.1",
"md5": "^2.3.0",
"qrcode": "^1.5.0",
"react": "^17.0.2",
"react-bootstrap": "^1.6.4",
"react-dom": "^17.0.2",
"react-geocode": "^0.2.3",
"react-google-maps": "^9.4.5",
```

Figura 23: Fragment de package.json.
Font: Elaboració pròpia amb IntellIJ.

Com n'hem utilitzat moltes, les descriurem breument.

7.2.2.1.1 Fortawesome

Aquesta llibreria l'utilitzem per a incloure icones a la nostra aplicació. Veiem un exemple a la Figura 24, què és un fragment de la barra lateral de l'Esquelet on s'inclou una icona de mans.

```
<li className="nav-item" >
    <Link className="nav-link collapsed" to="/general" data-toggle="col
        aria-expanded="true" aria-controls="collapseTwo">
        <FontAwesomeIcon icon={faHands} />     
        <span>Voluntariats</span>
    </Link>
</li>
```

Figura 24: FortAwesome a Esqueleto.jsx.
Font: Elaboració pròpia amb IntellIJ.

7.2.2.1.2 Axios

Axios és una llibreria basada en promeses que ens ajuda a fer les crides HTTP, la hem utilitzat per a fer totes les crides requerides a les API que utilitzem (7.2.2.2). Tenim la creació i instanciació d'una petició a la figura següent:

```
export default axios.create({
  baseURL: "http://localhost:8080/"/*,
  headers: {
    'Access-Control-Allow-Origin' : '*',
    'Access-Control-Allow-Methods':'GET,PUT,POST,DELETE,PATCH,OPTIONS',
  }*/
});
```

Figura 25: Fragment de axios a API.js.
Font: Elaboració pròpia amb IntellIJ.

7.2.2.1.3 MD5

Hem utilitzat una llibreria d'encriptació md5 per a poder transportar la contrasenya dels nostres usuaris de manera segura.
Un cop que l'usuari es registra, aquesta contrasenya s'encripta i la informació guardada a base de dades resulta en la dada segura. Ho podem veure a la Figura 26.

```
password: md5(password),
```

Figura 26: Fragment de md5 a "Registre.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.1.4 QRCode

Per a la funcionalitat que comporta la confirmació d'assistència, necessitem generar i llegir codis QR. Això ho farem amb les llibreries QRCode de *React (react-qr-reader i qrcode)*. Com podem observar a la següent figura, en aquest exemple estem utilitzant la llibreria per a llegir els QR.

```
return (
    <div>
        Escaneja el QR <br/>
    {/*<!-- Voluntariados listados -->*/}
    { /*<!-- Divider -->*/}
    <hr className="sidebar-divider d-none d-md-block"/>
        <div style={{  justifyContent: 'center', display : 'flex' }}>

            <QrReader
            delay={300}
            onError={handleError}
            onScan={handleScan}
            style={{ width: '70vh', height: 'auto', justifyContent: 'center', display : 'flex' }}
            />
        </div>
    </div>
);
```

Figura 27: Fragment QR Reader a "LlegeixQR.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.1.5 Sweet Modal / Alert

En aquest punt hem ajuntat diverses llibreries que s'utilitzen per funcionalitats similars.
La llibreria de Sweet Alert ens ofereix alertes personalitzades amb un estil molt més visual que les estàndards, donant-li una estètica més agradable a l'usuari.
La llibreria Sweet Modal, fa quelcom similar, pero amb els modals, i només l'hem utilitzat per a mostrar el codi QR.
A continuació podem veure un exemple d'alerta d'error en el cas que ja s'hagi valorat amb anterioritat un voluntariat i s'intenti tornar a valorar.

```
MySwal.fire("Oops!", 'Hi ha hagut un error amb el servidor o'+
'ja has valorat el voluntariat amb anterioritat', "error");
```

Figura 28: Fragment Sweet Alert d'error a "EnrolledItem.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.1.6 Universal Cookie

Finalment, per a poder guardar la informació necessària en la sessió de l'usuari connectat i poder conèixer-la, hem d'utilitzar una llibreria de *cookies*, com és Universal Cookies.
Aquesta llibreria l'utilitzem per saber quin usuari està connectat i amb la sessió iniciada, per a enviar notificacions, conèixer estats…
A la Figura 29 veurem un exemple en el que volem obtenir informació d'una cookie guardada per enviar-la a una api.

```
formData.append('token', cookies.get("token"));
await API.put("/confirmarAsistenciaAVoluntariat",formData)
```

Figura 29: Fragment cookies a "ConfirmarAsistencia.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.2 APIs

En aquest subapartat ens centrarem en explicar en detall quines han estat les crides i usos a les diferents API que hem fet servir.
Moltes d'aquestes API necessiten també una llibreria que haurem d'instalar mitjançant npm al package.json, pero no les hem anomenat a l'apartat anterior per a no duplicar informació.

7.2.2.2.1 API VoluntariApp

Com era d'esperar, i per què la necessitem per al funcionament complet de la nostra aplicació web, hem de cridar a la API de VoluntariApp creada per nosaltres mateixos i explicada en detall al punt 7.3.1 del document.
Aquesta és la API a la que més invoquem amb molta diferència, i ho fem aprofitant la llibreria anteriorment esmentada "Axios".

A la Figura 30 podrem veure un exemple d'una crida des del *React* a la API.
Es tracta de la crida a l'*endpoint* de "/creaExperiencia", on fem un *POST* amb un body amb la informació introduïda per l'usuari per a crear una Experiència, i després tractem la informació.

```
let iniciarSesion=async()=>{

  //window.location.href="./home";
  const body = { data: data+"T"+hora+":00.000Z", descripcio: descripcio, localitzacio: latt+"/"+lngg,
  name: name, organitzacioUser: {id: cookies.get('id') }, preuEnXapes: xapes, codiExp: codi/*, imatge: imageData*/};
  /*atLongFromAdress("Palau Sant Jordi");*/

  console.log("BODYYYY "+body);
  await API.post('/creaExperiencia', body)
  .then(response=>{
      return response.data;
  }).then(response=>{
      //setIdExperienciaCreat(response.id);

      console.log(response);
      console.log(response.id);
      setIdV(response.id);
      console.log("THEN id: "+idExperienciaCreat);

  })
  .catch(error=>{
      console.log("ERROR "+error);
      swal("Oops!","Ho sentim, alguna cosa ha sortit malament","error");

  }).finally(()=> {
    console.log("FINALLY id: "+idExperienciaCreat);
    updatePhoto();
  })

}
```

Figura 30: Fragment crida API VoluntariApp a "CrearExperiencia.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.2.2 Google Maps / GeoCoder

Per aquest subapartat hem ajuntat dos serveis de la API de Google que tenen que veure amb la localització i els mapes.

El primer de tots, el de Google Maps, l'hem fet servir per a poder visualitzar mapes als nostres components. Sobretot als Voluntariats i a les Experiències.

Tant és així que cada vista específica de qualsevol voluntariat o experiència renderitza un mapa. I també existeix una funcionalitat que s'encarrega de poder buscar Voluntariats o Experiències a través d'un mapa.
A continuació, tindrem un exemple de codi d'utilització en el que es crea un component anomenat MyMapComponent que renderitzarà un mapa.
En aquest exemple podem veure que GoogleMaps és una llibreria de react, i que cridem a la api amb la nostra key específica.

```
const MyMapComponent = compose(
    withProps({
      googleMapURL: "https://maps.googleapis.com/maps/api/js?key=AIzaSyBoAyRIT8JRaKLHLKx4jQqijJyVEQWld0g&v=3.exp&libraries=geometry,drawing,places",
      loadingElement: <div style={{ height: `100%` }} />,
      containerElement: <div style={{ height: `400px`,width: `75vw`}} />,
      mapElement: <div style={{ height: `100%` }} />,
    }),
    withScriptjs,
    withGoogleMap
)((props) =>
<GoogleMap
    defaultZoom={16}
    defaultCenter={{ lat:latt, lng: lngg }}

>
    {props.isMarkerShown && <Marker position={{ lat: latt, lng: lngg }} onClick={props.onMarkerClick} />}
</GoogleMap>
)
```

Figura 31: Métode MyMapComponent a "ExperienciaEspecifica.jsx"
Font: Elaboració pròpia amb IntellIJ.

Per altra banda, tenim el servei de Geocode. Aquest servei l'utilitzem per a poder tenir una adreça a través d'unes Longitud i Latitud determinades.
Això ens serveix ja que nosaltres emmagatzemem a base de dades aquesta Longitud / Latitud en ves d'una direcció de correu, per a poder-les representar als mapes mencionats anteriorment.
També la fem servir per a canviar una adreça a longitud i latitud, per a poder recollir les dades de l'usuari i enviar-les al backend transformades.

```
Geocode.setApiKey("AIzaSyBoAyRIT8JRaKLHLKx4jQqijJyVEQWld0g");

// set response language. Defaults to english.
Geocode.setLanguage("es");

// set response region. Its optional.
// A Geocoding request with region=es (Spain) will return the Spanish city.
Geocode.setRegion("es");

Geocode.setLocationType("ROOFTOP");

// Get address from latitude & longitude.
function adressFromLatLong (lat, long) {
    Geocode.fromLatLng(lat, long).then(
    (response) => {
      const add = response.results[0].formatted_address;
      setAddress(add);
      console.log(add);
    },
    (error) => {
      console.error(error);
    }
    );
  }
```

Figura 32: Fragment Geocode a "ExperienciaEspecifica.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.2.3 Cloudinary

La següent API a la que accedim és la de “Cloudinary”. Aquesta API l’utilitzem per a guardar imatges online i poder accedir-hi.
Després d’analitzar el cost d’emmagatzemar en base de dades tota la informació de les imatges pujades, hem cregut adient utilitzar altres formes per a guardar-les.
Cridant a la API de Cloudinary podem guardar imatges, generar un enllaç públic, i emmagatzemar-lo a la base de dades sense haver de guardar tota la informació, únicament l’enllaç.

A continuació podem veure la crida per a pujar una foto a la API i obtenir la url. Aquest mètode el cridem quan creem voluntariats, experiències i publicacions.

```
async function updatePhoto(){

  const data = new FormData()
  data.append("file", image)
  data.append("upload_preset", "voluntariapp")
  data.append("cloud_name","VoluntariApp")

  await API.post('https://api.cloudinary.com/v1_1/dxmmi55ok/image/upload',data)
  .then(response =>{
    console.log("Response data: "+JSON.stringify(response.data));
    return response.data;

  }).then(data =>{
      console.log(data);
      console.log(data.url);
      updateBBDD(data);
  })
}
```

Figura 33: Mètode updatePhoto a Cloudinary a “CrearVoluntariat.jsx”
Font: Elaboració pròpia amb IntellIJ.

7.2.2.2.4 Firebase

Per a desenvolupar la funcionalitat d’enviament de Notificacions en Temps Real per part de les organitzacions a tots els voluntaris que participin en el voluntariat, necessitavem incloure una API d’enviament de notificacions.

Firebase ens resol aquest problema amb la seva api de “Cloud Messaging”. Aquesta es pot utilitzar per enviar notificacions a tots aquells que generin un token de firebase.

A continuació veurem un codi amb la configuració necessària per a implementar aquesta API.
Primer, a la Figura 34, veiem l’arxiu firebase.js, que configura totes les claus per a accedir al servei.

Figura 34: firebase.js
Font: Elaboració pròpia amb IntellIJ.

Després, a la Figura 35 podrem veure que hem de crear un serviceWorker, per a que l'usuari pugui rebre les notificacions sempre que tingui el dispositiu amb un token registrat encès, encara que estigui el navegador en segon pla.

```
importScripts('https://www.gstatic.com/firebasejs/9.6.1/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/9.6.1/firebase-messaging.js');

if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('../firebase-messaging-sw.js')
      .then(function(registration) {
        console.log('Registration successful, scope is:', registration.scope);
      }).catch(function(err) {
        console.log('Service worker registration failed, error:', err);
      });
    }

firebase.initializeApp({
    messagingSenderId: "[redacted]",
  })

const initMessaging = firebase.messaging()
```

Figura 35: firebase-messaging-sw.js
Font: Elaboració pròpia amb IntellIJ.

A partir d'aquí, ja podem registrar a firebase cada usuari amb un token, i així, depenent dels tokens passats en cada notificació, podem notificar a uns usuaris o uns altres. Com veiem a la Figura 36.

```
useEffect(() => {
  const msg = getMessaging(firebase);
  getToken(msg).then((data)=>{
    if (data) {
      // Send the token to your server and update the UI if necessary
      cookies.set('token', data, {path: "/"});

    } else {
      // Show permission request UI
      console.log('No registration token available. Request permission to generate one.');
      // ...
      swal("Si us plau","Permet les notificacions per a poder rebre les comunicacions de l'organització", "error");
    }

  }).catch((err) => {
    console.log('An error occurred while retrieving token. ', err);
  });

}, []);
```

Figura 36: Obtenir token de registre a "Login.jsx"
Font: Elaboració pròpia amb IntellIJ.

Per acabar, només queda enviar la notificació des del Front-End com a organització per a que tots aquests usuaris registrats amb token i que ens interessin, puguin rebre-la amb la configuració anteriorment esmentada.

```
await axios.post('https://fcm.googleapis.com/fcm/send', body,{headers:
{ "Content-Type": "application/json",
  "Authorization": "key
}})
.then(response=>{
    console.log("resposta " +data);
    return response.data;
}).then(response=>{
    console.log("THEN id: "+JSON.stringify(response));

})
.catch(error=>{
    console.log("ERROR "+error);
    swal("Oops!","Ho sentim, alguna cosa ha sortit malament","error");
```

Figura 37: Enviar notificació a "EnviaNotificacio.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.2.2.5 Twitter / Facebook / Whatsapp

Finalment, hem integrat les API de Twitter, Facebook i Whatsapp per a l'apartat de Publicacions.
Gràcies a aquesta integració, podem compartir les publicacions que fem a VoluntariApp a qualsevol d'aquestes xarxes socials.
Tot això és possible gràcies a la llibreria "react-share".

Amb aquesta, l'únic que hem de fer per utilitzar aquestes API és utilitzar les anotacions que ens donen, personalitzant els paràmetres depenent de cada cas, com podem veure a la Figura 38.

```
<Form.Group size="lg" controlId="descripcio"style={sizeTitol} >
    <Form.Label >Comparteix!  <span>     </span></Form.Label>
    <FacebookShareButton url = {"voluntariapp.com"} quote = {name+"\n"+ descripcio} hashtag={"#VoluntariApp"}>
      <FacebookIcon width = "35" round ="true"/>
    </FacebookShareButton>  

    <WhatsappShareButton url = {"voluntariapp.com"} title = {name+"\n"+ descripcio} >
      <WhatsappIcon width = "35" round ="true"/>
    </WhatsappShareButton>  

    <EmailShareButton url = {""} subject = "name" body = {descripcio + "\n voluntariapp.com"} >
      <EmailIcon width = "35" round ="true"/>
    </EmailShareButton>  

    <TwitterShareButton url = {"\n voluntariapp.com"} title = {name+"\n"+ descripcio} via =  "VoluntariApp" >
      <TwitterIcon width = "35" round ="true"/>
    </TwitterShareButton>  

    <br/>
</Form.Group>
```

Figura 38: Crides a Twitter, Facebook, Twitter a "CrearPublicacio.jsx"
Font: Elaboració pròpia amb IntellIJ.

7.2.3 Patrons de disseny

Per acabar amb l'explicació de la implementació de la capa de presentació explicarem alguns dels patrons que s'han utilitzat en aquesta. Com és la capa de presentació, tots tenen a veure amb la disseny de la interacció.

7.2.3.1 Protecció

El primer patró de presentació que hem utilitzat ha estat el patró **Protecció**[11] .
Aquest patró realitza accions per ajudar a l'usuari a utilitzar el sistema software. En concret, quan l'usuari fa una acció que, si es fa de forma no desitjada, pot ser costosa de revertir o impossible.
La solució a aquest problema que aporta el patró és afegir un nivell extra de protecció per a que l'usuari hagi de confirmar l'acció com a pas extra.

Hem utilitzat aquest patró per a ajudar als usuaris a prendre les decisions més importants com, per exemple, quan hem de bescanviar una experiència, ja que si es fa les xapes que costa li desapareixeran a l'usuari. Ho podem veure a la figura 39:

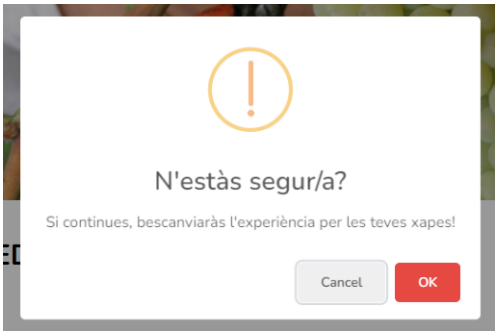


Figura 39:Vista del patró Protecció.
Font: Elaboració pròpia amb React.

7.2.3.2 Faceted Navigation

El següent patró utilitzat ha estat l'anomenat **Faceted Navigation**.[12]
Aquest patró s'utilitza perquè molts cops l'usuari voldrà filtrar una llista de ítems (com en el nostre cas els voluntariats o experiències) per cert tret característic i no veure'ls tots ja que els altres no l'interessen.
Aquest problema es soluciona permetent als usuaris obtenir els ítems desitjats de manera progressiva seleccionant i filtrant per les característiques concretes d'aquests ítems.

En el nostre cas, hem utilitzat inputs de forma de text per a poder buscar aquests ítems, per nom o localització.
Ho podem veure en les següents figures:

Figura 40: Vista del patró Faceted Navigation abans de buscar.
Font: Elaboració pròpia amb React.

Figura 41: Vista del patró Faceted Navigation despres de buscar.
Font: Elaboració pròpia amb React.

7.2.3.3 Main Navigation

Finalment, explicarem el **Main Navigation.**[13]
Aquest patró és utilitzat ja que l'usuari ha de saber on pot trobar les funcionalitats que està buscant.
La solució és senzilla, i com diu el nom del patró, és crear un menú navegacional en algun lloc visible de l'aplicació, com en el nostre cas la barra lateral.
En el nostre cas concret com hem dit, podem veure a la Figura 42 la barra lateral del voluntari.

Figura 42:Vista del patró Main Navigation en un voluntari

Font: Elaboració pròpia amb React.

7.3 API Back-End

Tal com hem comentat anteriorment, per a crear el Back-End estem utilitzant el framework de *Spring*[14] anomenat *Spring Boot.*[15]
Aquest framework utilitza el llenguatge de programació *Java.*[16]
L'elecció d'aquest framework també va ser definida a l'assignatura de PAE, i he volgut respectar-la de la mateixa manera que amb el Front-End, pero d'aquesta capa no hi havia cap desenvolupament fet.
La decisió es va dur a terme pel següent motiu: *Spring Boot* ens permet compilar les aplicacions web com a un arxiu .jar, per lo que podrem executar-les com una aplicació java normal, i això és possible gràcies a que el framework ens configura el servidor conjuntament amb l'aplicació, el que ens estalvia molta feina i ens deixa dedicar-nos al que ens interessa, el desenvolupament.

Tal com està fet aquest framework, hem decidit que la millor forma d'organitzar el Back-End és amb una arquitectura de serveis, als que puguem accedir per a fer lògica de negoci.

7.3.1 API VoluntariApp

En aquest projecte, hem fet servir el Back-End com a una API REST, amb la qual ens comuniquem desde el Front-End per a fer tota la lògica de negoci de l'aplicació, i retornar la informació requerida.
Ho hem desenvolupat d'aquesta manera per la seva facilitat d'utilització i per què funciona molt bé per la idea de negoci que tenim en ment, ja que cada funcionalitat estarà activa independentment de les altres.

Com podem observar en la Figura 43, on s'han tatxat les parts que no ens interessen per aquest punt, podem veure que el nostre Back-End està dividit en diferents carpetes.
Primer, els *controllers*, que es dediquen a rebre les peticions que se'ls hi fan mitjançant HTTP i mapejar-les amb el codi corresponent, que està als *services.* Aquests es dediquen a realitzar tota la lògica de negoci.
Els *models* són les entitats que coneix el Back-End, i amb les que interactua.
Els *DTO* són aquelles parts dels models que s'han de transportar, és a dir, que es relacionen amb les API i retornen dades.
I per acabar, els *repositoris*, que com els DTO i les entitats dels models, són els que es dediquen a la Persistència per a fer operacions, però els explicarem més a fons al següent punt 7.4.

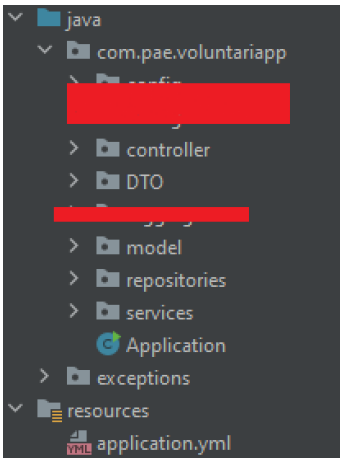


Figura 43: Directoris Back-End.
Font: Elaboració pròpia amb IntellIJ.

Està organitzat d'aquesta manera ja que és la més utilitzada per al desenvolupament d'aquest tipus d'aplicacions. Els controllers fan la feina que hem explicat abans, es comuniquen amb els serveis, que fan la lògica de negoci comunicant-se amb els models i la persistència mitjançant els repositoris, i retorna la informació necessària amb un DTO.

7.3.1.1 Codi i funcionament

Abans de passar a descriure la API, posarem un exemple de codi de cada part / directori, començant pels controllers.

El **controlador** que podem veure a la Figura 44 és el VoluntariatController.
Com podem observar, Spring Boot funciona amb anotacions precedides per una '@' per a fer certes accions. Les més importants, són les següents.
@RestController per a indicar que estem davant d'una classe que funciona com a Controlador de la API REST.
@CrossOrigin per a indicar a quins mètodes es poden accedir des de l'exterior.
@Post/Put/GetMapping per a indicar de quin tipus és la petició.
@Autowired per a mantenir un lligam entre les diferents classes, és a dir, et permet injectar dependències dins d'altres (explicació més a fons al punt 7.3.4).

```java
package com.pae.voluntariapp.controller;

import ...

@RestController
@CrossOrigin(origins = "*", methods= {RequestMethod.GET,RequestMethod.POST,RequestMethod.PUT,RequestMethod.DELETE})
public class VoluntariatController {

    @Autowired
    VoluntariatService voluntariatService;

    @PostMapping(value = "/creaVoluntariat")
    public ResponseEntity<Voluntariat> creaVoluntariat(@RequestBody Voluntariat objeto) {
        voluntariatService.creaVoluntariat(objeto);
        return new ResponseEntity(objeto, HttpStatus.OK);
    }

    @GetMapping(value ="/getNotaOrgVol")
    public ResponseEntity<Float> getNotaOrgVol(@RequestParam long idOrg ){

        return new ResponseEntity(voluntariatService.getNotaOrgVol(idOrg), HttpStatus.OK);
    }
```

Figura 44: VoluntariatController
Font: Elaboració pròpia amb IntellIJ.

A continuació, tenim els **serveis**. El que se'ns mostra a la Figura 45 és el dels Voluntariats, *VoluntariatService*.
Per a què Spring sàpiga que és un servei, ho indiquem amb l'anotació *@Service*:

```java
@Service
public class VoluntariatService {
```

Figura 45: VoluntariaService
Font: Elaboració pròpia amb IntellIJ.

Per a explicar un exemple específic, veiem el mètode del Servei anomenat *getNotaOrgVol.*
Aquest mètode s'ha cridat des del controlador i, com podem veure, està cridant a un repositori per a obtenir dades (voluntariats que son d'un usuari), fa operacions i retorna un float.

```java
public float getNotaOrgVol(long idOrg) {

    List<Voluntariat> esd = voluntariatRepository.findByOrganitzacioUserId(idOrg);
    float notaMitja = 0.0f;
    float sumNotes = 0.0f;
    int numFeedbacks = 0;
    //Tots els voluntariats de l'organització
    for(Voluntariat vol : esd){
        //Tots els feedbacks de cada vol.
        for(Feedback fed : vol.getFeedbacks()){
            sumNotes += fed.getRating();
            ++numFeedbacks;
        }
    }
    if(numFeedbacks != 0)notaMitja = sumNotes/numFeedbacks;

    return notaMitja;
}
```

Figura 46: mètode getNotaOrgVol de la classe VoluntariaService.
Font: Elaboració pròpia amb IntellIJ.

Les parts que ens falten les explicarem a l'apartat de Persistència, ja que creiem que han d'anar en aquell apartat perquè utilitzem l'API de Java Persistence API.

Aquesta estructuració és la més adient per al flux de treball que necessita la nostra aplicació, ja que desde la capa de presentació podrem demanar el que vulguem i deixar que el negoci faci el gruix de la lògica, communicant-se entre ell de la millor forma per facilitar aquesta forma de fer.
Controller -> Service ->(Persistència)->Service->Controller

7.3.1.2 Descripció de la API

Seguidament, passarem a explicar la nostra API.
Per fer un cop d'ull ràpid a com està organitzada, podem mirar la captura de pantalla següent (Figura 47):

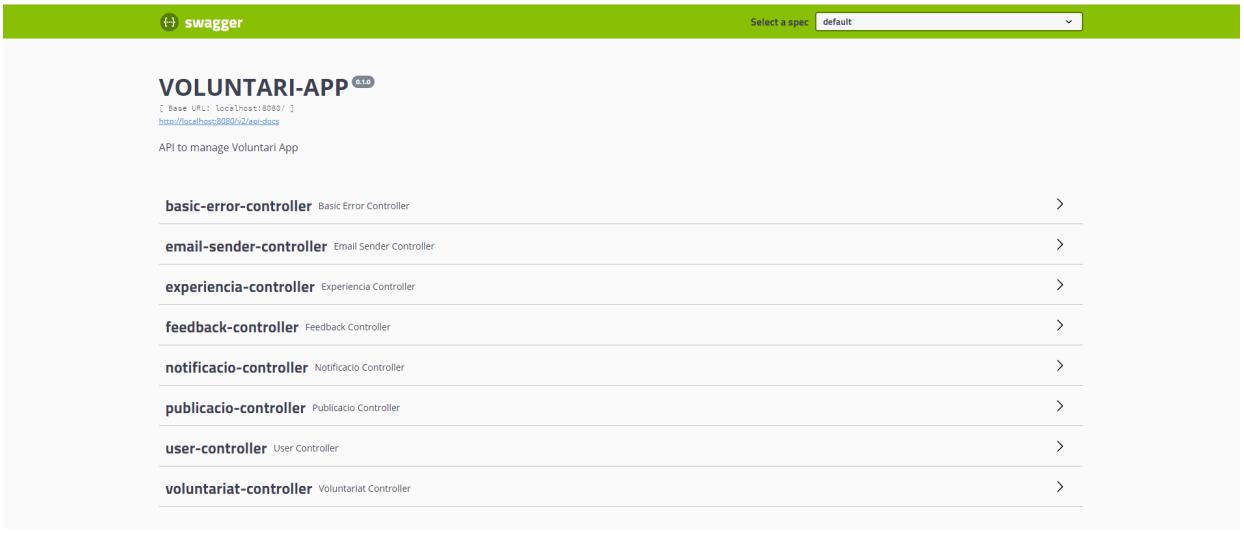


Figura 47: API VoluntariApp.
Font: Elaboració pròpia amb la API de swagger.

Com podem observar, tenim 7 controladors diferents, cadascun dedicat a un grup de funcionalitats que ataquen el mateix objecte (Voluntariat, Experiència, Feedback, Notificació, Publicació, Usuari, Correu Electrònic). I un extra per controlar els errors bàsics.
En aquest projecte hem definit un total de 44 *endpoints*. 1 per a les funcionalitats del Correu, 11 per als Voluntariats, 9 per a les Experiències, 4 de Feedback, 5 de Publicació, 12 d'Usuari i 2 de notificació.
A continuació, els passarem a explicar en detall per a conèixer la API en la seva totalitat.

Abans d'entrar en detall, cal dir que com aquest projecte tenia un temps limitat, hem hagut de prioritzar les funcionalitats més complexes devant les que no ho eren tant. Per aquest motiu ens hem estalviat d'implementar alguna de les funcionalitats més simples d'un *CRUD*[17], com per exemple algunes edicions (Updates) i esborrades (Deletes).

7.3.1.2.1 Voluntariat

Farem una llista amb una petita descripció de cada *endpoint*[18]dels que podem observar en la (Figura 48):

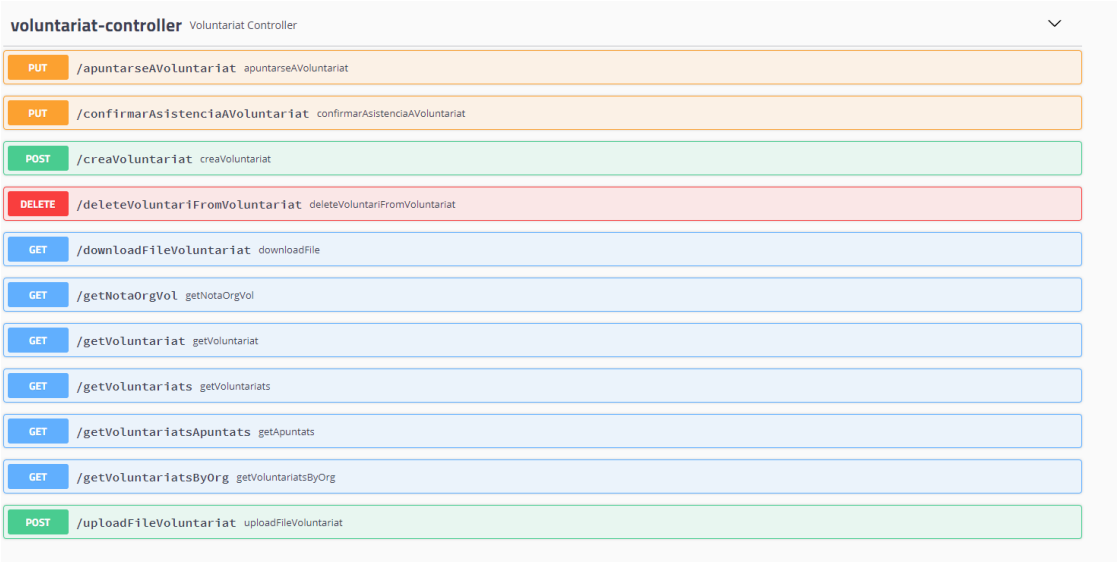


Figura 48: Controlador Voluntariat.
Font: Elaboració pròpia amb la API de swagger.

- **/apuntarseAVoluntariat**
  - **PUT** amb paràmetres: *integer* voluntariId, *integer* voluntariatId.
  - El voluntari amb id voluntariId s'apunta al voluntariat amb voluntariatId.
  - **RETORNA** el voluntariat.
- **/confirmarAsistenciaAVoluntariat**
  - **PUT** amb paràmetres: *integer* voluntariId, *integer* voluntariatId.
  - El voluntari amb id voluntariId confirma l'assistència al voluntariat amb voluntariatId.
  - **RETORNA** el voluntariat.
- **/creaVoluntariat**
  - **POST** amb body un voluntariat..
  - Es crea un voluntariat.
  - **RETORNA** el voluntariat.

- **/uploadFileVoluntariat**
  - **POST** amb paràmetres: *string* fileURL, *integer* id.
  - Es crea una imatge amb URL fileURL i s'assigna al voluntariat amb id *id*.
  - **RETORNA** el voluntariat.
- **/downloadFileVoluntariat**
  - **GET** amb paràmetres: *string* fileId.
  - Retorna la Imatge amb id fileId.
  - **RETORNA** la Imatge.
- **/getNotaOrgVol**
  - **GET** amb paràmetres: *integer* idOrg.
  - Obté la nota dels tots els feedbacks que han fet els voluntaris en voluntariats de l'organització amb id *idOrg*.
  - **RETORNA** la nota.
- **/getVoluntariat**
  - **GET** amb paràmetres: *integer* id.
  - Obté el el voluntariat amb id id.
  - **RETORNA** el voluntariat.

- **/getVoluntariats**
  - **GET**
  - Obté tots els voluntariats posteriors a la data actual.
  - **RETORNA** tots els voluntariat.
- **/getVoluntariatsApuntats**
  - **GET** amb paràmetres: *integer* idVoluntari.
  - Obté tots els voluntariats en els que està inscrit idVoluntari.
  - **RETORNA** els voluntariats.
- **/getVoluntariatsByOrg**
  - **GET** amb paràmetres: *integer* idOrg.
  - Obté tots els voluntariats que organitza l'organització amb id idOrg.
  - **RETORNA** els voluntariats.
- **/deleteVoluntariFromVoluntariat**
  - **DELETE** amb paràmetres: *integer* idVoluntari, *integer* idVoluntariat.
  - El voluntari amb id idVoluntari es desapunta del voluntariat amb id idVoluntariat.
  - **RETORNA** el voluntariat.

7.3.1.2.2 Experiència

Com amb els Voluntariats, explicarem els *endpoints* de les Experiències de la (Figura 49):

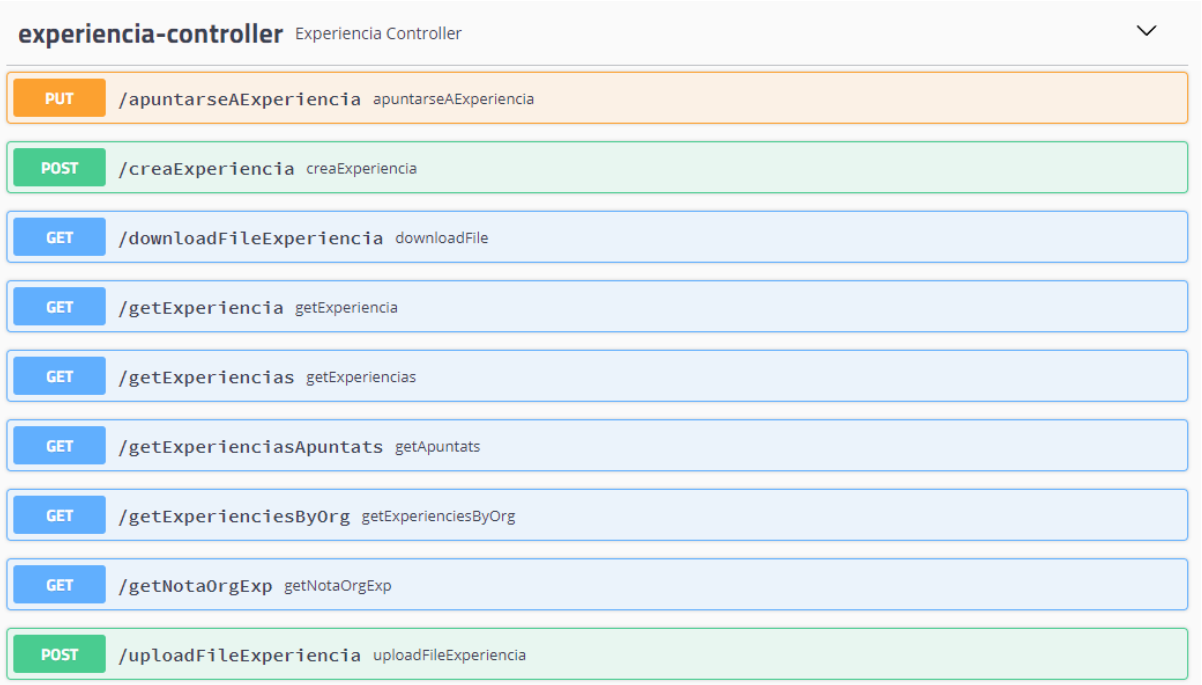


Figura 49: Controlador Experiència.
Font: Elaboració pròpia amb la API de swagger.

- **/apuntarseAExperiencia**
  - **PUT** amb paràmetres: *integer* voluntariId, *integer* experienciaId.
  - El voluntari amb id voluntariId s'apunta a l'experiència amb experienciaId.
  - **RETORNA** l'experiència.
- **/creaExperiencia**
  - **POST** amb body una experiència.
  - Es crea una experiència.
  - **RETORNA** l'experiència.
- **/uploadFileExperiencia**
  - **POST** amb paràmetres: *string* fileURL, *integer* id.
  - Es crea una imatge amb URL fileURL i s'assigna a l'experiència amb id *id*.
  - **RETORNA** l'experiència.
- **/downloadFileExperiencia**
  - **GET** amb paràmetres: *string* fileId.
  - Retorna la Imatge amb id fileId.
  - **RETORNA** la Imatge.
- **/getNotaOrgVExp**
  - **GET** amb paràmetres: *integer* idOrg.
  - Obté la nota dels tots els feedbacks que han fet els voluntaris en experiències de l'organització amb id *idOrg*.
  - **RETORNA** la nota.
- **/getExperiencia**
  - **GET** amb paràmetres: *integer* id.
  - Obté l'experiència amb id id.
  - **RETORNA** l'experiència.
- **/getExperiencias**
  - **GET**

    - Obté totes les experiències.
    - **RETORNA** totes les experiències.
- **/getExperienciasApuntats**
    - **GET** amb paràmetres: *integer* idVoluntari.
    - Obté totes les experiències que ha bescanviat el voluntari amb id idVoluntari.
    - **RETORNA** les experiències.
- **/getExperienciesByOrg**
    - **GET** amb paràmetres: *integer* idOrg.
    - Obté totes les experiències que organitza l'organització amb id idOrg.
    - **RETORNA** les experiències.

7.3.1.2.3 Usuari

A continuació, a la (Figura 50) veiem els *endpoints* de les funcionalitats que tenen que veure amb els Usuaris. També els explicarem.

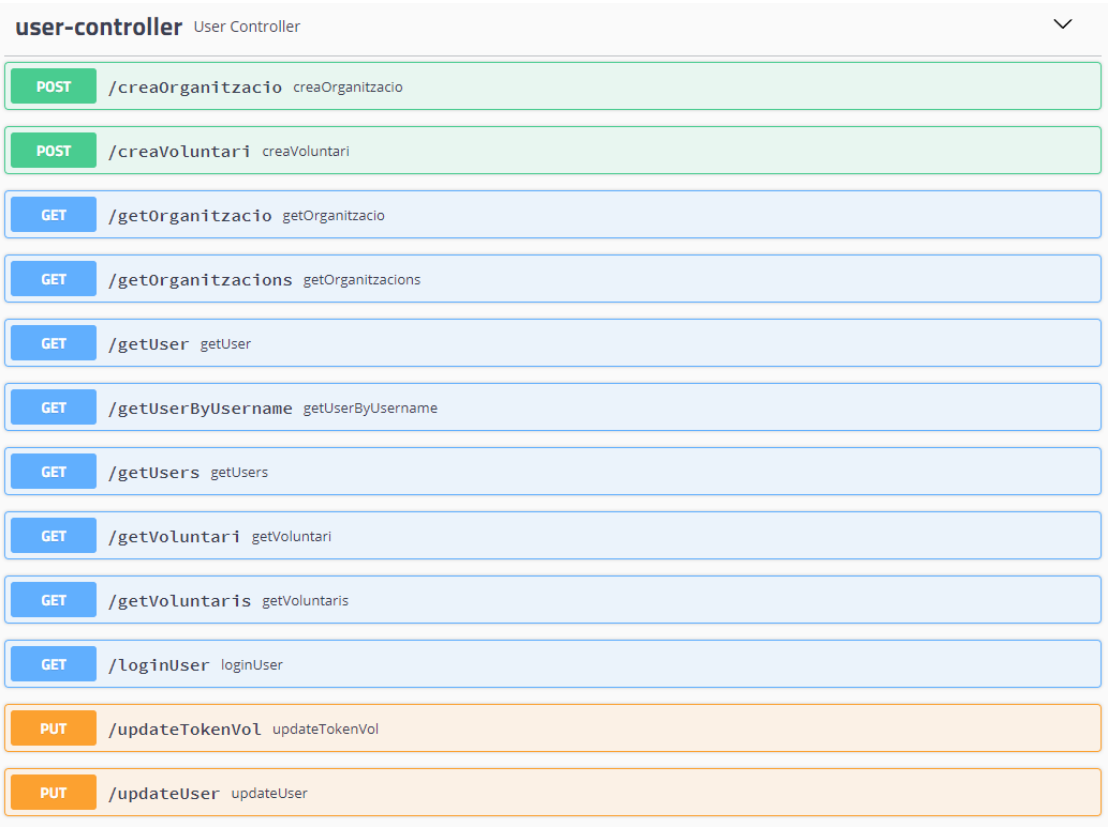


Figura 50: Controlador Usuari.
Font: Elaboració pròpia amb la API de swagger.

- **/updateTokenVol**
    - **PUT** amb paràmetres: *integer* id, *integer* token.
    - S'actualitza el token de l'usuari que servirà per a enviar-li notificacions.
    - **RETORNA** l'experiència.
- **/updateUser**
    - **PUT** amb paràmetres: *integer* idUser, *string* description, *string* email, *string* name.
    - S'actualitza els camps indicats del voluntari amb id idUser
    - **RETORNA** l'experiència.

- **/creaVoluntari**
  - **POST** amb body un voluntari.
  - Es crea un voluntari.
  - **RETORNA** el voluntari.
- **/creaOrganitzacio**
  - **POST** amb body una organització.
  - Es crea una organització.
  - **RETORNA** l'organització.
- **/getVoluntari**
  - **GET** amb paràmetres: *integer* id.
  - Obté el voluntari amb id id.
  - **RETORNA** el voluntari.
- **/getOrganitzacio**
  - **GET** amb paràmetres: *integer* id.
  - Obté l'organització amb id id.
- **/getUser**
  - **GET** amb paràmetres: *integer* id.
  - Obté l'usuari amb id id.
  - **RETORNA** l'usuari.
- **/getVoluntaris**
  - **GET**
  - Obté tots els voluntaris.
  - **RETORNA** el voluntaris.
- **/getOrganitzacios**
  - **GET**
  - Obté totes les organitzacions.
  - **RETORNA** les organitzacions.
- **/getUsers**
  - **GET**
  - Obté tots els usuaris.
  - **RETORNA** els usuaris.
- **/getUserByUsername**
  - **GET** amb paràmetres: *string* username.
  - Obté l'usuari que té com a username username..
  - **RETORNA** les experiències.
- **/loginUser**
  - **GET** amb paràmetres: *string* username, string password.
  - Si existeix un usuari amb username username i contrasenya password, retorna l'usuari per a fer login.
  - **RETORNA** l'usuari per a fer login.

7.3.1.2.4 Notificació

Com en tots els anteriors casos, veurem els *endpoints* de Notificació a la Figura 51 i els descriurem.

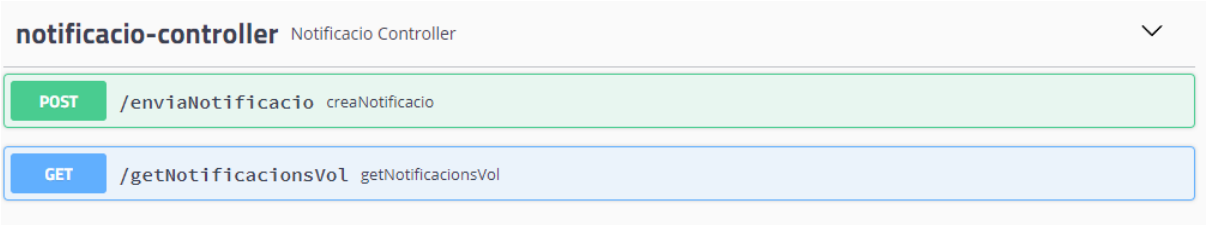


Figura 51: Controlador Notificació.
Font: Elaboració pròpia amb la API de swagger.

- **/enviaNotificacio**
  - **POST** amb body una notificació.
  - Es crea una notificació.
  - **RETORNA** la notificació.
- **/getNotifiacionsVol**
  - **GET** amb paràmetres: *integer* idVoluntariat.
  - Obté totes les notificacions que s'han enviat als voluntaris del voluntariat amb id idVoluntariat.
  - **RETORNA** la notificació.

7.3.1.2.5 Publicació

Podem veure, també, els *endpoints* del controlador de Publicació a la Figura 52.

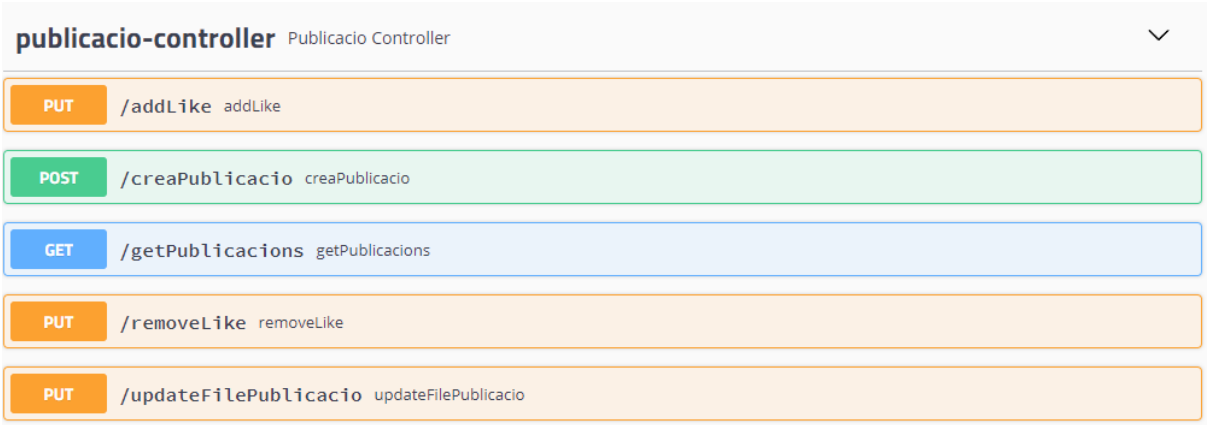


Figura 52: Controlador Publicació.
Font: Elaboració pròpia amb la API de swagger.

- **/addLike**
  - **PUT** amb paràmetres: *integer* PublicacioId, *integer* UserId.
  - L'usuari amb id UserId li dona like a la publicació amb id PublicacioId.
  - **RETORNA** la publicació.
- **/removeLike**
  - **PUT** amb paràmetres: *integer* PublicacioId, *integer* UserId.
  - L'usuari amb id UserId li treu el like a la publicació amb id PublicacioId.
  - **RETORNA** la publicació.
- **/updateFilePublicació**
  - **PUT** amb paràmetres: *string* fileURL, *integer* id.
  - Es crea una imatge amb URL fileURL i s'assigna a la publicació amb id *id*.
  - **RETORNA** la publicació.

- **/creaPublicació**
  - **POST** amb body una publicació.
  - Es crea una publicació.
  - **RETORNA** la publicació.
- **/getPublicacions**
  - **GET**
  - Obté totes les publicacions.
  - **RETORNA** les publicacions.

7.3.1.2.6 Feedback

Ara veurem i explicarem els *endpoints* del controlador de Feedback (Figura 53):

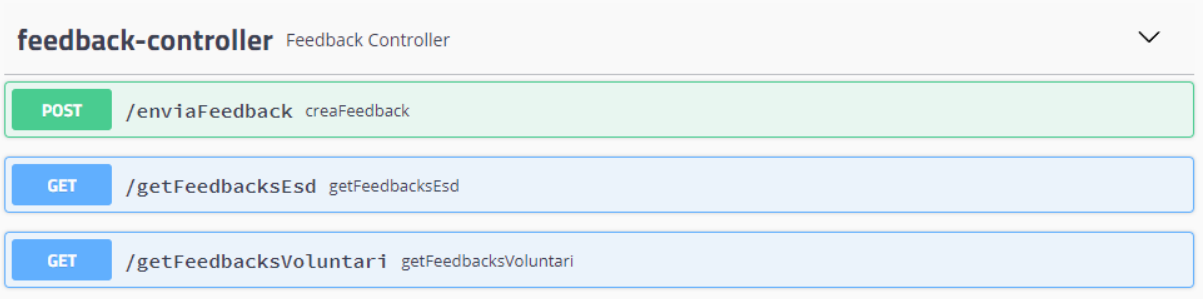


Figura 53: Controlador Feedback.
Font: Elaboració pròpia amb la API de swagger.

- **/enviaFeedback**
  - **POST** amb paràmetres *integer* esdevenimentId, *integer* voluntariId, *integer* rating, *string* comentaris.
  - Es crea un feedback.
  - **RETORNA** el feedback.
- **/getFeedbacksEsd**
  - **GET** amb paràmetres *integer* esdevenimentId
  - Obté tots els feedbacks que hagin fet els voluntaris a l'esdeveniment amb id esdevenimentId.
  - **RETORNA** els feedbacks.
- **/getFeedbacksVoluntari**
  - **GET** amb paràmetres *integer* idVoluntari
  - Obté tots els feedbacks que hagi fet el voluntari amb id idVoluntari.
  - **RETORNA** els feedbacks.

7.3.1.2.7 Email

Finalment, per a poder enviar correus quan es crea una Experiència, hem creat el controlador d'Email. El podem veure a la Figura 54:

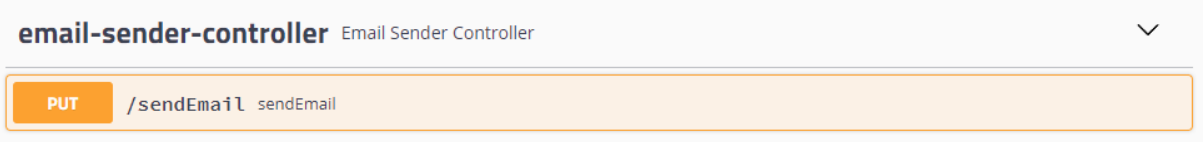


Figura 54: Controlador Email.
Font: Elaboració pròpia amb la API de swagger.

- **/sendEmail**
  - **PUT** amb paràmetres: *integer* experienciaId, *integer* voluntariId.
  - El voluntari amb id voluntariId rebrà un correu electrònic amb la informació necessària per a bescanviar l'experiència amb id experienciaId.
  - **RETORNA** l'experiència.

7.3.2 Llibreries i APIs utilitzades

Per a poder facilitar-nos segons quines tasques, o per a poder implementar segons quines funcionalitats al Back-End, hem hagut d'utilitzar llibreries i APIs externes, i en aquest apartat les descriurem breument.

7.3.2.1 Swagger

La primera API que hem integrat amb l'aplicació ha sigut ***Swagger.***[19]
Swagger s'utilitza, com hem vist al punt anterior, per a visualitzar d'una manera clara i poder probar l'API que estem creant. Tenint una interfície amigable i que ens facilita les tasques de desenvolupament i testeig.
És necessari configurar-la, i ho hem fet a partir d'una classe *SwaggerConfiguration* que podem veure a la Figura 55.
Com podem veure, utilitzem les etiquetes *@Configuration* i *@EnableSwagger2* per a indicar la configuració i activació de Swagger.

```
@EnableSwagger2
@Configuration
public class SwaggerConfiguration {

    //Swagger ui en http://localhost:8080/swagger-ui.html
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                //.apis(RequestHandlerSelectors.basePackage("com.pae.voluntariapp.controller"))
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build()
                .apiInfo(apiInfo());

    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("VOLUNTARI-APP")
                .description("API to manage Voluntari App")
                .version("0.1.0")
                .build();
    }
}
```

Figura 55: SwaggerConfiguration
Font: Elaboració pròpia amb IntellIJ.

7.3.2.2 SLF4J

La segona API que utilitzem és la de ***slf4j***, és a dir, la de Simple Logging Facade For Java. Aquesta API ens permet afegir "logs" a l'execució de l'aplicació per a controlar tot el que ens

interessi durant el desenvolupament. És molt útil per a poder fer debugs, saber els estats de les variables i classes, obtenir la informació que necessitem…
Aquest mètode implementa un patró de tipus *Factoria*[20] per a crear logs (ho expliquem amb més detall a l'apartat 7.3.4).
A la Figura 56 observem la classe *LoggingController* a la que es pot cridar per a crear Logs.
A la Figura 57 podem veure com s'instància desde el VoluntariatController, i a la Figura 58, com podem utilitzar el mètode per obtenir la informació.

```
package com.pae.voluntariapp.logging;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * Classe per a desenvolupament, serveix per a generar logs adaptats a cada classe, instanciant-la i cridant a getLogger(Class).
 */

public class LoggingController {

    public LoggingController(){

    }

    public Logger getLogger(Class<?> cla) { return LoggerFactory.getLogger(cla); }

    /*@RequestMapping("/")
    public String index() {
        logger.trace("A TRACE Message");
        logger.debug("A DEBUG Message");
        logger.info("An INFO Message");
        logger.warn("A WARN Message");
        logger.error("An ERROR Message");

        return "Howdy! Check out the Logs to see the output...";
    }*/
}
```

Figura 56: LoggingController
Font: Elaboració pròpia amb IntellIJ.

```
LoggingController logCont = new LoggingController();
Logger logger =  logCont.getLogger(VoluntariatController.class);
```

Figura 57: Instància de loggingController a la classe VoluntariatController
Font: Elaboració pròpia amb IntellIJ.

```
logger.info("++++++++++++Voluntari id  "+idVoluntari);
```

Figura 58: Com utilitzar la instància per a crear un log.
Font: Elaboració pròpia amb IntellIJ.

7.3.2.3 SendGrid

Finalment, hem utilitzat la llibreria de correu d'**Spring *Mail* i *Email*** amb la API de ***SendGrid***[21].
D'aquesta manera, podem enviar un correu electrònic des del Back-End, personalitzant tots els aspectes necessaris per fer aquests correus dinàmics.
A la Figura 59 podem observar la configuració de la connexió smtp, i a la Figura 60 tenim un fragment de codi que s'encarrega de fer la crida a la API de SendGrid per a enviar el correu.

```yaml
mail:
  host: smtp.ethereal.email
  port: 587
  username: [redacted]@ethereal.email
  password: [redacted]
  properties:
    mail:
      smtp:
        auth: true
        starttls:
          enable: true
```

Figura 59: Configuració de la connexio smtp.
Font: Elaboració pròpia amb IntellIJ

```java
private void send(String from, String to, String subject, String content, VoluntariUser voluntariUser, Experiencia experiencia) throws IOExceptio

    Email frome = new Email(from);
    Email toe = new Email(to);
    //Content contente = new Content("text/plain", content);
    Mail mail = new Mail();
    DynamicTemplatePersonalization personalization = new DynamicTemplatePersonalization();
    personalization.addTo(toe);

    mail.setFrom(frome);
    mail.setSubject(subject);

    personalization.addDynamicTemplateData("first_name", voluntariUser.getName());
    personalization.addDynamicTemplateData("experiencia_name", experiencia.getName());
    personalization.addDynamicTemplateData("experiencia_date", String.valueOf(experiencia.getData()));
    personalization.addDynamicTemplateData("experiencia_orgname", experiencia.getOrganitzacioUser().getName());
    personalization.addDynamicTemplateData("codi_experiencia", experiencia.getCodiExp());
    mail.addPersonalization(personalization);
    mail.setTemplateId("d-94ae09fe49994e8484b242f80607b5e1");
    logger.info("AAAAAAAA PRE crida api: ");
    SendGrid sg = new SendGrid( apiKey: "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\n");
    Request request = new Request();
    try {
        logger.info("try: ");
        request.setMethod(Method.POST);
        request.setEndpoint("mail/send");
        request.setBody(mail.build());
        Response response = sg.api(request);
        logger.info(response.getBody());
    } catch (IOException e) {
        logger.info("AAAAAAAA ERROR: "+e.getMessage());
    }
}
```

Figura 60: Métode send de la classe EmailService.
Font: Elaboració pròpia amb IntellIJ.

7.3.3 Tests

Per acabar amb el Back-End, i per assegurar-nos del correcte funcionament d'aquesta part de l'Aplicació, hem desenvolupat tests.
Hem decidit desenvolupar tests només a la part del Back-End per què és on és la lògica de negoci, i per tant és el que ens interessa que funcioni perfectament. Mentre que la lògica existent al Front-End és molt menys significativa.

Hem decidit implementar tests unitaris per a cada mètode rellevant de cada classe controladora de l'API per a confirmar el correcte funcionament de cada funcionalitat.
Per a poder fer tests, hem creat una base de dades de test que no s'utilitzarà més que per tests.
A la Figura 61 podrem veure quines classes hem creat per a testejar.

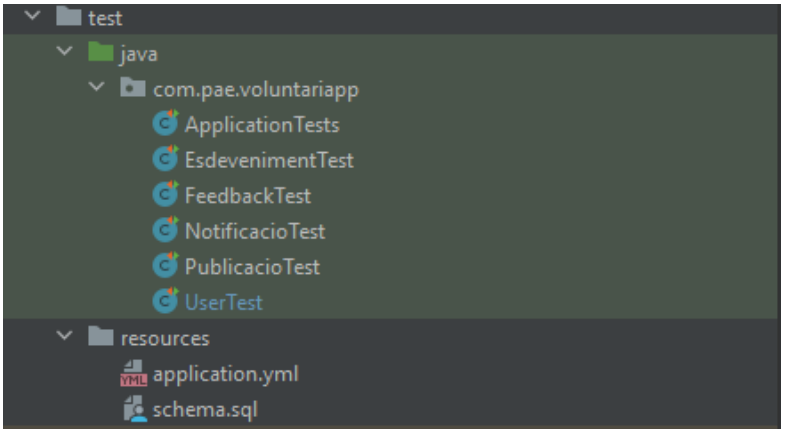


Figura 61: Carpeta test de VoluntariApp.
Font: Elaboració pròpia amb IntellIJ.

A continuació podem veure l'exemple de la classe UserTest, que s'encarrega de testejar les funcionalitats dels Usuaris.
Com veiem en la Figura 62, per escriure tests a SpringBoot necessitem l'anotació *@SpringBootTest*, que és la que fa que interpreti la classe com a test.

```
@SpringBootTest
class UserTest {
```

Figura 62: Classe UserTest.
Font: Elaboració pròpia amb IntellIJ.

Seguidament, podem observar que el mètode *testVoluntari* de la classe anterior ha d'estar marcat com a *@Test*, i a partir d'aquí desenvolupar el codi necessari. En aquest cas, es crea el voluntari a partir de la crida a la UserController.

```
@Test
void testVoluntari(){
    VoluntariUser voluntariUser = new VoluntariUser();

    voluntariUser.setXapes(100);
    voluntariUser.setDescription("M'encanten els voluntariats");
    voluntariUser.setEmail("a@b.com");
    voluntariUser.setName("Carlos");
    voluntariUser.setUsername("username"+d);

    ResponseEntity<String> entity = userController.creaVoluntari(voluntariUser);
    assertEquals(entity.getBody(), actual: "Voluntari is created");
}
```

Figura 63: Mètode testVoluntari de la classe UserTest.
Font: Elaboració pròpia amb IntellIJ.

Per a executar el codi només hem de fer un "Run Test" i confirmar que ha passat el test a la consola de l'*IntellIJ (Figures 64 i 65)*.

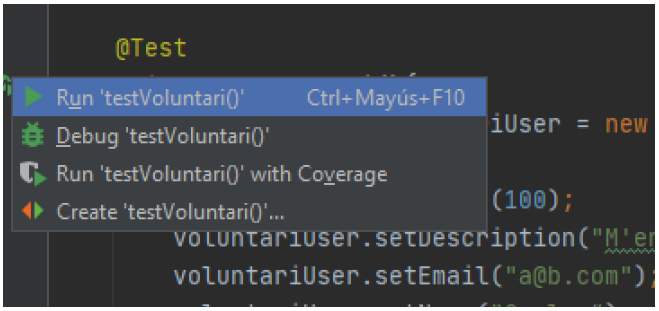


Figura 64: Run testVoluntari de la classe UserTest.
Font: Elaboració pròpia amb IntellIJ.

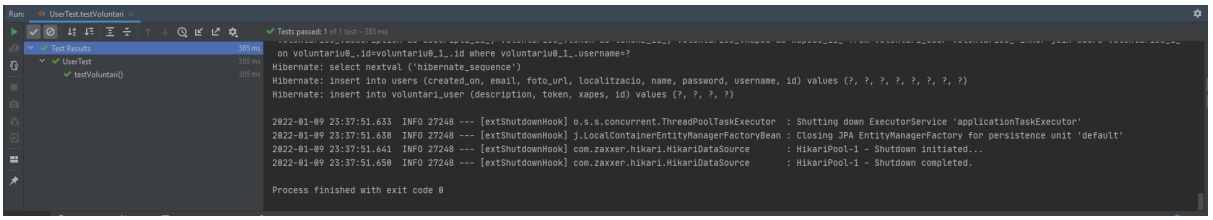


Figura 65: Resultat de córrer el testVoluntari de la classe UserTest.
Font: Elaboració pròpia amb IntellIJ.

Per a comprovar que tots els tests passen correctament, els passem tots junts.
En total, tenim 21 mètodes test inclosos en 6 classes de test diferents.

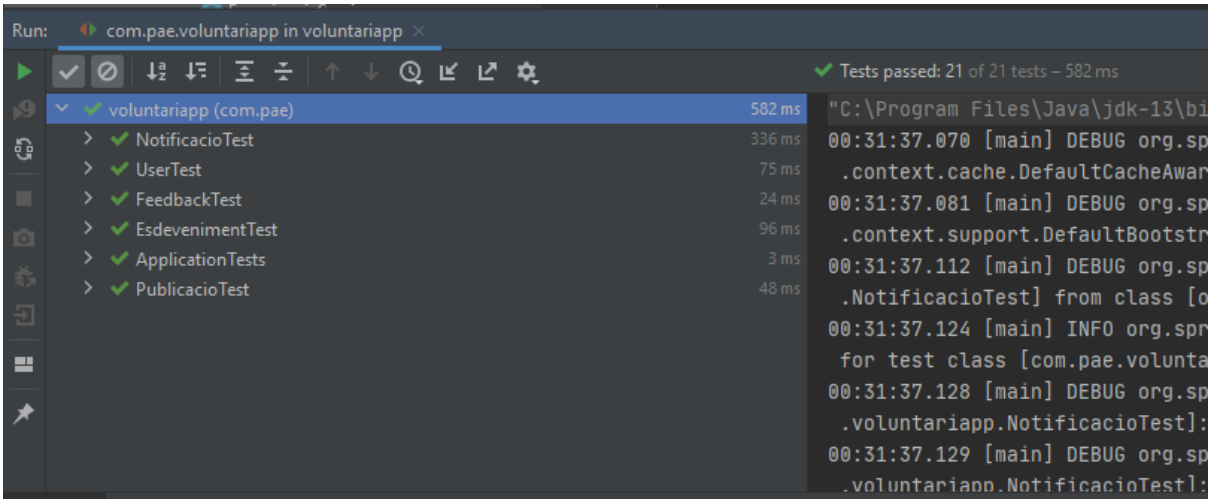


Figura 66: Run tots els tests de l'aplicació.
Font: Elaboració pròpia amb IntellIJ

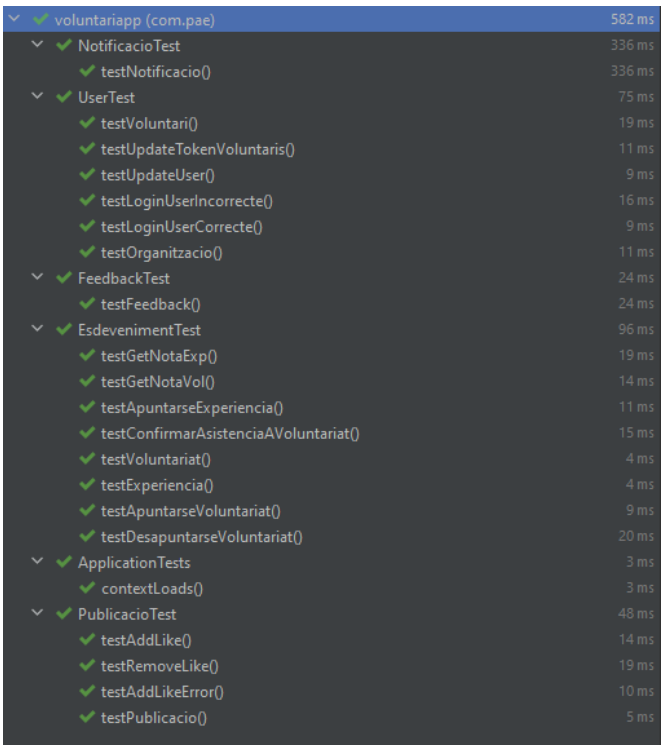


Figura 67: Run tots els tests de l'aplicació en detall.
Font: Elaboració pròpia amb IntellIJ

7.3.4 Patrons de disseny

En aquest subapartat explicarem els diferents patrons de disseny que hem utilitzat en el Back-End de la nostra aplicació.

7.3.4.1 Injecció de Dependències

Com hem mencionat anteriorment, el primer patró que hem utilitzat ha estat el patró **Injecció de Dependències**. Aquest patró ens el facilita el framework que fem servir, Spring.

La injecció de dependències ens ajuda a separar el nostre codi per responsabilitats, és a dir, que extreu la responsabilitat de la creació d'objectes d'un component i se l'encarrega a un altre.
A Spring, per utilitzar aquest patró només hem d'indicar a la declaració de la classe que volem injectar l'anotació *@Autowired* (hi ha un exemple de codi a la Figura 44 a l'apartat 7.3.1.1). A la Figura 68 podem fer-nos una idea de com injecta aquestes dependències.
Gràcies a aquest patró, podem oblidar-nos de la creació d'objectes per a acoplar les classes que necessitem utilitzar de manera conjunta.

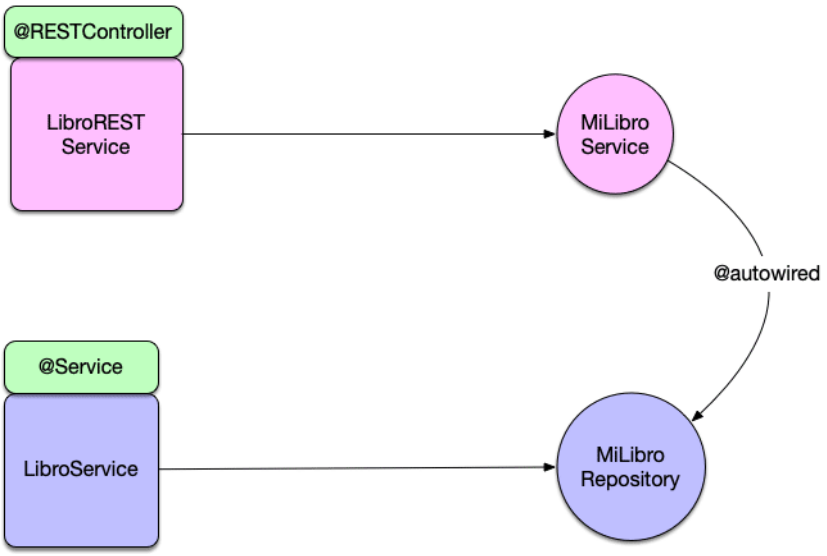


Figura 68: Esquema @autowired entre un Repositori i un Servei.
Font: [49]

7.3.4.2 Factoria

Un altre patró que també hem mencionat i que hem fet servir és el patró **Factoria**.
Aquest patró l'utilitzem per a delegar la creació d'instàncies dels logs. Utilitzem un LoggerFactory que és cridat des del LoggerController i que crida a l'obtenció de la instància sense haver d'instanciar-lo ell mateix. Podem veure el codi a les Figures 56, 57, 58 de l'apartat 7.3.2.
Fent-ho així, tenim una manera segura i dinàmica de crear instàncies per a fer logs en cada classe, ja que la creació serà una o una altre depenent de la classe passada per paràmetre.

7.3.4.3 Expert

El tercer patró que estem fent servint a la capa de domini o negoci és el patró **Expert**.
Amb aquest patró, assignem a la responsabilitat de creació d'un objecte o la implementació d'un mètode a aquella classe que en coneix tota la informació.
En l'exemple següent, podem veure un fragment de la classe *Esdeveniment*, la coneixedora de tota la informació, i observem que la lògica d'afegir voluntaris a l'esdeveniment recau en el métode *addVoluntaris* de la mateixa classe.

```
public void addVoluntaris(EsdevenimentVoluntari voluntaris) {
    if(this.voluntaris == null){
        this.voluntaris = new ArrayList<>();
    }
    this.voluntaris.add(voluntaris);
}
```

Figura 69: Esdeveniment
Font: Elaboració pròpia amb IntellIJ

7.3.4.4 Controlador

El quart i últim patró que estem fent servint a la capa de domini o negoci és el patró **Controlador**.
Amb aquest patró, aïllem el negoci de la presentació, i el que fem és crear una classe intermitja, el controlador. Aquest s'encarrega de rebre les dades de l'usuari, i els envia a la classe corresponent segons el que s'hagi de fer amb aquestes dades.
A la figura següent, podrem veure com la nostra classe controladora *PublicacioController* fa de classe intermitja i reparteix la informació:

```
@RestController
@CrossOrigin(origins = "*", methods= {RequestMethod.GET,RequestMethod.POST,RequestMethod.PUT,RequestMethod.DELETE})
public class PublicacioController {

    @Autowired
    PublicacioService publicacioService;

    @PostMapping("/creaPublicacio")
    public ResponseEntity<Publicacio> creaPublicacio(@RequestBody Publicacio objeto) {
       return publicacioService.creaPublicacio(objeto);

    }

    @GetMapping("/getPublicacions")
    public ResponseEntity<List<DTOPublicacio>> getPublicacions(){

        return publicacioService.getPublicacions();
    }

    @PutMapping("/updateFilePublicacio")
    public ResponseEntity<Publicacio> updateFilePublicacio(@RequestParam("fileURL") String file,@RequestParam("id") long id){

        return publicacioService.updateFilePublicacio(file, id);
    }
```

Figura 70: PublicacióController
Font: Elaboració pròpia amb IntellIJ

7.3.4.5 Serveis

Encara que és un patró estructural, també cal esmentar que estem utilitzant el patró **Serveis** o **Service Layer**.
Aquest patró soluciona una problemàtica d'arquitectura del domini, i organitza l'execució de totes les funcionalitats de manera que en cada Servei (o capa lògica del servei) hi hagi totes les funcionalitats que comparteixin objectes o que tinguin a veure.
En el nostre cas, hem separat en serveis les funcionalitats que ataquen sobre cada objecte. A la Figura 71 podem veure l'exemple del servei de Publicació:

```java
@Service
public class PublicacioService {

    @Autowired
    PublicacioRepository publicacioRepository;
    @Autowired
    ImagesRepository repositoryImg;
    @Autowired
    UserRepository userRepository;

    LoggingController logCont = new LoggingController();
    Logger logger =  logCont.getLogger(ObjetoPruebaController.class);

    public ResponseEntity<Publicacio> creaPublicacio(Publicacio objeto) {
        User u = userRepository.findById(objeto.getAutor().getId());
        objeto.setAutor((OrganitzacioUser) u);
        publicacioRepository.save(objeto);
        logger.info("++++++AUTOR POST name:" +objeto.getAutor().getName());
        return new ResponseEntity(objeto, HttpStatus.OK);
    }
```

Figura 71: Fragment de PublicacióService
Font: Elaboració pròpia amb IntellIJ

7.4 Persistència i Base de Dades

Per a la capa de persistència de la nostra aplicació, hem aprofitat una API que és molt fàcil d'utilitzar i que s'acopla molt bé amb l'ecosistema de *Spring.* Aquesta és la Java Persistence API (JPA).
JPA és una especificació que indica com s'hauria de programar la persistència dels objectes en Java. Per tant, l'hem d'implementar o usar un framework que la implementi.
En el nostre cas utilitzem Spring JPA, una solució perfecta per al nostre projecte ja que inclou *Hibernate*[22] per a implementar JPA, pel que no haurem d'implementar aquesta persistència, i alhora s'adapta a l'ecosistema d'*Spring* d'una forma senzilla.
A més, *Spring JPA* ens afavoreix la utilització del patró repositori, que explicarem en el punt 7.4.3, i que és molt útil per a crear aplicacions basades en els serveis.
Spring també inclou JDBC, el que ens facilita accedir a les dades i és molt útil per a fer consultes natives.
Per aquests motius l'hem triat com a la millor opció per a VoluntariApp.

7.4.1 Codi i funcionament

Per a utilitzar aquesta persistència i incloure-la en l'aplicació, haurem de configurar-la enllaçant-la amb la base de dades triada.
Hem triat una base de dades SQL, com és PostgreSQL, ja que és de codi obert i per al tipus d'aplicació que estem desenvolupant, les bases de dades relacionals funcionen molt bé, i la integració amb les tecnologies utilitzades és senzilla.
De la mateixa manera que amb el Front-End i el Back-End, a PAE vam decidir que, de posar-se en marxa el projecte, el faríem amb aquest tipus de BBDD, i he respectat la decisió i m'he encarregat d'instal·lar el servidor, el client, i de crear i poblar aquesta BBDD.

A la figura següent podem veure com està configurat aquest enllaç:

```
spring:
  jpa:
    database: POSTGRESQL
    show-sql: true
    hibernate:
      ddl-auto: update
    properties:
        hibernate:
          jdbc:
            non_contextual_creation: true
  datasource:
    platform: postgres
    url: jdbc:postgresql:[redacted]
    username: [redacted]
    password: [redacted]
    driverClassName: org.postgresql.Driver
```

Figura 72: Configuració de la persistència.
Font: Elaboració pròpia amb IntellIJ

A continuació, el que necessitem per a utilitzar les dades a la persistència és mapejar les entitats que definirem en aquesta amb les relacions, taules i atributs existents a la base de dades relacional.
Amb *Spring*, el que fem és mapejar els objectes com a entitats, com podem veure a la Figura 73.

En aquesta figura, estem veient la classe Esdeveniment, que és una classe abstracta de la que hereden Voluntariat i Experiència.
Com hem comentat en l'apartat del Back-End, *Sprint* utilitza anotacions per funcionar, i *Sprint JPA* també.
Utilitzem *@Entity* per a indicar que aquesta classe és una entitat de la base de dades.
@Table per indicar el mapeig amb la nostra base de dades relacional, és a dir, a la BBDD PostgreSQL la taula a la que ens referim l'anomenarem "esdeveniment".
L'anotació *@JsonIdentifyInfo* s'utilitza per a evitar les referències circulars que es puguin provocar quan es llegeixi.
@Id indica quin serà l'identificador d'aquest objecte, el que seria la clau primària a la BBDD. I el generem amb una estrategia automàtica.
@Column s'utilitza per a indicar el mapeig entre la columna de la BBDD i l'atribut de la classe.
Per a crear relacions entre les diferents classes, i alhora entre les diferents entitats de la BBDD, existeixen diferents anotacions,
L'anotació que podem observar a la figura, *@ManyToOne,* es refereix a la relació molts a un, el que significa que, en l'exemple de la figura, poden existir molts esdeveniments per cada usuari organitzador. És a dir, una organització pot organitzar molts esdeveniments diferents.

```java
@Entity
@Table(name= "esdeveniment")
@Inheritance(
        strategy = InheritanceType.JOINED
)
@JsonIdentityInfo(
        generator = ObjectIdGenerators.PropertyGenerator.class,
        property = "id")
public abstract class Esdeveniment {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "name")
    private String name;

    @Column(name = "descripcio", length = 500)
    private String descripcio;

    @Column(name = "localitzacio")
    private String localitzacio;

    @Column(name = "data")
    private Date data;

    @ManyToOne
    private OrganitzacioUser organitzacioUser;
```

Figura 73: Classe Esdeveniment.
Font: Elaboració pròpia amb IntellIJ

L'anotació @OneToMany representa la mateixa relació pero a l'inrevés, per si volem guardar la informació a l'altre banda de la relació.
En aquest cas, la relació és entre esdeveniment i els seus feedbacks, per tenir-los a mà al domini.

```java
@OneToMany(mappedBy = "esdeveniment",
        cascade = CascadeType.ALL)
private List<Feedback> feedbacks;
```

Figura 74: Classe Esdeveniment atribut feedbacks.
Font: Elaboració pròpia amb IntellIJ

Per a la relació que s'explica amb una relació de molt a molts, creem classes extres, no usem anotacions.
Com veiem a la Figura 75, la classe EsdevenimentVoluntari té un identificador que és alhora una classe que conté les claus foranes de l'usuari i l'esdeveniment. Aquest és un clar exemple d'una relació de molts a molts entre dues entitats traduït a un objecte Java.
Per a que el framework ho reconegui com una classe associativa hem d'utilitzar *@AssociationOverrides* amb les columnes que volem que siguin les claus foranes referenciades (en aquest cas, l'id d'usuari i l'id d'esdeveniment).
A més, en la següent figura podem veure l'última anotació per a l'última relació que ens falta definir, el *@OneToOne*.
La relació d'un a un s'indica amb aquesta anotació, on s'han d'explicitar l'Entitat a la que ens referim i, si volem, el tipus d'estratègia de cascada que volem implementar.

```java
@Entity
@Table(name = "rel_esdeveniment_voluntari")
@AssociationOverrides({
        @AssociationOverride(name = "primaryKey.user",
                joinColumns = @JoinColumn(name = "fk_voluntari")),
        @AssociationOverride(name = "primaryKey.esdeveniment",
                joinColumns = @JoinColumn(name = "fk_esdeveniment")) })
public class EsdevenimentVoluntari {

    public EsdevenimentVoluntari(){}
    // composite-id key
    private VoluntariEsdevenimentId primaryKey = new VoluntariEsdevenimentId();

    @EmbeddedId
    public VoluntariEsdevenimentId getPrimaryKey() { return primaryKey; }

    @Column(name = "ha_assistit",columnDefinition="Boolean default false", nullable = true)
    private Boolean haAssistit = false;

    @Column(name = "token", nullable = true)
    private String token;

    private Feedback feedback = new Feedback();

    @OneToOne(targetEntity = Feedback.class,mappedBy = "esdevenimentVoluntari", cascade=CascadeType.ALL)
    public Feedback getFeedback(){return feedback;}

    public void setPrimaryKey(VoluntariEsdevenimentId primaryKey) { this.primaryKey = primaryKey; }
```

Figura 75: Classe EsdevenimentVoluntari.
Font: Elaboració pròpia amb IntellIJ

7.4.2 Llibreries i APIs utilitzades

També hem utilitzat diferents llibreries i API per la persistència. De la mateixa manera que en els apartats anteriors, les enumerarem i explicarem a continuació.

7.4.2.1 Spring JPA

Tal com hem dit abans, JPA no ens ofereix cap classe sobre la qual treballar, ens ofereix unes interfícies les quals podem fer servir per a implementar aquesta capa de persistència, però encara hauriem d'implementar-la.
Spring JPA ens ofereix una llibreria que inclou tot el necessari per a utilitzar *Hibernate* com a implementació de JPA, pel que no hauriem d'implementar-la, i s'adapta als projectes Spring, ja que és específic d'aquests, pel que va perfecte per a la nostra aplicació en *Spring Boot*.

Com la persistència s'ha explicat en profunditat al punt anterior, ja hem vist que aquesta API l'utilitzem per a generar la persistència amb les annotacions de *Spring*, pel que no entrarem més en detall.

7.4.2.2 PostgreSQL

L'altre API que hem utilitzat en la capa de persistència és la de PostgreSQL.
Això ha sigut necessari per a poder connectar aquesta capa amb la base de dades i poder accedir-hi. La connexió està explicada al punt anterior i es pot veure a la Figura 72.

7.4.3 Patrons de disseny

Per acabar amb la implementació, parlarem dels patrons de disseny utilitzats a la capa de persistència.
Com hem utilitzat una API per a generar la persistència, en tenim dos.

7.4.3.1 Repository

Una aplicació web necessita accedir a les dades diversos cops per a obtenir o desar informació.
Si aquestes dades estan emmagatzemades en una base de dades relacional, com en el nostre cas, seria convenient que el nostre negoci no accedís a la gestió d'aquesta informació de manera directa, per això serveix la capa de persistència.
Però, a més, necessitem persistir totes les classes que referencien a les entitats amb les modificacions adients, i segons aquest patró, la millor forma de dur a terme aquesta pràctica és mitjançant repositoris[50] .
Un repositori s'encarrega d'encapsular la lògica requerida per accedir a les dades. Cada repositori es centra en unes funcionalitats relacionades, en el nostre cas, un per objecte.

Com estem generant la persistència, i estem utilitzant *Spring JPA*, el que hem fet per a incloure aquest patró de la manera més senzilla i complementària és utilitzar la interfície CrudRepository que ens proporciona el propi *Spring, que s'encarrega de totes les operacions CRUD de la base de dades*.

A la figura següent podem veure tots els repositoris creats en el nostre projecte. Un per objecte.

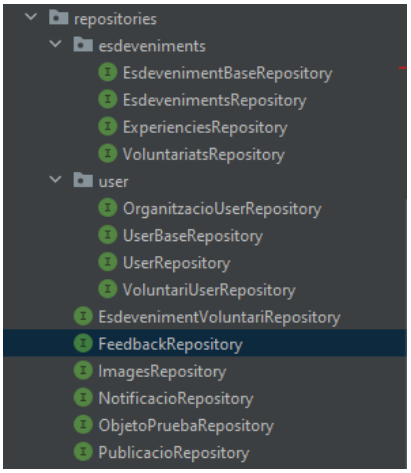


Figura 76: Directori repositories.
Font: Elaboració pròpia amb IntellIJ

Per a posar un exemple de la seva implementació, el que podem veure és que hem creat una classe interfície que estén aquest CrudRepository i que servira com a classe a la que invocar des del domini.
Aquesta classe és el *FeedbackRepository*, l'encarregada de fer el CRUD per a l'objecte Feedback. Ho podem veure a la Figura 77.

```
@Repository
public interface FeedbackRepository extends CrudRepository<Feedback, String>{
    List<Feedback> findAll();
    List<Feedback> findById(long id);

    List<Feedback> findByEsdevenimentId(long idVoluntariat);

    Feedback findByEsdevenimentVoluntariPrimaryKeyUserIdAndEsdevenimentVoluntariPrimaryKeyEsdevenimentId(long voluntariId, long esdevenimentId);
}
```

Figura 77: FeedbackRepository.
Font: Elaboració pròpia amb IntellIJ

En el següent exemple podem veure com des de un mètode del servei, s'invoca aquest repository i aquest, per la persistència, fa una consulta a base de dades.
Gràcies al CrudRepository podem fer una búsqueda a BBDD amb la clàusula findBy, que es tradueix en un *SELECT*. Si acompanyem aquesta clàusula amb els camps que volem filtrar i li passem per paràmetre els valors d'aquells camps pels quals volem que es filtri, podrem obtenir tots els registres desitjats sense necessitat de fer una Query (encara que la podrem fer si aquesta té una complexitat superior).

```
esdevenimentVoluntariRepository.findByPrimaryKeyUserIdAndPrimaryKeyEsdevenimentId(voluntariId,esdevenimentId);
```

Figura 78: creaFeedback a FeedbackService.
Font: Elaboració pròpia amb IntellIJ

Finalment, per a guardar els registres a la BBDD, hem de fer una crida al métode save de les interfícies del CrudRepository després de crear o d'actualitzar els camps que ens interessin guardar:

```
feedbackRepository.save(objeto);
```

Figura 789: creaFeedback a FeedbackService.
Font: Elaboració pròpia amb IntellIJ

7.4.3.2 Data Access Object

Acabant amb la secció, tenim el patró Data Access Object.
Aquest patró consisteix en un objecte que s'encarrega de l'accés a les dades. Aquest és el DAO.
Aquest DAO és una classe que localitza dades per a nosaltres, però en el nostre cas, l'usarem per a poder accedir a les dades i retornar-les amb la informació justa que necessitem per al domini.

Com hem utilitzat un repositori com a patró per a accedir a les dades i ja no necessitem cobrir aquesta part, hem utilitzat aquest patró DAO per a retornar objectes de la BBDD cap al domini a la presentació per a no passar tot l'objecte existent a la base de dades i només passar el que ens interessa.

Com podem veure a l'exemple de la Figura 80, tenim la classe *DTOFeedback* per a representar l'entitat Feedback amb els camps que ens interessen.

```
public class DTOFeedback {

    private String usernameVol;
    private long id;
    private String comentaris;
    private Integer rating;
    private Date data;
    private Images imatge;

    private DTOEsdeveniment esdeveniment;
    private EsdevenimentVoluntari esdevenimentVoluntari;


    public DTOFeedback(){}
    public DTOFeedback(Feedback esdeveniment){

        this.id = esdeveniment.getId();
        this.data = esdeveniment.getData();
        this.comentaris = esdeveniment.getComentaris();
        this.rating = esdeveniment.getRating();
        if(esdeveniment.getImatge() != null) this.imatge = esdeveniment.getImatge();
        else this.imatge = null;
        this.esdeveniment = new DTOEsdeveniment(esdeveniment.getEsdeveniment());
        this.esdevenimentVoluntari = esdeveniment.getEsdevenimentVoluntari();
        this.usernameVol = esdeveniment.getEsdevenimentVoluntari().getUser().getUsername();
```

Figura 80: DTOFeedback.
Font: Elaboració pròpia amb IntellIJ

I aquesta classe es pot utilitzar per a que, com en l'exemple de la Figura 81, en el domini i després d'accedir al repositori, s'instancii i es retorni mitjançant la API aquest DAO.

```
public List<DTOFeedback> getFeedbacksVoluntari(long idVoluntari) {
    VoluntariUser voluntari = voluntariRepository.findById(idVoluntari);

    List<EsdevenimentVoluntari> lEsdVol = voluntari.getEsdeveniments();
    List<DTOFeedback> lFeed = new ArrayList<~>();
    for( EsdevenimentVoluntari esdVol : lEsdVol){

        if(esdVol.getFeedback() != null && !esdVol.getFeedback().getComentaris().isEmpty() ){
            logger.info("+++++esdVol.getFeedback().getComentaris():" +esdVol.getFeedback().getComentaris());
            lFeed.add(new DTOFeedback(esdVol.getFeedback()));
        }
    }

   return lFeed;
}
```

Figura 81: getFeedbacksVoluntari a FeedbackService.
Font: Elaboració pròpia amb IntellIJ

8. Manual d'usuari

Per a poder entendre la totalitat de l'aplicació i per a que qualsevol usuari pugui desenvolupar-se en ella, crearem un manual d'usuari amb captures de pantalla de l'aplicació, explicant-ne les possibilitats de cada una.
Això ho dividirem en dos seccions, una amb tot lo relacionat amb el Voluntari i una altre relacionada amb l'Organització.

8.1 Voluntari

8.1.1 Inici Sessió

Per començar, com a tota aplicació, ens trobem la pantalla d'Inici de sessió.
Per accedir com a voluntari, si no tenim compte, podem registrar-nos donant-li al botó de "Registra't per ser voluntari".

Figura 82: Inici Sessió.
Font: VoluntariApp.

En aquest apartat podem omplir totes les dades per a registrar-nos, i incloure una foto.

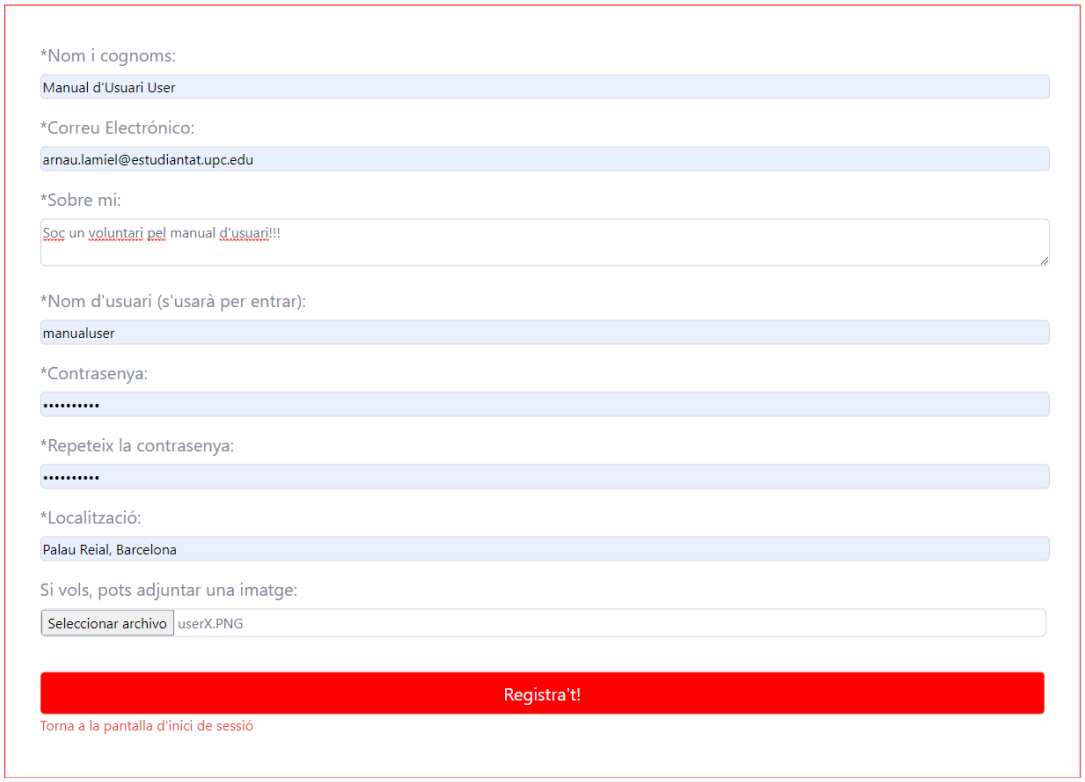


Figura 83: Registre.
Font: VoluntariApp.

Després d'omplir les dades i registrar-nos, tornarem a la pàgina d'inici de sessió i podrem accedir amb les nostres credencials.

Figura 84: Inici Sessió.
Font: VoluntariApp.

8.1.2 Publicacions

Quan hem accedit, ja podem veure totes les funcionalitats disponibles (a l'esquerra), i les publicacions fetes per cada organització (al centre).

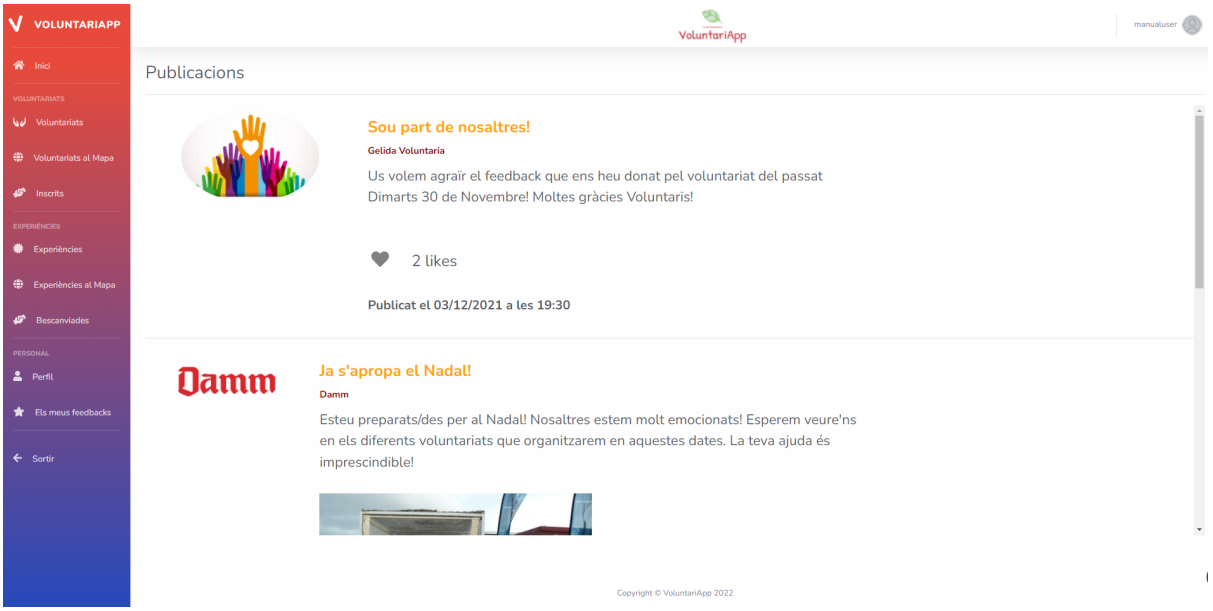


Figura 85: Inici.
Font: VoluntariApp.

Podem donar-li "m'agrada" a les publicacions que vulguem.

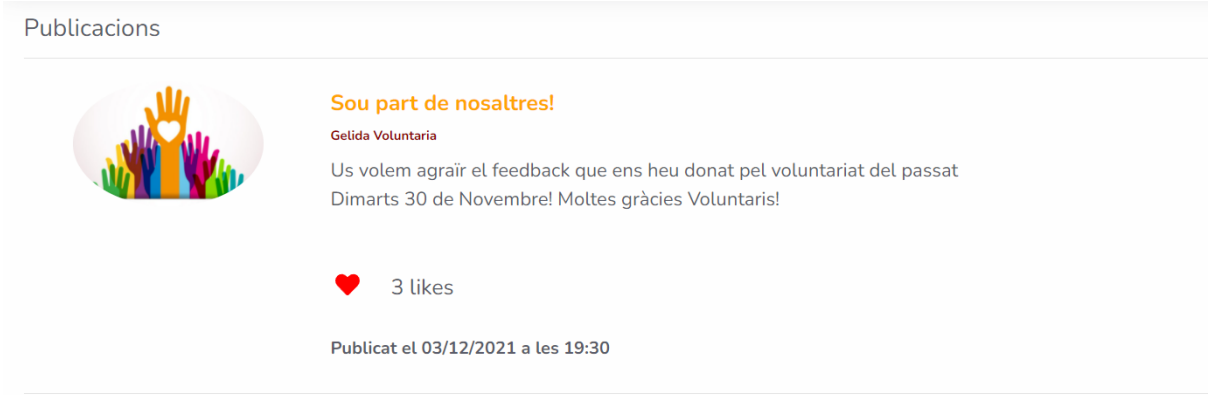


Figura 86: Publicació amb "m'agrada".
Font: VoluntariApp.

8.1.3 Voluntariats

Podem accedir a l'apartat "Voluntariats" per veure tots els voluntariats futurs als que ens podem apuntar.
A més, podem buscar per nom o per localització aquests voluntariats.

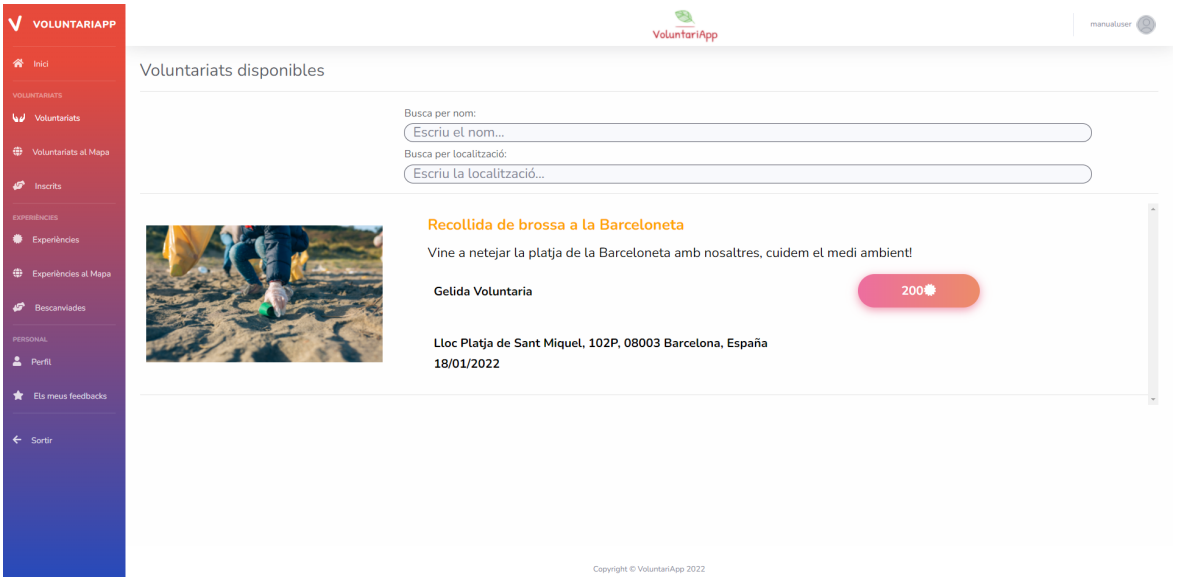


Figura 87: Voluntariats general.
Font: VoluntariApp.

També, accedint a l'apartat "Voluntariats al Mapa", podrem accedir sobre un mapa a tots els voluntariats.
A partir d'aquí també podrem accedir als voluntariats.

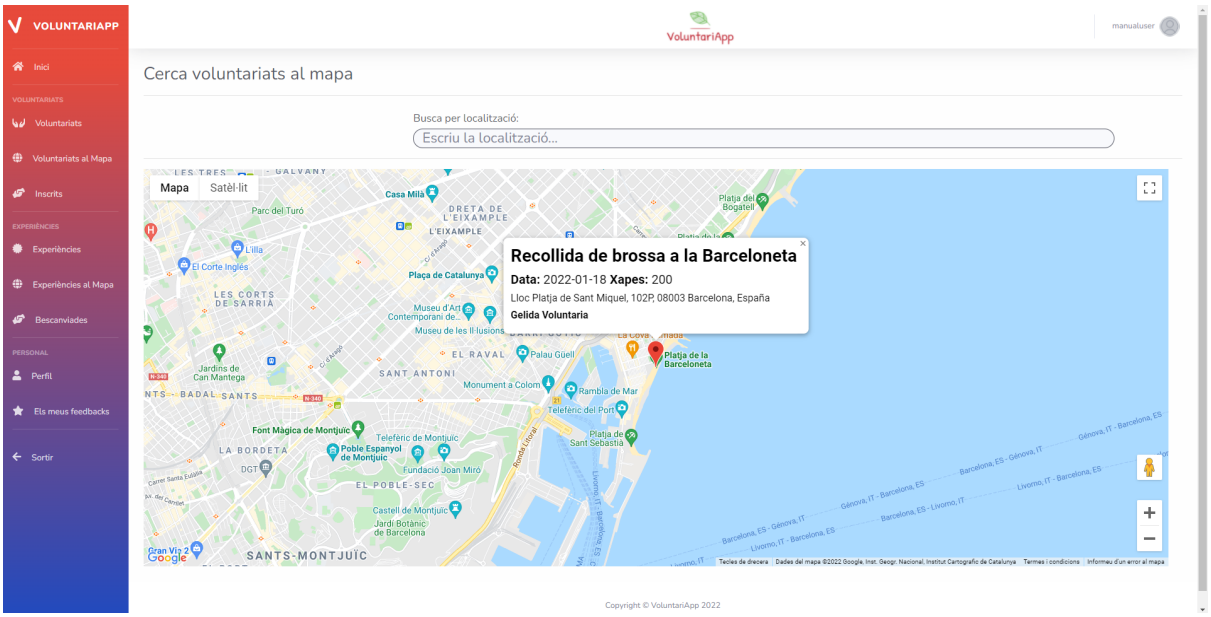


Figura 88: Voluntariats al Mapa.
Font: VoluntariApp.

Dins de cada voluntariat, podrem accedir a la informació d'aquest. Podrem veure el nom, descripció, data, hora, el número de xapes que obtindrem per fer-lo, l'organització que el fa, i la localització (en escrit i sobre un mapa). També podrem veure la puntuació que li han donat els voluntaris als anteriors voluntariats d'aquest organitzador.

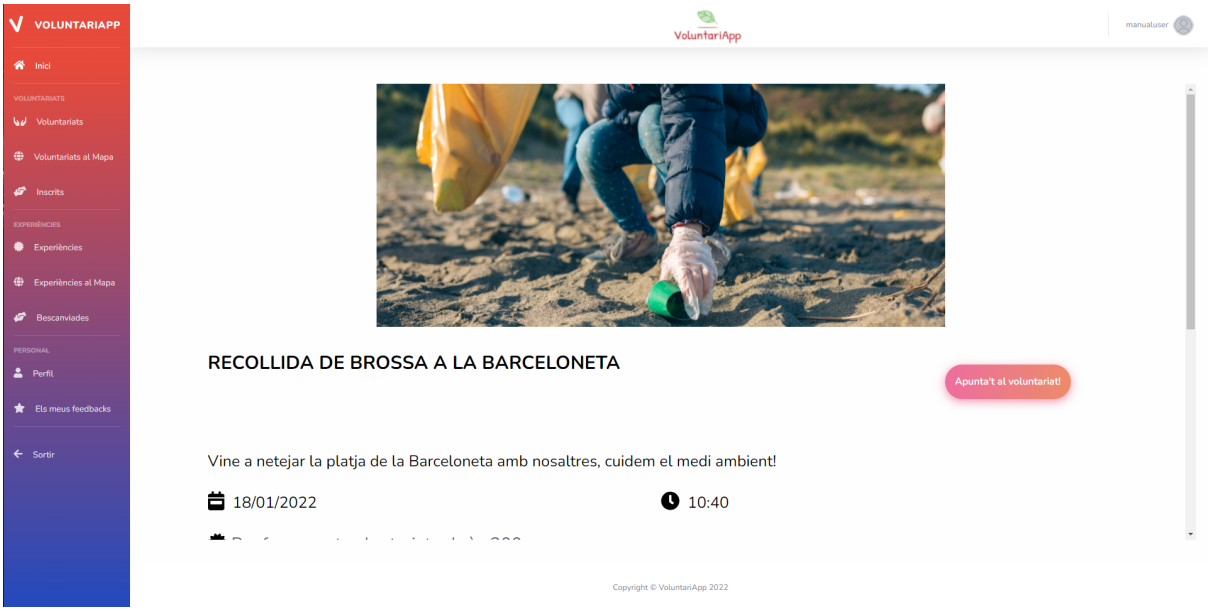


Figura 89: Voluntariat Específic 1.
Font: VoluntariApp.

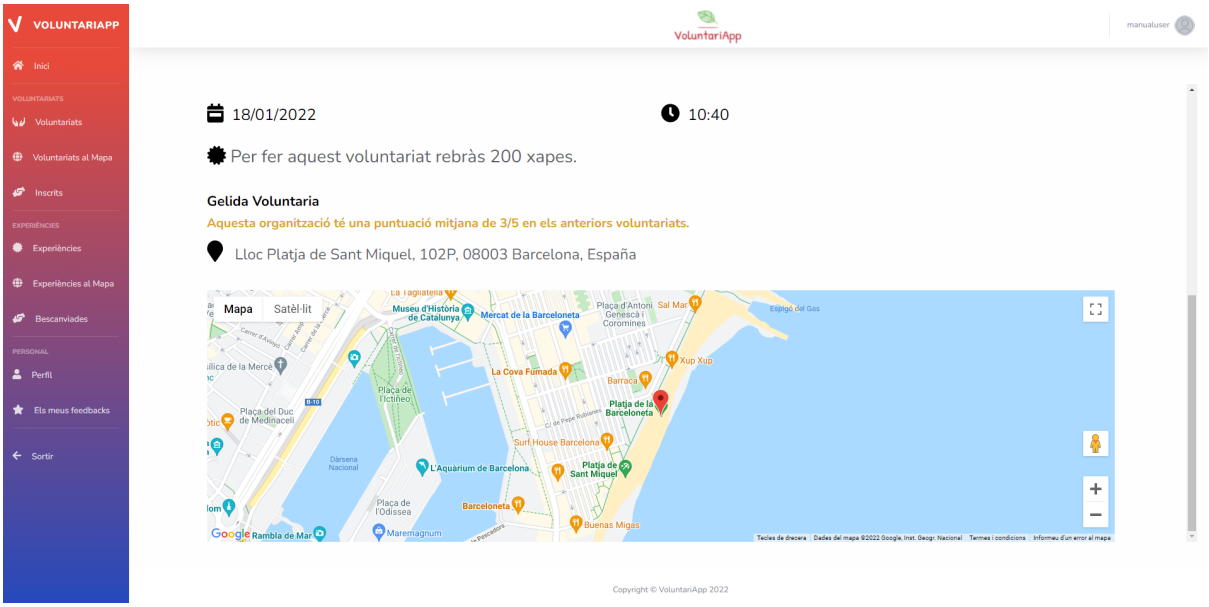


Figura 90: Voluntariat Específic 2.
Font: VoluntariApp.

Dins del voluntariat, si cliquem al botó de “Apunta’t al Voluntariat”, podrem apuntar-nos al voluntariat.

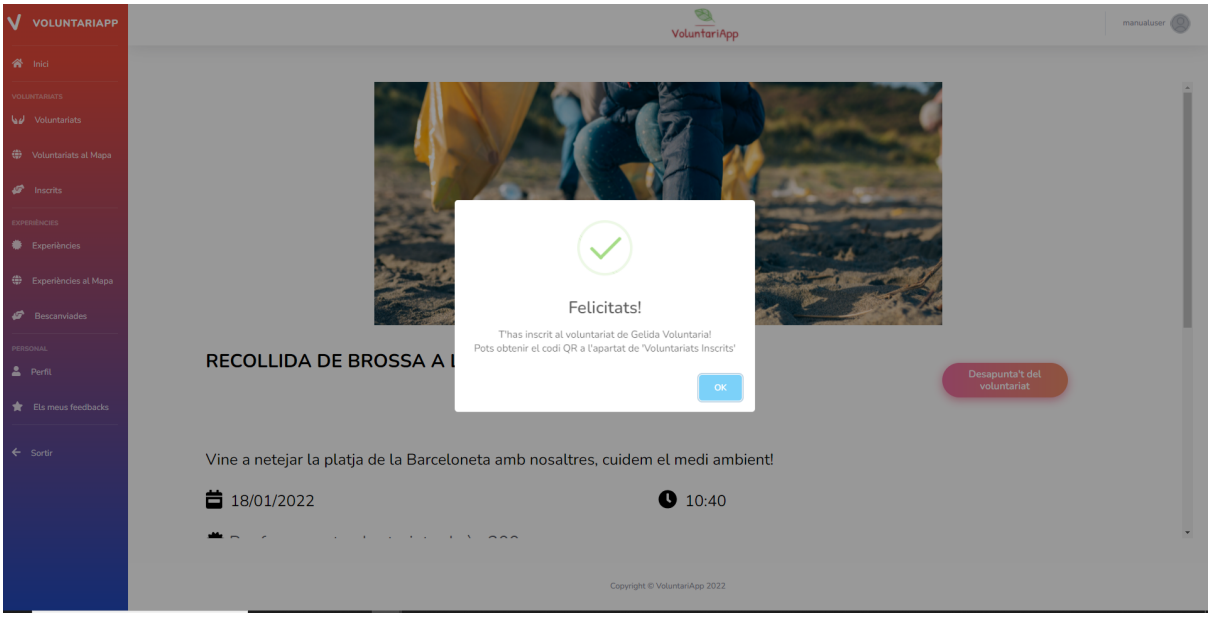


Figura 91: Apuntar-se al voluntariat.
Font: VoluntariApp.

A partir d'ara, podrem accedir a tots els voluntariats a través de l'apartat "Inscrits" dins de l'apartat "Voluntariats" de la barra lateral esquerra.

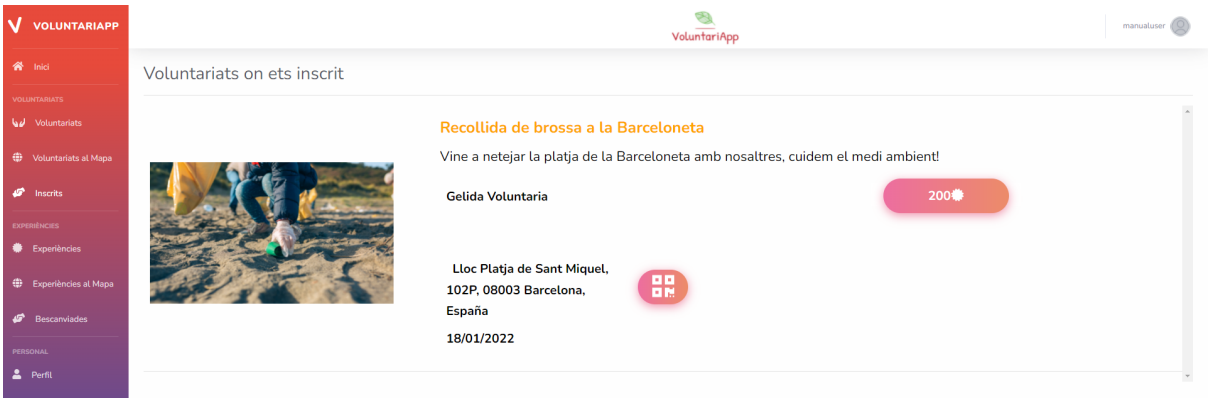


Figura 92: Inscrits.
Font: VoluntariApp.

Dins d'aquesta, podrem accedir al QR que ens permetrà confirmar el nostre voluntariat per part de l'organització. Si el confirmem, obtindrem les xapes.

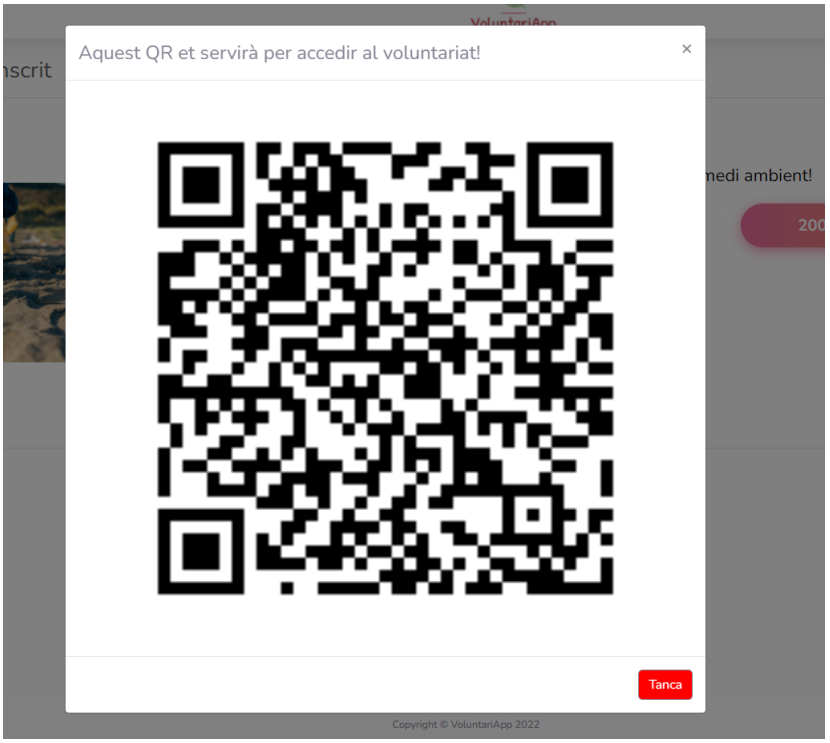


Figura 93: QR voluntariat.
Font: VoluntariApp.

8.1.4 Experiències

Si accedim a l'apartat "Experiències" podrem veure totes les Experiències futures, i filtrar per nom i localització.

Figura 94: Experiències general.
Font: VoluntariApp.

Igual que amb els Voluntariats, també podrem cercar Experiències al mapa.

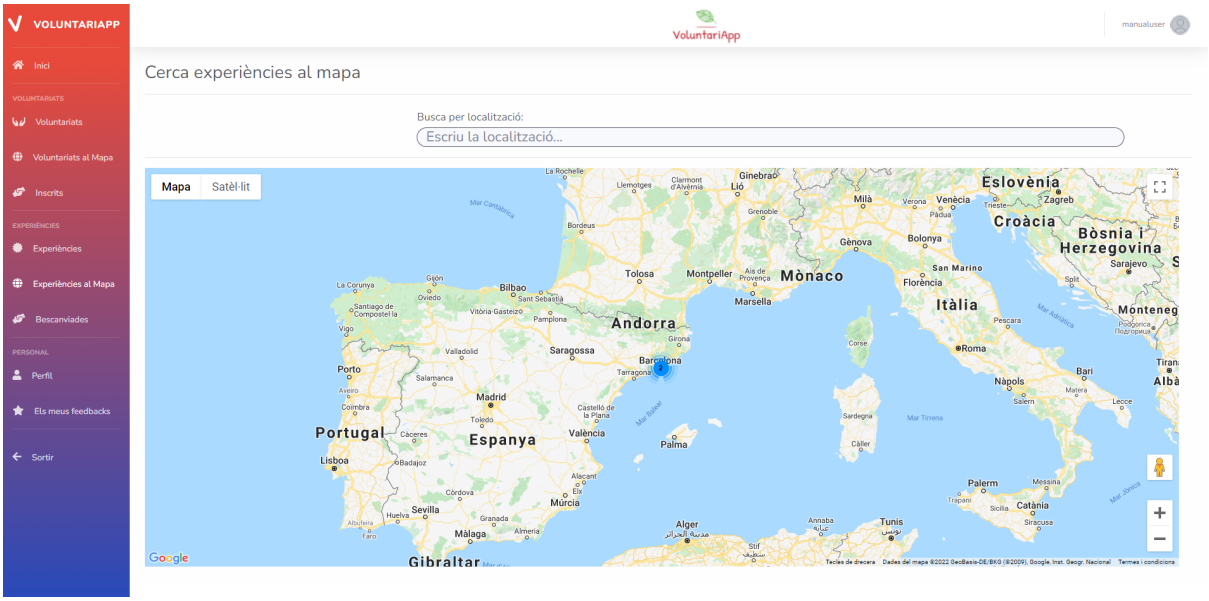


Figura 95: Experiències al Mapa.
Font: VoluntariApp.

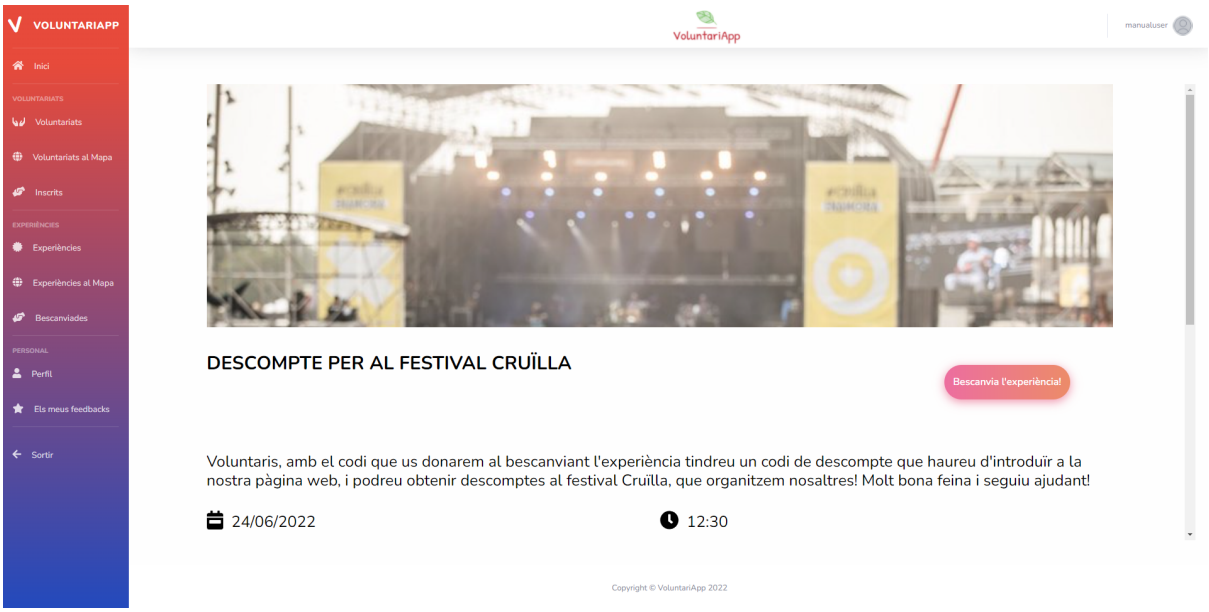


Figura 96: Experiència Específica 1.
Font: VoluntariApp.

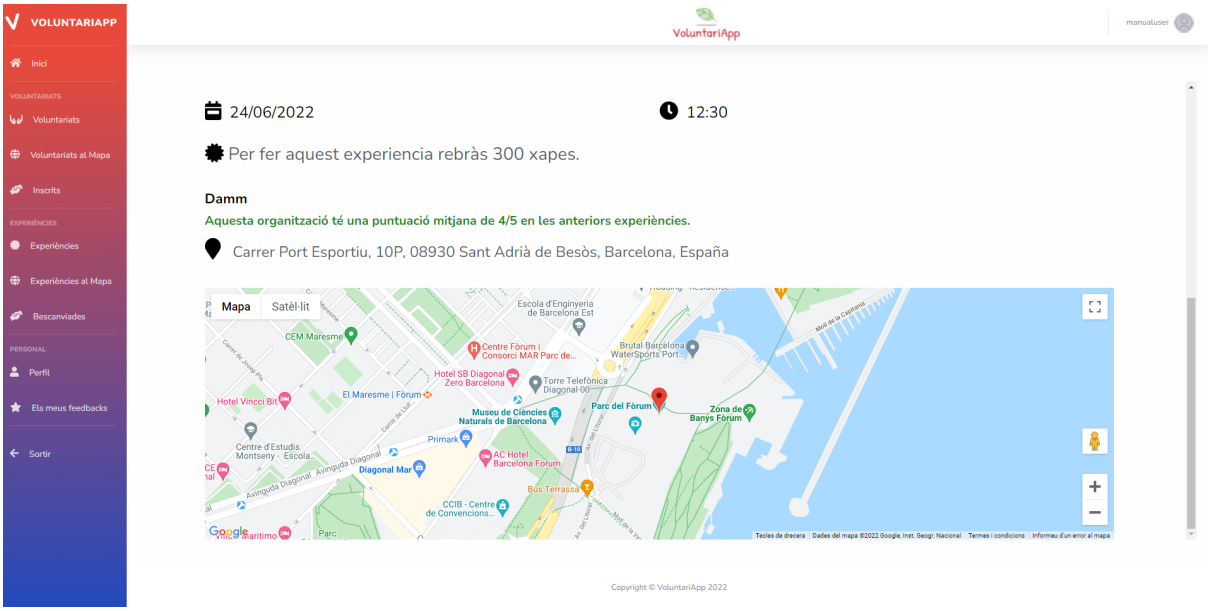


Figura 97: Experiència Específica 1.
Font: VoluntariApp.

Si cliquem el botó “Bescanviar Experiència” podrem bescanviar l’experiència per xapes, per poder obtenir la recompensa.

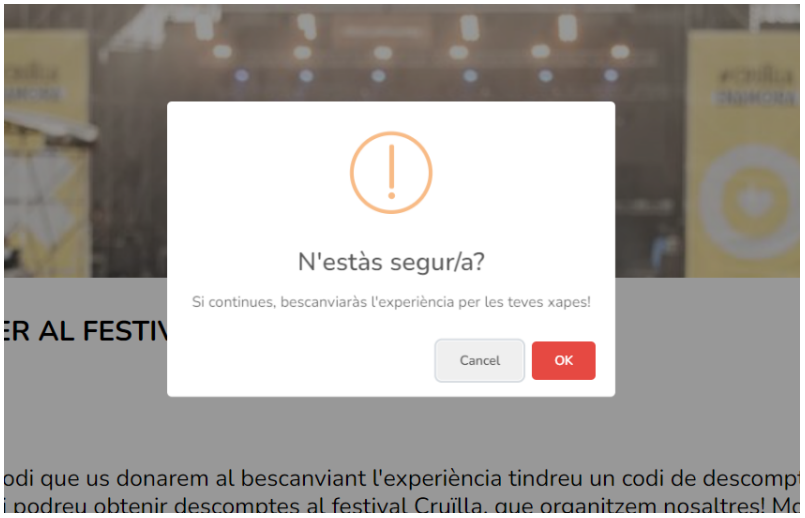


Figura 98: Bescanviar experiència pas 1.
Font: VoluntariApp.

Si no tenim suficients xapes, no podrem bescanviar-la.

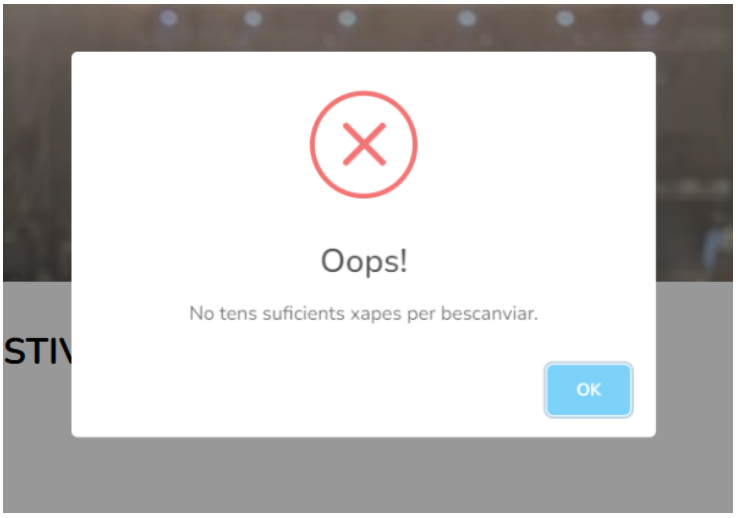


Figura 99: Bescanviar experiència pas 2 incorrecte.
Font: VoluntariApp.

Ens apuntarem a l'experiència creada a l'apartat 8.2.3.

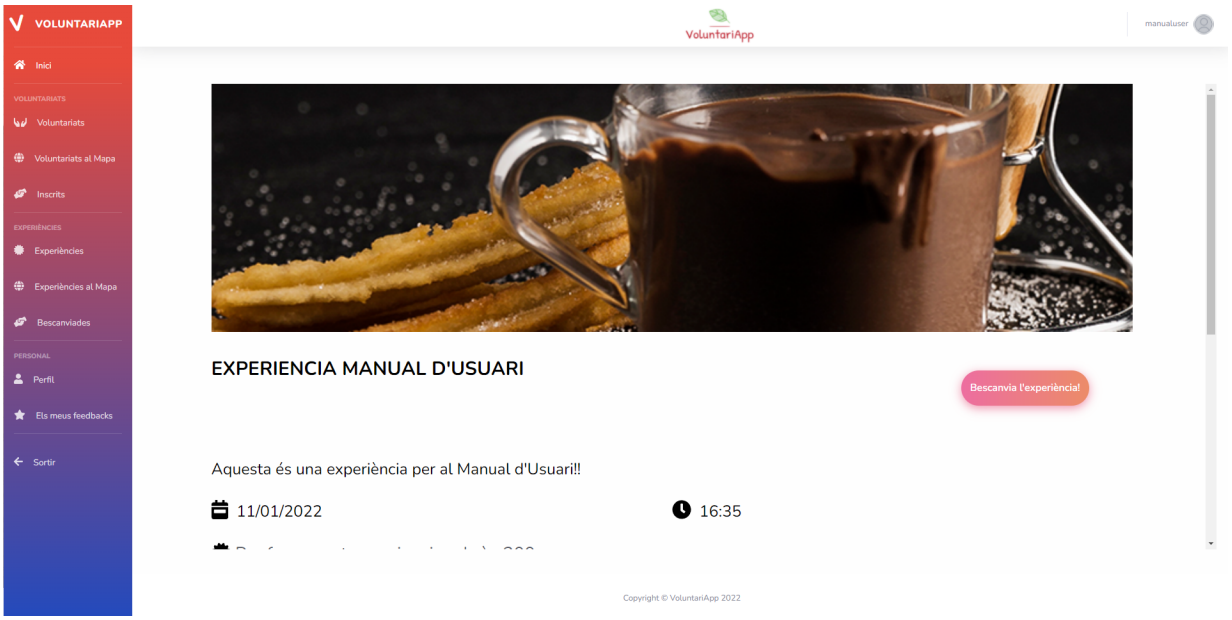


Figura 100: Experiència creada al 8.2.3.
Font: VoluntariApp.

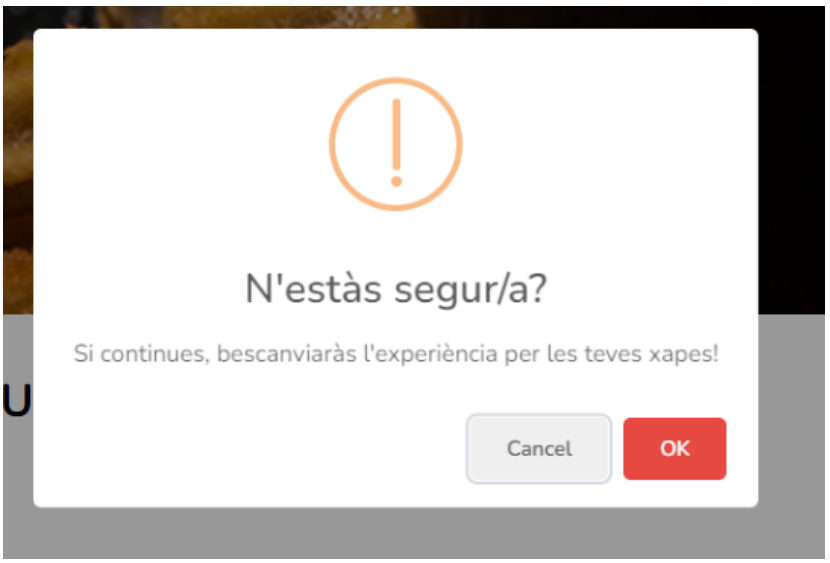


Figura 101: Bescanviar experiència pas 1.
Font: VoluntariApp.

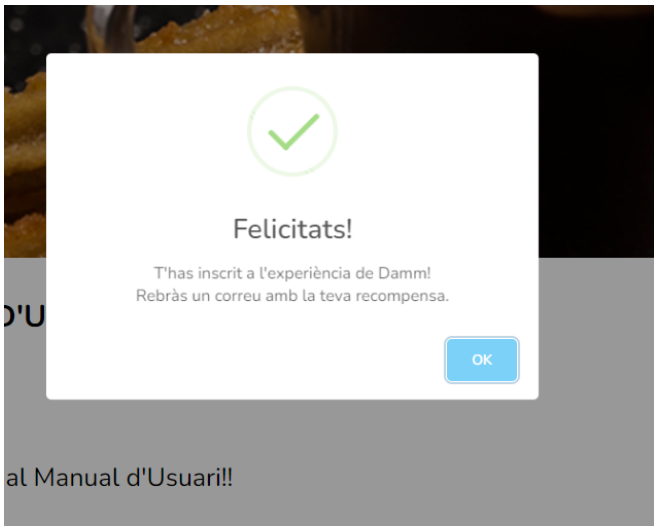


Figura 102: Bescanviar experiència pas 2 correcte.
Font: VoluntariApp.

A continuació rebrem un correu en el que se'ns indicarà el codi a bescanviar per obtenir l'experiència.

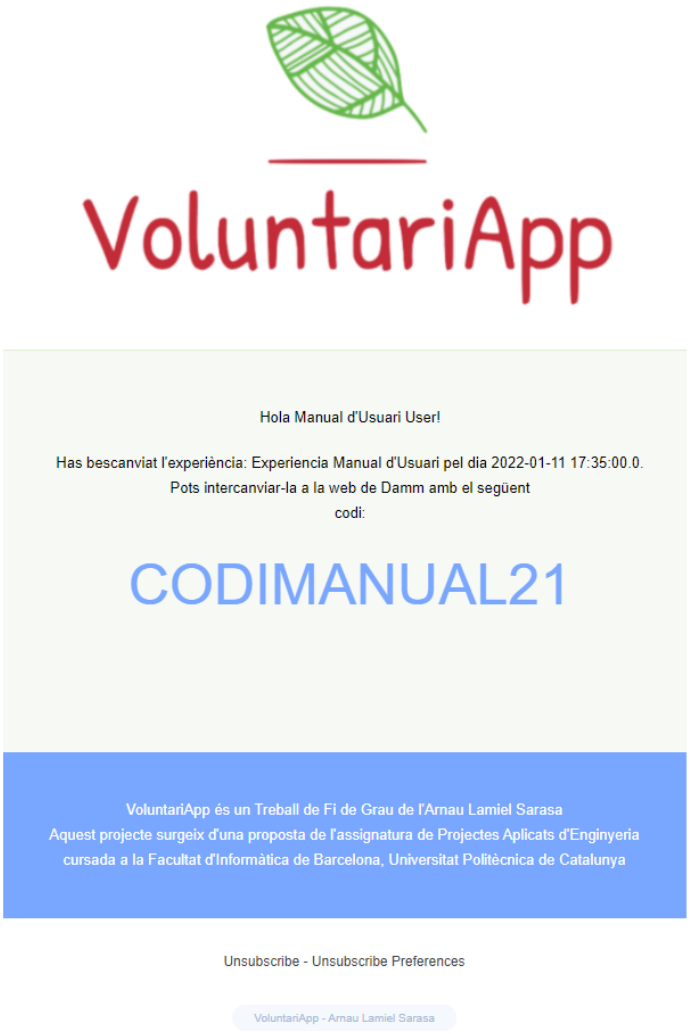


Figura 103: Correu que rep el voluntari al bescanviar experiència.
Font: VoluntariApp.

Si observem el perfil a l'apartat "Perfil", podrem veure tota la nostra informació d'usuari, veure tots els voluntariats i experiències fetes i editar-lo.

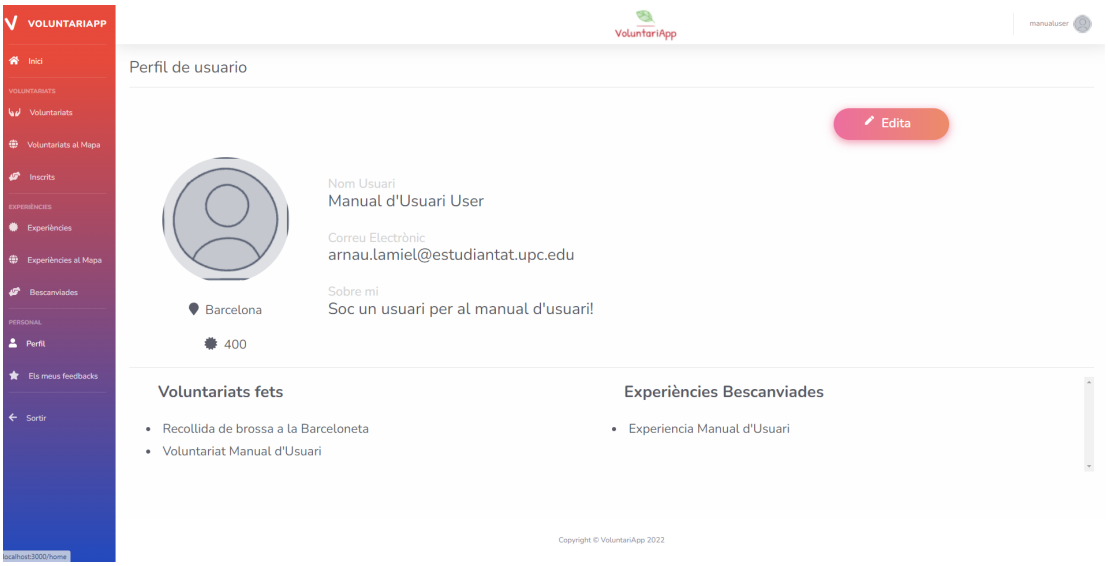


Figura 104: Perfil.
Font: VoluntariApp.

8.1.5 Feedback

A l'apartat d'inscrits, si hem confirmat assistència a un voluntariat tindrem l'opció de puntuar-lo.

Figura 105: Voluntariat amb assistència confirmada.
Font: VoluntariApp.

Haurem d'inidicar la nota del voluntariat clicant a les cares (de l'u al cinc anant d'esquerra a dreta), i posteriorment, indicant-hi els comentaris addicionals que volem que l'organització rebi.

Figura 106: Feedback.
Font: VoluntariApp.

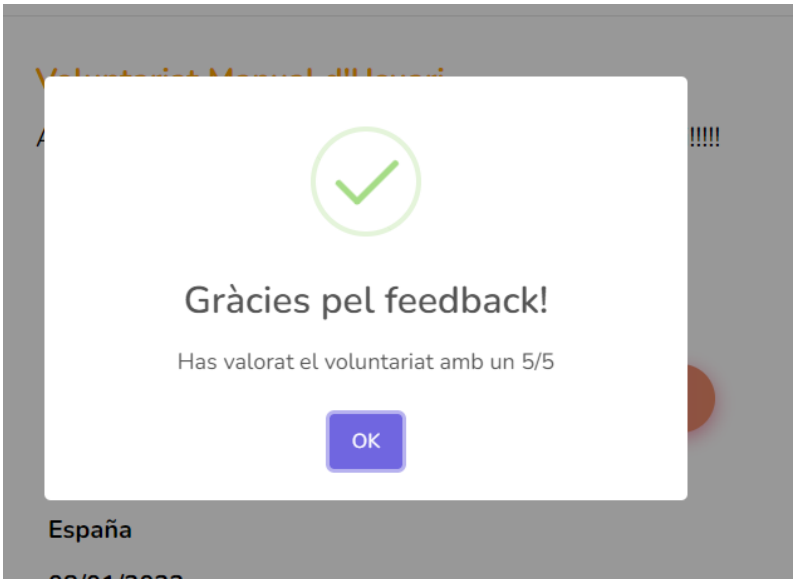


Figura 107: Feedback correcte.
Font: VoluntariApp.

I finalment, a l'apartat de "Els meus feedbacks" podrem obtenir tots els feedbacks que hem fet, si son a voluntariats o a experiències, i l'accés a aquests clicant-hi en ells.

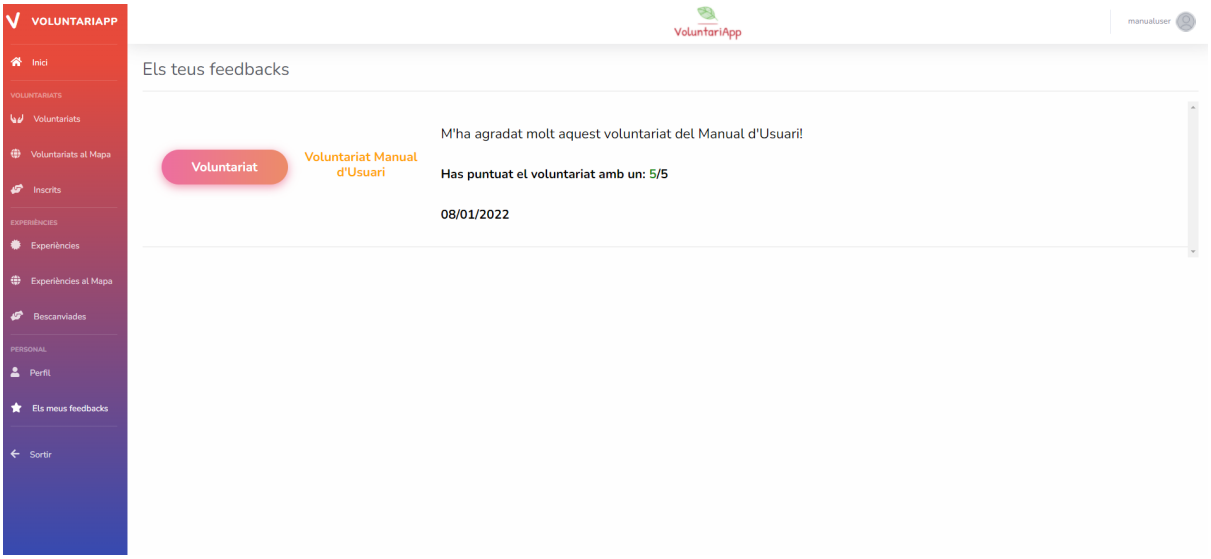


Figura 108: Els meus feedbacks.
Font: VoluntariApp.

A més, també tenim accés a les experiències que hem bescanviat a l'apartat "Experiències Bescanviades", i allà les podrem puntuar mitjançant un feedback igual que amb els voluntariats

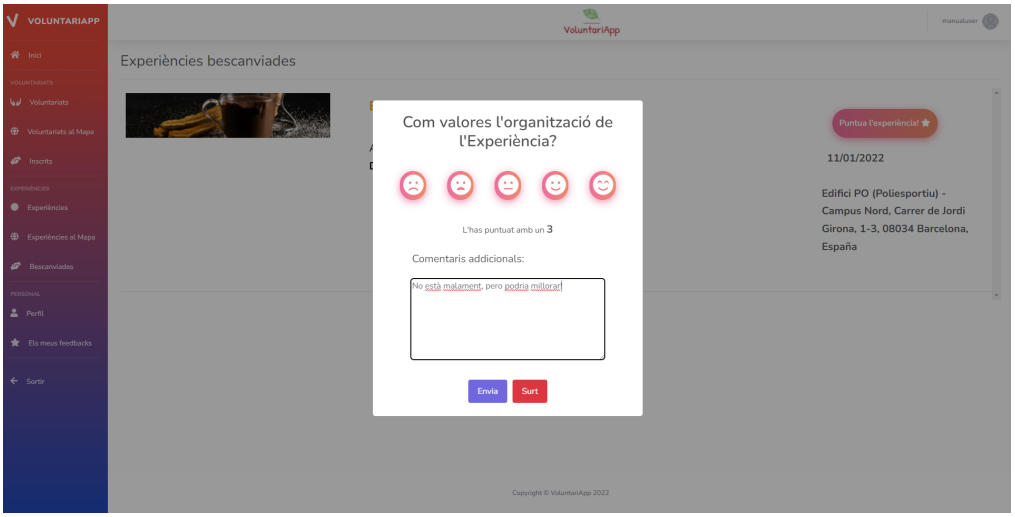


Figura 109: Feedback Experiència.
Font: VoluntariApp.

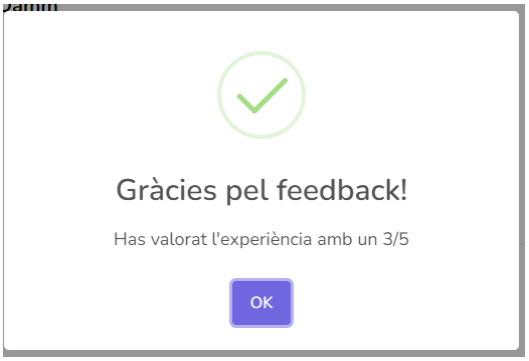


Figura 110: Feedback Experiència confirmació.
Font: VoluntariApp.

I veiem que “Els meus feedbacks” s’ha actualitzat.

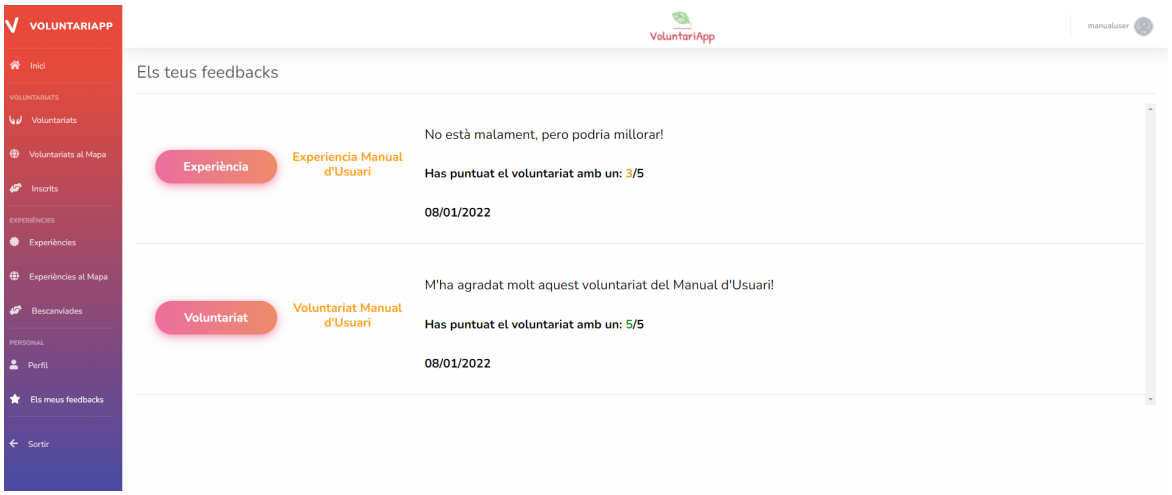


Figura 111: Els meus feedbacks.
Font: VoluntariApp.

8.2 Organització

8.2.1 Inici Sessió

Igual que en l'apartat anterior, podrem accedir amb les credencials de la nostra organització, que se'ns hauran otorgat amb anterioritat.

Figura 112: Inici Sessió Organització.
Font: VoluntariApp.

Com a Organització, tenim més funcionalitats a la barra lateral esquerra que com a voluntari.
Començarem per l'apartat de Publicacions.

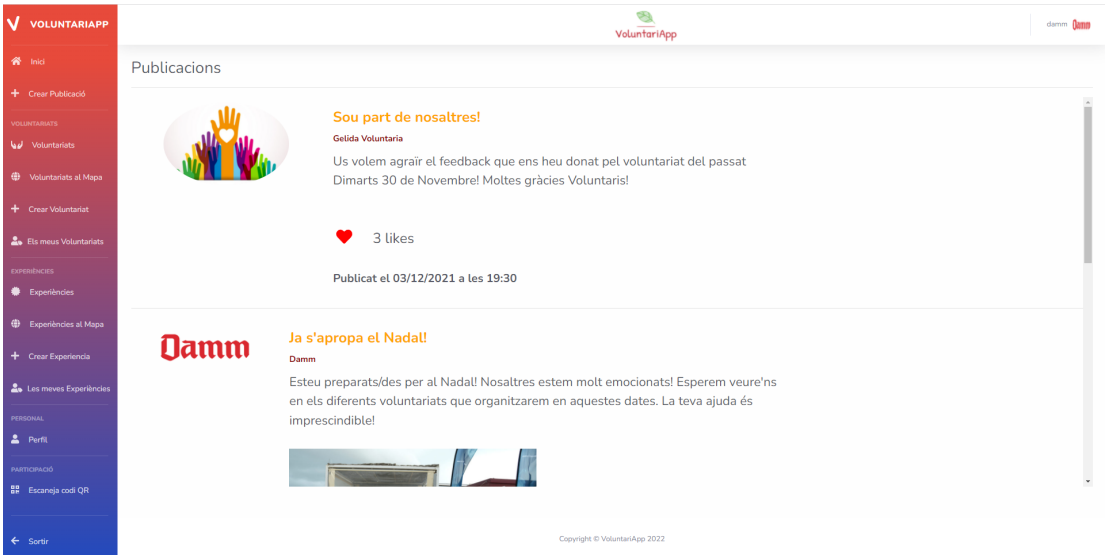


Figura 113: Publicacions.
Font: VoluntariApp.

8.2.2 Publicacions

Com a Organització, podem crear una publicació clicant al botó lateral “Crear Publicació”. Podem indicar-ne el títol i el cos de la publicació, i també una imatge que ho acompanyi.

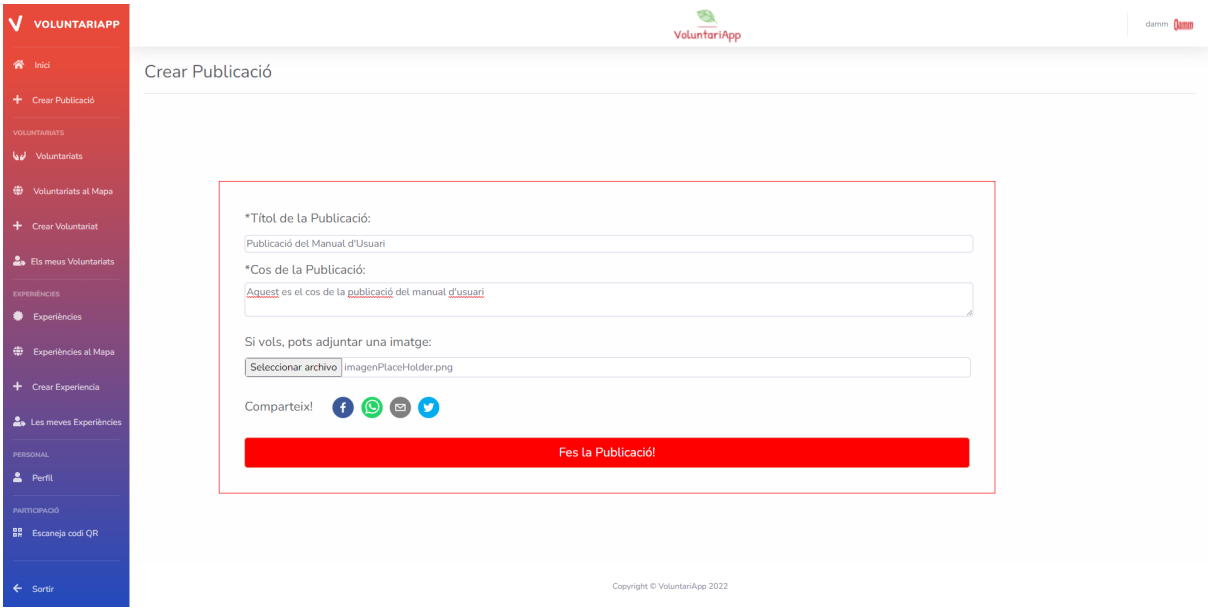


Figura 114: Crear Publicació
Font: VoluntariApp.

Si volem, a més, podrem compartir la publicació a Facebook, Twitter, Correu electrònic, o Whatsapp.

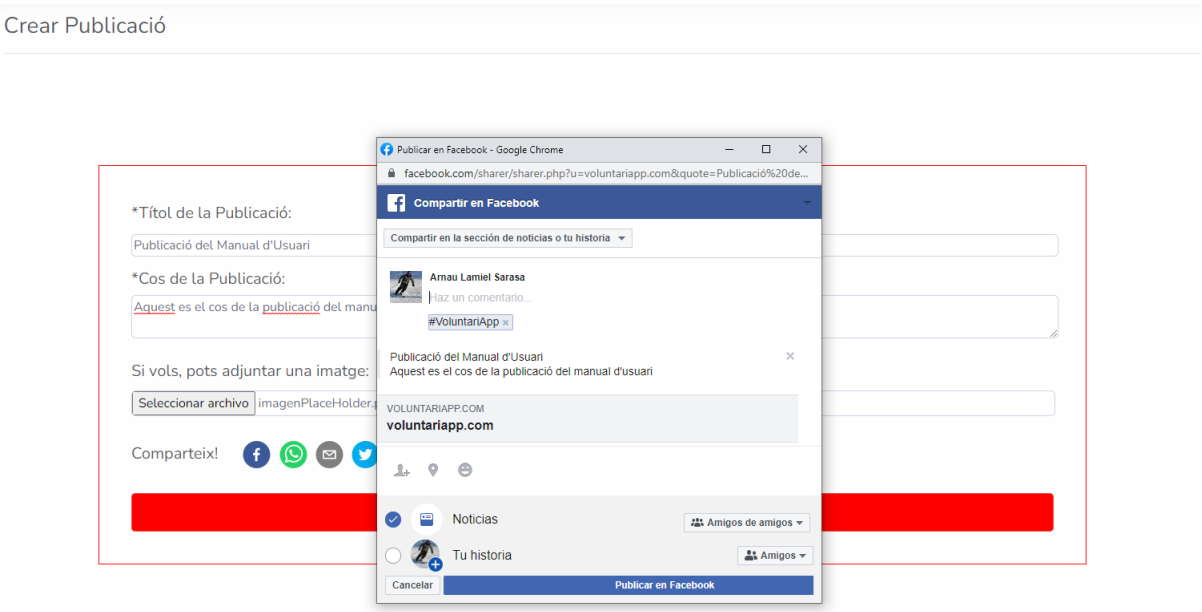
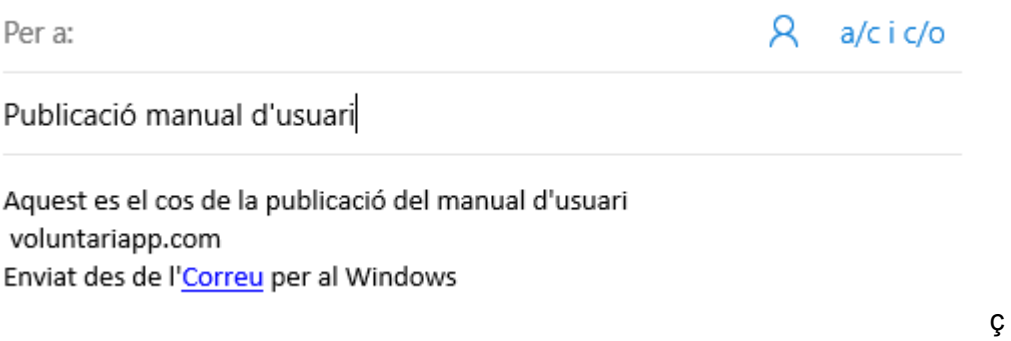


Figura 115: Publicació compartida a Facebook.
Font: VoluntariApp.

Figura 116: Publicació compartida a Twitter.
Font: VoluntariApp.



Ç

Figura 117: Publicació compartida per correu.
Font: VoluntariApp.

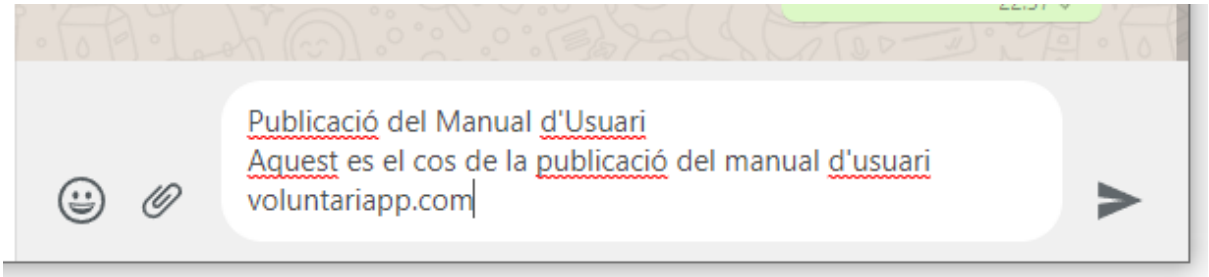


Figura 118: Publicació compartida per WhatsApp.
Font: VoluntariApp.

I si la publiquem a *VoluntariApp* la podrem veure a l'inici.

Figura 119: Publicacions amb la nova Publicació.
Font: VoluntariApp.

8.2.3 Voluntariats

Com a Organització, podrem veure els nostres voluntariats a l'apartat "Els meus Voluntariats". I en cadascun dels que sigui posterior al moment en que es miri, podrem veure tots els feedbacks rebuts per part dels voluntaris en cada voluntariat.

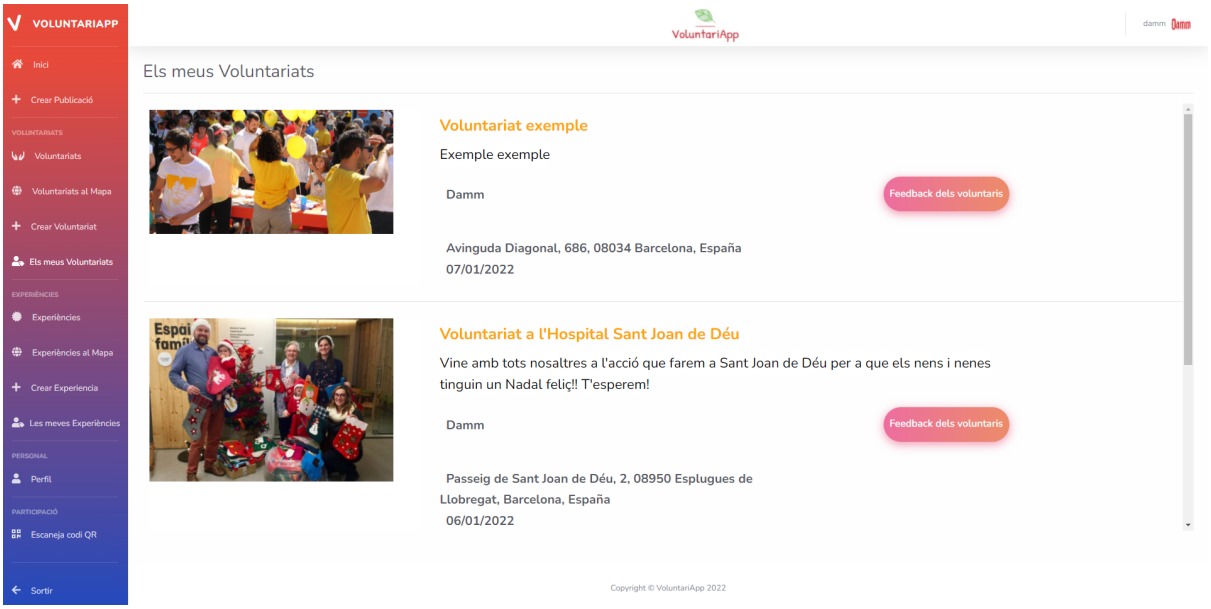


Figura 120: Els meus voluntariats.
Font: VoluntariApp.

Figura 121: Feedbacks rebuts d'un voluntariat.
Font: VoluntariApp.

Podem crear un voluntariat anant a l'apartat "Crear Voluntariat" de la barra lateral omplint les dades necessàries, i si volem, una foto..

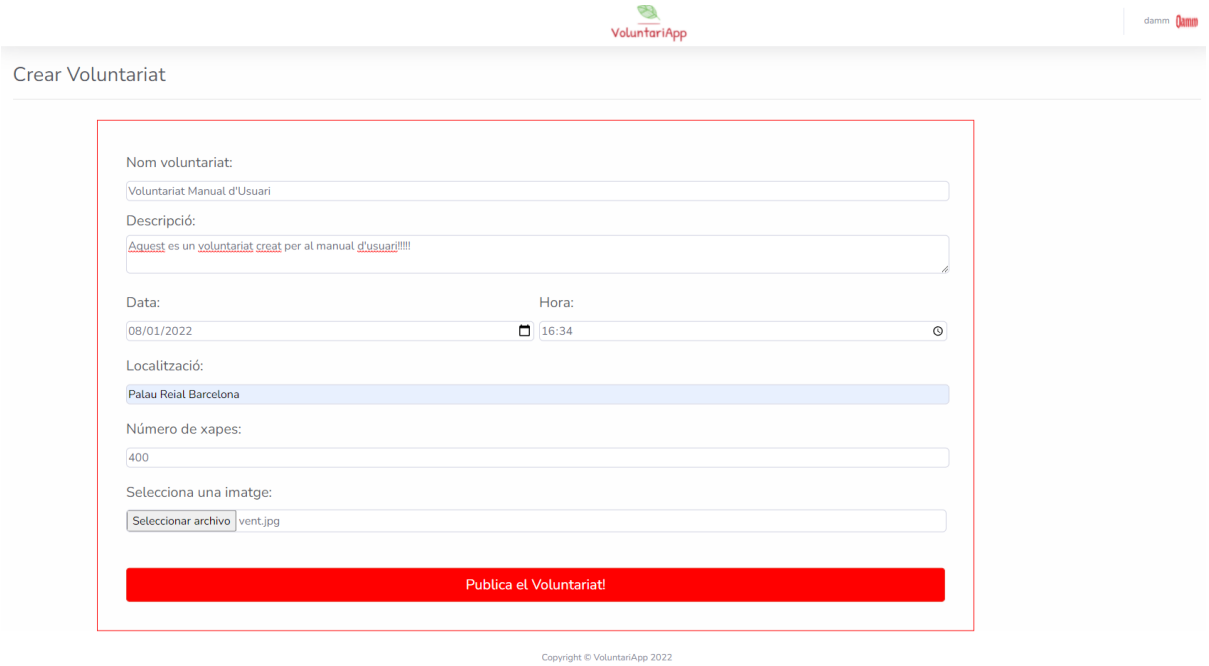


Figura 122: Crear voluntariat.
Font: VoluntariApp.

Com aquest voluntariat l'hem creat a data d'avui, podrem enviar notificacions en temps real als voluntaris que hagin confirmat asistència al voluntariat.

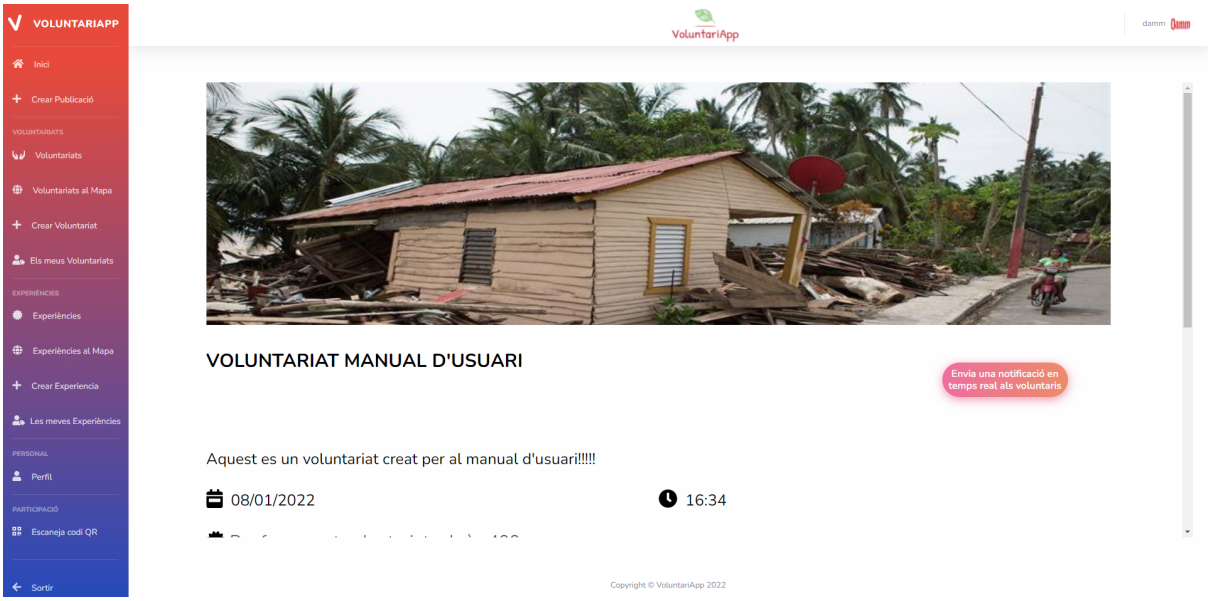


Figura 123: Voluntari Creat.
Font: VoluntariApp.

Abans d'això, veurem com podem confirmar assistència d'un voluntari a un voluntariat.
A partir de l'apartat "Escaneja codi QR" podrem escanejar el QR de cada voluntari per aixi confirmar-la, i que se li otorguin les xapes al voluntari.

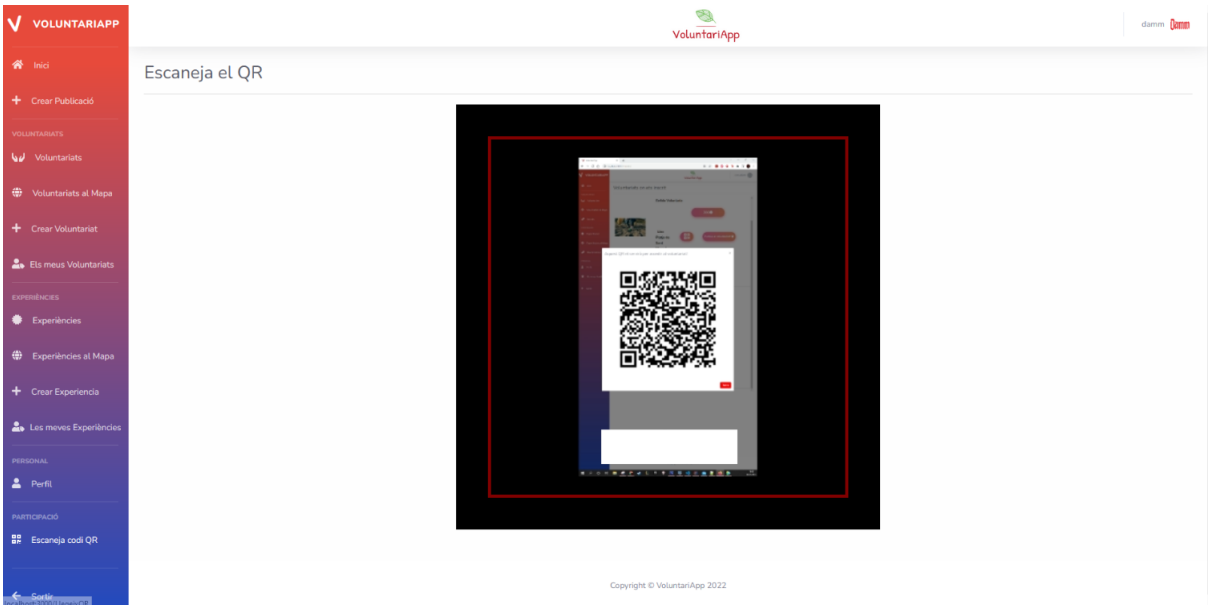


Figura 124: Escanejar QR.
Font: VoluntariApp.

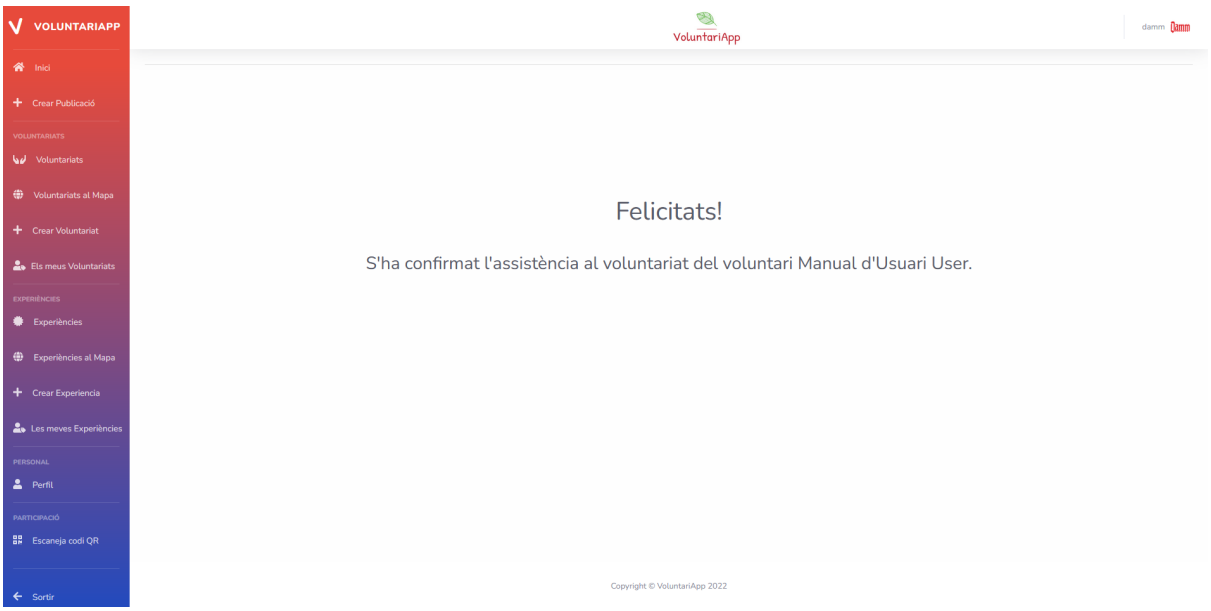


Figura 125: Assistència confirmada.
Font: VoluntariApp.

8.2.4 Experiències

També podem consultar les nostres experiències a l'apartat "Les meves Experiències", i de la mateixa manera que amb els voluntariats, si són passats podrem veure-hi els feedbacks.

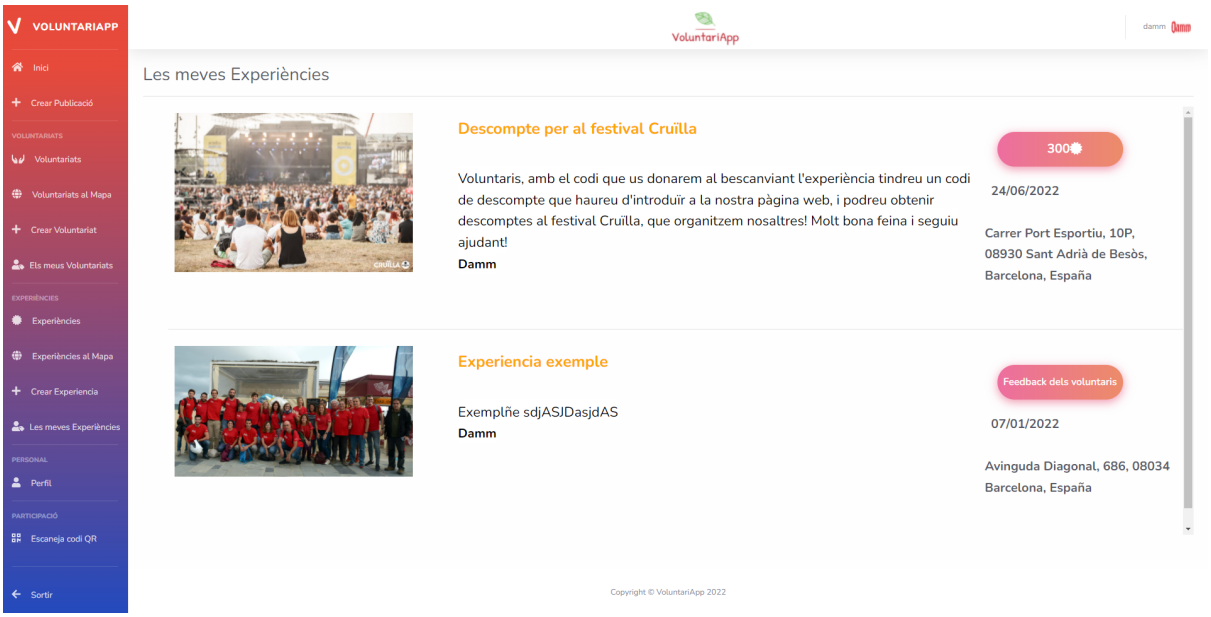


Figura 126: Les meves experiències.
Font: VoluntariApp.

Igual que amb els voluntariats, podem crear experiències desde l'apartat "Crear Experiència" si omplim tota la informació requerida.

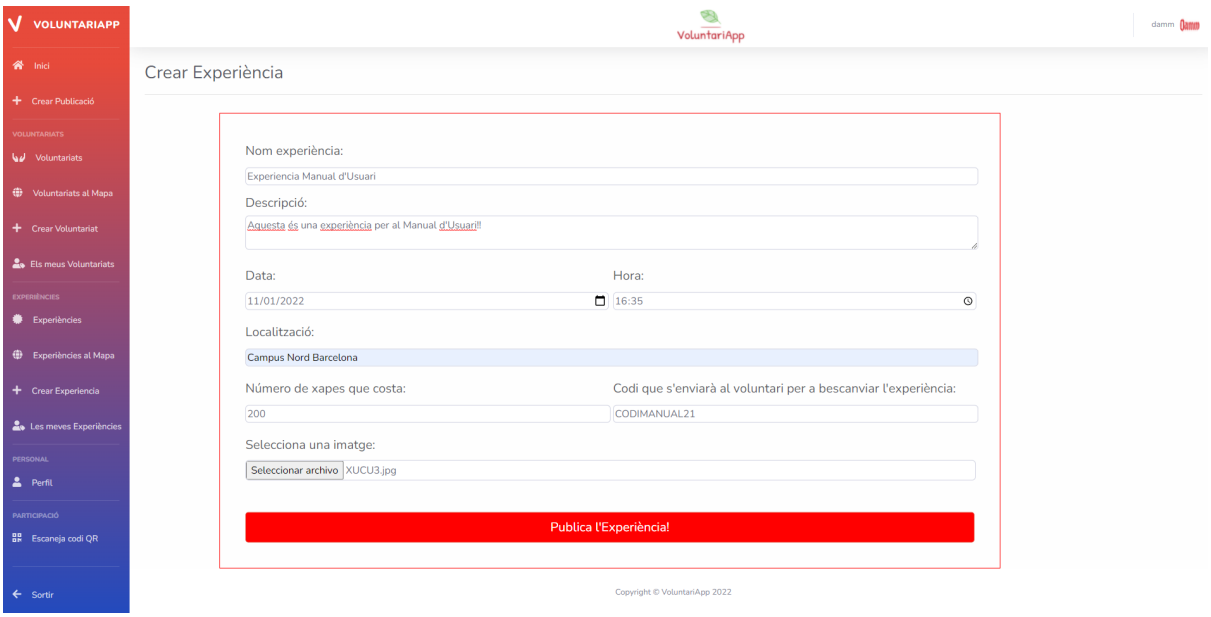


Figura 127: Crear Experiència.
Font: VoluntariApp.

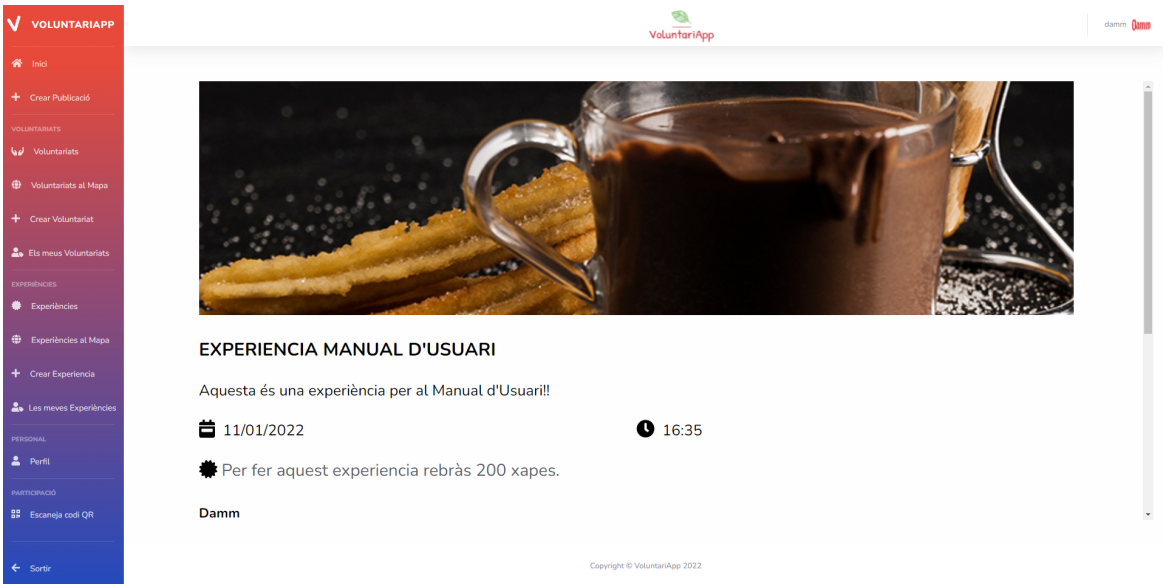


Figura 128: Experiència creada.
Font: VoluntariApp.

8.2.5 Notificacions

Finalment, si entrem al voluntariat creat en l'apartat 8.2.3 podrem veure el botó "Envia notificació en temps real". Si hi cliquem, podrem enviar una notificació a tots els voluntariats que hagin confirmat asistència al voluntariat.

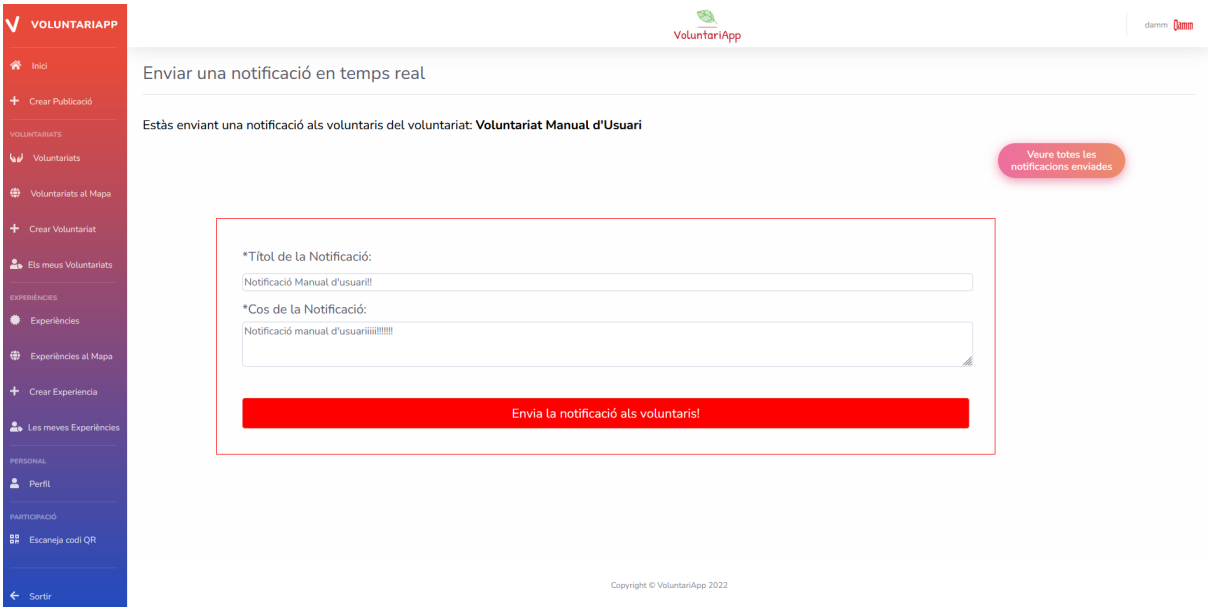


Figura 129: Envia notificació en temps real.
Font: VoluntariApp.

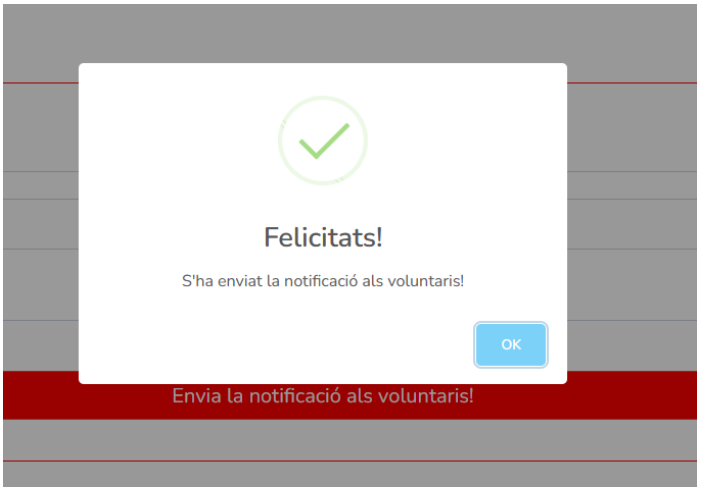


Figura 130: Envia notificació en temps real correcte.
Font: VoluntariApp.

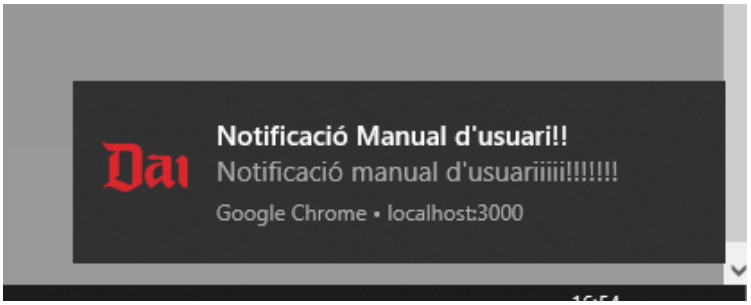


Figura 131: Visió d'un voluntariat que ha confirmat l'assistència al voluntariat on s'ha enviat la notifiació en temps real. Utilitzant Google Chrome.
Font: VoluntariApp.

Si cliquem al botó de “Veure totes les notificacions enviades” veurem una llista de totes les notificacions del voluntariat que haguem enviat.

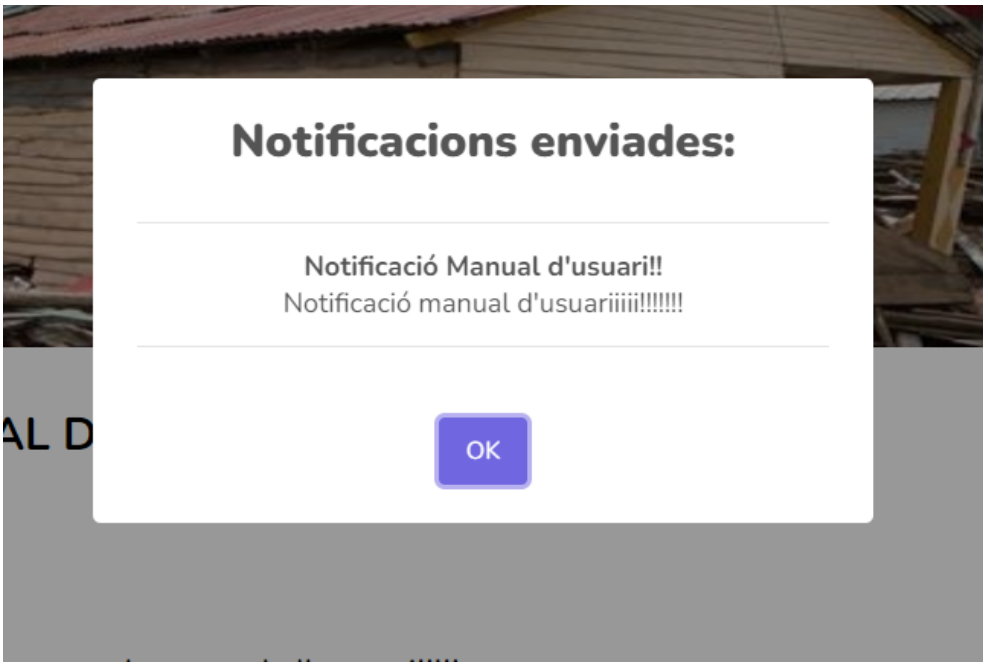


Figura 132: Notificacions enviades.
Font: VoluntariApp.

9. Metodologia i seguiment

9.1 Metodologia de treball

Aquest projecte seguirà una metodologia àgil o *Agile*[23], concretament Scrum[24]

A grans trets, Scrum és una metodologia iterativa i incremental que, a partir d'un *product backlog* [25], que és la llista de tasques pendents obtingudes de les històries d'usuari, crea *Sprints*[26], que són períodes de temps delimitat pels participants on s'hi inclouen certes tasques del *backlog* per a acabar-les.
Cada tasca té el seu pes, que indica la dificultat, i la intenció és crear *Sprints* amb pesos semblants per a repartir el treball equitativament.
Aquesta metodologia divideix l'equip en diferents rols, però com només hi ha un desenvolupador en aquest treball, modificarem una mica la metodologia.

Com a petita modificació de l'Scrum tradicional, en aquest projecte actuaré, a part de desenvolupador, com a Proxy Product Owner[27], ja que em dedicaré a recollir les necessitats dels clients, estructurar i realitzar el backlog i validar el producte a cada iteració.
Això s'ha fet d'aquesta manera perquè el contacte amb la DAMM és més aviat funcional i no té coneixements d'Agile.

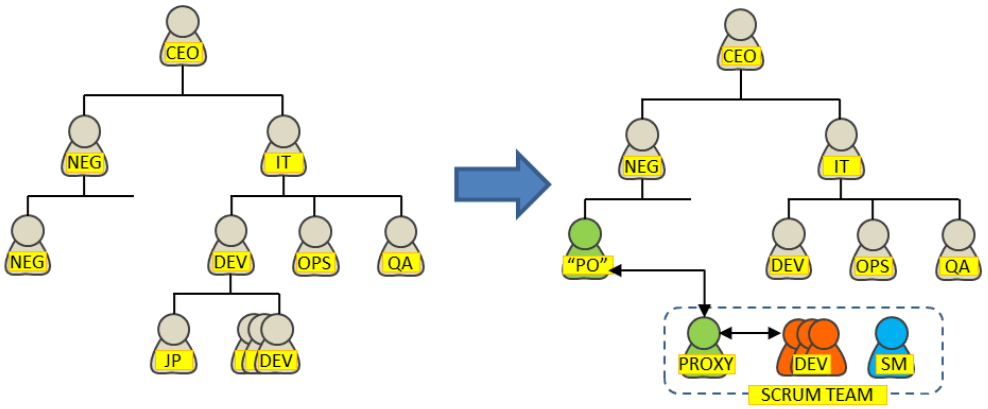


Figura 133: La figura del Proxy Product Owner.
Font[27]

L'elecció de la metodologia s'ha fet tenint en compte diferents factors[28]:

1. Scrum és una metodologia que usa iteracions i és incremental, per tant tenim un producte desenvolupat en cada iteració (corresponent a certes Històries d'Usuari) que podrem ensenyar la DAMM quan ens anem reunint amb ells periòdicament.

2. Al ser incremental, si durant el desenvolupament veiem que ens hem oblidat quelcom, o se'ns acut una idea d'una funcionalitat nova que pot ser molt favorable al projecte, es pot afegir en els següents sprints sense problema. A més a més, els canvis són molt més fàcils d'aplicar.

3. Al ser iterativa i incremental, si no arribem a la data establerta amb tota la plataforma, sempre podrem tenir més funcionalitats desenvolupades per a fer una versió bàsica de la plataforma.

9.2 Eines de seguiment

Per a poder desenvolupar en Scrum, necessitem unes eines indispensables per a fer el seguiment i el desenvolupament continu que es requereix cada *Sprint*.

9.2.1. Gestió del Backlog - Taiga

Per a poder portar un control de les Històries d'Usuari, les tasques del backlog, les èpiques[29], els pesos, i els Sprints, és necessària una aplicació.

Taiga[30] ens permet crear èpiques d'on treure històries d'usuari amb el seu pes concret, que alhora es poden dividir en tasques més específiques per a poder generar els diferents Sprints. A cada *sprint* les tasques es poden posar en un *Scrum Board*[31] que indica quines tasques estan per fer, quines estan fent-se, quines estan per provar, i quines estan acabades.
Finalment, també et deixa gestionar els *issues* o problemes que puguin sortir.

9.2.2 Control de Versions - Git

Per a poder gestionar les versions del codi de l'aplicació, hem d'usar un control de versions com Git[32].
Aquest control de versions és el més utilitzat arreu del món, per la seva facilitat i per què et permet portar un control exhaustiu dels canvis en el codi i tornar a una versió anterior en cas que falli quelcom.
Per això, Git crea repositoris de codi, els quals permeten la creació de branques per a estructurar el codi i que hi hagi el mínim conflicte possible entre desenvolupadors.

En el nostre cas, utilitzarem la branca *master* per a les funcionalitats acabades i testejades, i la branca *development* per totes les que encara no ho estan.
Aquesta branca *development* tindrà una subbranca per cada funcionalitat, perquè tot estigui més ben estructurat, i així podrem anar desplegant a la branca master tot el que funciona sense cap inconvenient.

Concretament, el nostre projecte està al gestor de repositoris de *GitHub*[33].

9.3 Mètodes de Validació

Per a assegurar-nos que el nostre projecte va pel bon camí, i que el que s'està desenvolupant és el que es vol desenvolupar, calen mètodes de validació.

En aquest projecte hem pensat en diferents mètodes, que enunciarem a continuació:

- **Reunions de seguiment amb el director**: Es faran unes reunions de seguiment amb el director del TFG per a poder anar avançant si tenim dubtes, problemes, peticions a la universitat... per a poder superar els possibles *stoppers*, és a dir, qualsevol cosa que ens eviti avançar.

- **Reunions de validació inherents a *Scrum:*** Farem reunions dins de la metodologia *Scrum* (com sprint plannings, sprint retrospective…) per a validar tasques i requeriments d'usuari.

- **Tests:** Realitzarem *tests* en el desenvolupament per a assegurar-nos i validar les funcionalitats que implementem.

10. Planificació temporal i Estimació del projecte

Aquest projecte va començar a mitjans de juliol de 2021, amb una aturada a l'Agost. I té la intenció d'acabar-se, aproximadament, el 17 de gener de 2022, ja que la lectura del TFG es fa la setmana del 24 de gener de 2022, i hauriem de tenir-lo entregat una setmana abans per a que el tribunal el pugui llegir.

En quant a l'organització del projecte, s'han dedicat 15 hores de feina a la presa de requisits, sumant les reunions amb la DAMM, amb el director, i treball personal (del dia 16 de juliol al 3 d'Agost).
Per altra banda, en lo referent a la gestió del projecte (que inclou definir l'abast, context, planificació temporal, anàlisi de riscos, costos, pressupostos i informes de sostenibilitat) , s'hi dedicaran del 7 de setembre fins que acabi Gestió de Projectes (GEP), aproximadament el 19 d'octubre. Si per cada lliurament s'hi dediquen 25 hores, s'hi dedicaran un total aproximat de 70 hores.
Finalment, a partir del 19 d'octubre començarà el desenvolupament del projecte, seguint la metodologia *Scrum*, i acabarà aproximadament el 10 de gener de 2022. Si hi dediquem de 35 a 40 hores setmanals, acabarem fent aproximadament de 420 a 480 hores de desenvolupament.
Finalment, del 10 de gener de 2022 al 17 de gener de 2022 s'hi dedicaran unes 40 hores a acabar d'escriure la documentació i acabar la memòria.

Independentment del que acabem d'estimar, cada setmana hi haurà una reunió periòdica amb el director del TFG d'una hora, per a comentar avenços, dubtes, i altres preocupacions per a que el treball avanci de manera satisfactòria, fent un total de 25 hores.
A més, si es necessita a l'inici d'algun *sprint* la participació de DAMM, també tindrem una reunió amb ells per cada sprint. El que suposa una o dues hores per sprint, si fem 4 sprints, resulta en un màxim de 8 hores de reunions amb DAMM.
Per tant, hi dedicarem un total aproximat de 580 hores.

Per tal de fer-nos una idea més clara de l'estimació del projecte, farem una taula:

Etapa	Data d'inici	Data de finalització
Presa de Requisits	16 Juliol 2021	7 Setembre 2021
Gestió del projecte	7 Setembre 2021	19 Octubre 2021
Sprint 1	20 Octubre 2021	7 Novembre 2021
Sprint 2	8 Novembre 2021	28 Novembre 2021
Sprint 3	29 Novembre 2021	19 Desembre 2021
Sprint 4	20 Desembre 2021	9 Gener 2022
Documentació final	10 Gener 2022	17 Gener 2022

Figura 134: Taula d'Estimació del Projecte. Font: Elaboració Pròpia

10.1 Definició de les tasques

Com que estem utilitzant una metodologia àgil com *Scrum*, funcionarem en *Sprints.*
En cada *sprint* tindrem una sèrie d'històries d'usuari que haurem de desglossar en tasques. Començarem definint aquestes tasques amb un títol, descripció, i una duració aproximada per tasca, per tal de fer-nos una idea sobre el que ens ocuparà.
La nomenclatura que usarem serà GPX per a les tasques de Gestió de Projectes i DX a les de desenvolupament.

10.1.1 Tasques de gestió del projecte

- **GP1 - Presa de Requisits (7h):** Reunions amb DAMM i treball individual per a la presa de requisits del projecte.
  Dependencies: Cap.
  Recursos:
  - *Materials:* Ordinador amb internet, *Google Meet*.
  - *Humans:* Cap de projecte (CP) , PPO, director TFG (dTFG), contacte DAMM (cDamm).
- **GP2 - Definició de les històries d'usuari (10h):** Després de la presa de requisits, elaborar les històries d'usuari corresponents a les funcionalitats de la plataforma.
  Dependencies: GP1.
  Recursos:
  - *Materials:* Ordinador amb internet, *Google Drive*.
  - *Humans:* PPO, cap de projecte, director TFG.
- **GP3 - Contextualització i abast del projecte (25h):** Redacció i definició del context i l'abast del projecte, la justificació i la metodologia duta a terme.
  Dependencies: Cap.
  Recursos:
  - *Materials:* Ordinador amb internet, Google Drive.
  - *Humans:* Cap de projecte, director TFG.
- **GP4 - Planificació temporal (20h):** Redacció i planificació temporal del projecte. Això inclou la planificació de les tasques, diagrama de Gantt, i la gestió dels obstacles i riscos del projecte.
  Dependencies: GP3.
  Recursos:
  - *Materials:* Ordinador amb internet, *Google Drive, Taiga, Gantt Project*.
  - *Humans:* Cap de projecte, director TFG.
- **GP5 - Pressupost i sostenibilitat (20h):** Redacció i càlcul del pressupost, i estudi de la sostenibilitat del projecte.
  Dependencies: GP4.
  Recursos:
  - *Materials:* Ordinador amb internet, *Google Drive*.
  - *Humans:* Cap de projecte, director TFG.
- **GP6 - Recopilació de GEP (15h):** Agrupar els documents anteriors tenint en compte el feedback dels professors de GEP per a tenir una definició exhaustiva del projecte.
  Dependencies: GP5.
  Recursos:

    - *Materials:* Ordinador amb internet, *Google Drive*.
    - *Humans:* Cap de projecte.
- **GP7 - Reunions amb el director TFG (1h):** Les reunions amb el director per al desenvolupament del projecte, dubtes i temes relacionats.
  Dependencies: Cap.
  Recursos:
    - *Materials:* Ordinador amb internet, *Google Meet*.
    - *Humans:* Cap de projecte / desenvolupador(D), director TFG.
- **GP8 - Retrospective Meeting (3h):** Reunió que es fa al final de cada *sprint* per evaluar el que s'ha fet, com s'ha fet, i que es podria millorar.
  Dependencies: Totes les tasques de l'*sprint* actual.
  Recursos:
    - *Materials:* Ordinador amb internet, *Google Meet*.
    - *Humans:* Cap de projecte/desenvolupador, director TFG.
- **GP9 - Documentació *sprint* (3h):** La redacció de la documentació amb la informació de cada sprint, per a posar-ho a la memòria del TFG.
  Dependencies: Totes les tasques de l'*sprint* actual.
  Recursos:
    - *Materials:* Ordinador amb internet, *Google Drive*.
    - *Humans:* Cap de projecte/desenvolupador.
- **GP10 - Documentació final (25h):** La redacció de la documentació amb la informació de cada sprint, per a posar-ho a la memòria del TFG.
  Dependencies: L'última tasca de l'*sprint* 4. T4SX..
  Recursos:
    - *Materials:* Ordinador amb internet, *Google Drive*.
    - *Humans:* Cap de projecte/desenvolupador.
- **GP11 - Sprint Planning (2h):** Reunió que es fa al principi de cada *sprint* per a planejar el proper sprint, i retocar les tasques i estimacions si cal.
  Dependencies: GP8 (menys el primer sprint, que depèn de GP6)
  Recursos:
    - *Materials:* Ordinador amb internet, *Google Meet*.
    - *Humans:* Cap de projecte/desenvolupador, director TFG, PPO.

10.1.2 Tasques de desenvolupament

Per a poder seguir amb més facilitat les tasques, les agrupem per *sprints*, amb nomenclatura SXTX (*Sprint* X Tasca X):

10.1.2.1 Sprint 1

- **DS1T1 - Posar en marxa l'estructura de l'aplicació (25h):** Tot lo necessari per a posar en marxa l'estructura sobre la qual desenvoluparem l'aplicació.
  Dependencies: GP11 del primer *sprint*.
  Recursos:
    - *Materials:* Ordinador amb internet, *IntelliJ IDEA, Visual Studio Code, pgAdmin, Google Chrome*.
    - *Humans:* Desenvolupador.

- **DS1T2 - Creació d'usuaris (40h):** Crear a la base de dades els usuaris i el seu CRUD mitjançant una api al backend.
  Dependencies: DS1T1.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador
- **DS1T3 - Registre d'usuaris (FrontEnd[34]) (25h):** Crear un formulari per al registre que comuniqui amb el backend, també l'inici i el tancar sessió.
  Dependencies: DS1T2.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador
- **DS1T4 - Creació de voluntariats (35h):** Crear a la base de dades els voluntariats i el seu CRUD mitjançant una api al backend.
  Dependencies: DS1T1.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, Visual Studio Code, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.

10.1.2.2 Sprint 2

- **DS2T1 - Gestió voluntariats (FrontEnd) (45h):** Crear un frontend per visualitzar, consultar i apuntar-se a un voluntariat que comuniqui amb el backend .
  Dependencies: GP11 del segon *sprint.*
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS2T2 - Creació d'experiències (35h):** Crear a la base de dades les experiències i el seu CRUD mitjançant una api al backend.
  Dependencies: GP11 del segon *sprint*.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS2T3 - Gestió d'experiències (FrontEnd) (45h):** Crear un frontend per visualitzar, consultar i bescanviar experiències que comuniqui amb el backend .
  Dependencies: DS2T2.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS2T4 - Geolocalització (FrontEnd) (35h):** Crear un frontend per veure els esdeveniments al mapa, i poder buscar per zona.
  Dependencies: GP11 del segon *sprint.*
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.

10.1.2.3 Sprint 3

- **DS3T1 - Crear Feedback d'un voluntariat (10h):** Crear a la base de dades un feedback per a un voluntariat i el seu CRUD mitjançant una api al backend.
  Dependencies: GP11 del tercer *sprint.*
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T2 - Crear Feedback d'una experiència (7h):** Crear a la base de dades un feedback per a una experiència i el seu CRUD mitjançant una api al backend.
  Dependencies: GP11 del tercer *sprint.*
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T3 - Fer Feedback d'un voluntariat (FrontEnd) (15h):** Crear un frontend per a puntuar i fer feedback d'un voluntariat.
  Dependencies: DS3T1.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T4 - Fer Feedback d'una experiència (FrontEnd) (7h):** Crear un frontend per a puntuar i fer feedback d'una experiència.
  Dependencies: DS3T2.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T5 - Crear una publicació (7h):** Crear a la base de dades una publicació i una api al backend per a poder fer-lo.
  Dependencies: GP11 del tercer *sprint*.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA. pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T6 - Crear una publicació (FrontEnd) (15h):** Crear un frontend per a poder fer una publicació i veure les fetes.
  Dependencies: DS3T5.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS3T7 - Crear l'assistència a un voluntariat (15h):** Crear a la base de dades la possibilitat d'assistir a un voluntariat i fer la api per la qual es connecta.
  Dependencies: GP11 del tercer *sprint*.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.

- **DS3T8 - Confirmar asistència a un voluntariat (FrontEnd) (20h):** Crear un frontend per a poder confirmar l'assistència a un voluntariat.

Dependencies: DS3T7.
Recursos:
- *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
- *Humans:* Desenvolupador.

- **DS3T9 - Crear l'assistència a una experiència (15h):** Crear a la base de dades la possibilitat d'assistir a una experiència i fer la api per la qual es connecta.
  Dependencies: GP11 del tercer *sprint*.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.

10.1.2.4 Sprint 4

- **DS4T1 - Confirmar asistència a una experiència (FrontEnd) (20h):** Crear un frontend per a poder confirmar l'assistència a una experiència.
  Dependencies: GP11 del quart *sprint*..
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS4T2 - Crear les notificacions en temps real (50h):** S'han de crear a la base de dades i fer lo necessari al backend perquè es puguin rebre i enviar notificacions a temps real.
  Dependencies: GP11 del quart *sprint*.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS4T3 - Notificacions en temps real (FrontEnd) (35h):** Crear un front end per a poder enviar i rebre notificacions en temps real.
  Dependencies: DS4T2.
  Recursos:
  - *Materials:* Ordinador amb internet, *Visual Studio Code, Google Chrome*.
  - *Humans:* Desenvolupador.
- **DS4T3 - Veure tots els Feedbacks i Notificacions (25h):** Crear un frontend amb unes tots els feedbacks i un altre per a les notificacions.
  Dependencies: DS4T2.
  Recursos:
  - *Materials:* Ordinador amb internet, *IntelliJ IDEA, pgAdmin,Visual Studio Code, Google Chrome*[37].
  - *Humans:* Desenvolupador.

ID	Temps (en h)	Dependències	Rols
GP1	7	-	CP,PPO,dTFG,cDAMM
GP2	10	GP1	CP,PPO,dTFG
GP3	25	-	CP,dTFG
GP4	20	GP3	CP,dTFG
GP5	20	GP4	CP,dTFG
GP6	15	GP5	CP
GP7	1	-	CP/D,dTFG
GP8	3	Totes les tasques de l'*sprint*	CP/D,dTFG
GP9	3	GP8	CP/D
GP10	25	DS4T4	CP/D
GP11	2	GP8 (menys en el primer sprint, que depèn de GP6)	CP/D,PPO,dTFG
DS1T1	25	GP11 del primer sprint	D
DS1T2	40	DS1T1	D
DS1T3	25	DS1T2	D
DS1T4	35	DS1T1	D
DS2T1	45	GP11 del segon sprint.	D
DS2T2	35	GP11 del segon sprint.	D
DS2T3	45	DS2T2	D
DS2T4	35	GP11 del segon sprint.	D
DS3T1	10	GP11 del tercer sprint	D
DS3T2	7	GP11 del tercer sprint	D
DS3T3	15	DS3T1	D
DS3T4	7	DS3T2	D
DS3T5	7	GP11 del tercer sprint	D
DS3T6	15	DS3T5	D
DS3T7	15	GP11 del tercer sprint	D
DS3T8	20	DS3T7	D
DS3T9	15	GP11 del tercer sprint	D
DS4T1	20	GP11 del quart sprint	D
DS4T2	50	GP11 del quart sprint	D
DS4T3	35	DS4T2	D
DS4T4	25	DS4T2	D

Figura 17: Taula de tasques amb hores i dependències. Font: Elaboració Pròpia

10.2 Recursos

Per a poder desenvolupar el projecte, i com hem esmentat en les diferents tasques, hem d'assignar uns recursos humans i materials.

10.2.1 Recursos Humans

Els recursos humans són totes les persones relacionades amb el projecte.
En el nostre cas, tenim:

- **Cap de projecte:** En aquest cas seré jo, és la persona encarregada del projecte i que tot surti bé per totes les àrees. També s'encarrega de la gestió dels requisits i validacions.
- **PPO:** També seré jo. Aquest recurs, com hem explicat en la metodologia, s'encarrega de recollir els requisits i generar el backlog.
- **Director TFG:** Aquest recurs ens guia per al correcte desenvolupament del projecte desde una visió més acadèmica.
- **Contacte de la DAMM:** Aquest recurs s'encarregarà d'expressar els requisits i validar-los conforme es facin.
- **Desenvolupador:** Aquest recurs també seré jo. S'encarrega de desenvolupar totes les funcionalitats de l'aplicació.

10.2.2 Recursos Materials

Els recursos materials són tot aquells recursos (hardware, software…) que ens ajuden a poder desenvolupar el projecte. En aquest cas són els següents:

- **Ordinador amb internet:** Necessari per a poder buscar informació a internet i per a poder desenvolupar el projecte, documentació i les reunions.
- **Google Meet[35]:** Ens serveix per a realitzar les reunions.
- **Google Chrome[36]:** Navegador que fem servir per accedir a la informació i totes les eines en línia.
- **Google Drive[37]:** Editor de text online per a la documentació.
- **Taiga:** Gestor de projectes Àgils. El fem servir per organitzar la nostra feina.
- **IntelliJ IDEA[38]:** IDE per a poder programar la part del backend.
- **Visual Studio Code[39]:** IDE per a poder programar la part del frontend.
- **pgAdmin[40]:** Programa que ens permet veure i gestionar la BBDD.
- **GanttProject[41]:** Ens serveix per a generar el *Gantt* per al nostre projecte.

11. Gestió econòmica i pressupost

En aquest apartat procedirem a analitzar els costos econòmics del projecte.
En primer lloc, identificarem els costos, després els estimarem econòmicament i per acabar, controlarem les desviacions que puguin aparèixer en relació amb el pressupost.

11.1 Identificació i estimació de costos

Per a poder identificar tots els costos del projecte, hem d'analitzar tots els recursos humans i materials que utilitzarem, i alhora, les possibles contingències i imprevistos.

11.1.1 Recursos humans

Per als recursos humans, hem d'identificar tots els costos de personal per cada tasca.
Per començar, hem de definir el salari per cadascun dels rols del projecte. Per fer-ho, ens hem basat en informació sobre els salaris actuals a Barcelona[42][43][44]. Els veurem en la següent taula:

Rol	Cost (€)/h brut
Cap de Projecte	22
Proxy Product Owner (PPO)	19
Desenvolupador	16

Figura 135: Sous de cada rol del projecte.
Font: Elaboració Pròpia

Com podem observar a la taula anterior, hem estimat que el cap del projecte, encarregat de la planificació, el disseny del sistema i la redacció de la documentació, costarà 19 euros bruts l'hora, ja que és un agent amb molta responsabilitat i que requereix un coneixement més elevat.
En segon lloc, hem estimat que el PPO costarà 15 euros bruts l'hora, ja que és el que s'encarrega de la presa de requisits i la definició de tasques del projecte i la seva responsabilitat és més alta que la d'un desenvolupador.
Per acabar, hem assignat un cost de 13 euros bruts l'hora al desenvolupador, que és el que més hores farà.

A continuació, veiem una taula resum dels costos per tasca.

ID	Temps (en h)	PPO (€)	Cap de projecte (€)	Desenvolupador (€)	Total(€)
GP1	7	133	154	0	287
GP2	10	190	220	0	410
GP3	25	475	550	0	1025
GP4	20	0	440	0	440

GP5	20	0	440	0	440
GP6	15	0	330	0	330
GP7	1	0	22	16	38
GP8	12	0	264	192	456
GP9	12	0	264	192	456
GP10	25	0	550	400	950
GP11	8	152	176	128	456
DS1T1	25	0	0	400	400
DS1T2	40	0	0	640	640
DS1T3	25	0	0	400	400
DS1T4	35	0	0	560	560
DS2T1	45	0	0	720	720
DS2T2	35	0	0	560	560
DS2T3	45	0	0	720	720
DS2T4	35	0	0	560	560
DS3T1	10	0	0	160	160
DS3T2	7	0	0	112	112
DS3T3	15	0	0	240	240
DS3T4	7	0	0	112	112
DS3T5	7	0	0	112	112
DS3T6	15	0	0	240	240
DS3T7	15	0	0	240	240
DS3T8	20	0	0	320	320
DS3T9	15	0	0	240	240
DS4T1	20	0	0	320	320
DS4T2	50	0	0	800	800
DS4T3	35	0	0	560	560
DS4T4	25	0	0	400	400

Figura 136: Cost per tasca per rol.
Font: Elaboració Pròpia

Pel que tenim un cost total per recursos humans de 13704 € bruts.

11.1.2 Recursos materials

Utilitzarem un ordinador de sobretaula per a desenvolupar el projecte i per a escriure la documentació de la memòria del TFG.
Com el programari que fem servir per al desenvolupament del projecte és totalment gratuït (editor de text, compilador, frameworks…), no l'afegirem a la taula.

Com que el hardware (hw) té una vida limitada, calcularem el que ens costa l'amortització d'aquest. Aquesta amortització es calcularà de la forma següent:

$$[cost\ de\ l'ordinador(€)\ \div\ (anys\ de\ vida\ hw\ *\ \ dedicació\ a\ la\ setmana(h)\ \ *\ 52\ setmanes\ a\ l'any\ )]\ \ *\ durada(h)$$

Si tenim en compte que el projecte durarà 580 hores, i farem 40 hores a la setmana, tenim que:

Material	Cost ordinador (€)	Anys de vida hw	Amortització (€)
Ordinador sobretaula	2000	5 anys	11.53

Figura 137: Amortització del hardware.
Font: Elaboració Pròpia

11.1.3 Costos indirectes

Per a poder valorar tots els costos indirectes arrelats al projecte d'una manera realista, considerarem que per a desenvolupar el projecte, utilitzarem un espai de coworking.
D'aquesta manera, no haurem de calcular les despeses en internet, llum, gas, etc.

Per al nostre projecte, hem considerat l'espai de coworking Co23 a Vilafranca del Penedès, el qual ens proporciona un espai per un ús continuat amb connexió a internet per 150 €[45] al mes.

Servei	Cost per mes(€)	Cost amb IVA per mes (€)	Mesos	Total(€)
Co23	150	181.5	4	726

Figura 138: Cost de l'espai de coworking.
Font: Elaboració Pròpia

11.1.4 Contingències

Per a poder anar preparats, considerarem un percentatge de l'11% per a les possibles contingències que ens apareguin. Hem estimat unes contingències tan altes per a assegurar-nos del màxim que podem assolir.
Es mostra a la taula:

Costos	Cost (€)	Contingències (%)	Cost final (€)
Recursos humans	13704	11	15211,44
Recursos materials	2000	11	2220
Costos indirectes	726	11	805,86

Figura 139: Cost total per apartats amb les contingències.
Font: Elaboració Pròpia

11.1.5 Imprevistos

Per acabar, haurem de tenir en compte certs imprevistos que poden fer augmentar els costos del projecte.
Com ja hem tingut en compte els imprevistos pel que fa a les tasques, proposant solucions com sobreestimar les tasques, ara ens centrarem en els referents als recursos materials.
L'imprevist més comú pot ser que hi hagi una fallada del hardware, i per solucionar-lo hauríem d'enviar-lo a reparar o, en el pitjor cas, comprar-ne un de nou.

En el cas que requerim una reparació, ens hem informat sobre el preu mitjà de les reparacions a l'estat espanyol[45], i ronden un màxim de 200 €, per tant estimarem això.
En el cas que l'ordinador es trenqui definitivament, requerirem de 2000 € per a solucionar l'imprevist.
Per tant, tindrem una partida de màxim 2000 € per imprevistos.

11.1.6 Cost Total

Per acabar, mostrarem una taula amb el cost total del projecte.

	Cost final (€)
Recursos humans	15211,44
Recursos materials	2220
Costos indirectes	805,86
Total	18237,3

Figura 23: Cost de l'espai de coworking. Font: Elaboració Pròpia

11.2 Control de gestió

Anteriorment hem definit les possibles incidències que ens podem trobar i que poden fer augmentar el cost del projecte, però no hem parlat de com tractarem les desviacions en el pressupost.
Per aquest motiu, hem cregut convenient calcular-les matemàticament per a tenir una dada numèrica amb la qual treballar. Hem agrupat les diferents desviacions que ens podem trobar en diferents grups.
Anirem anomenant les possibles desviacions i com les calcularem:

- **Desviació d'hores d'una tasca:**
  $(hores\ estimades\ de\ la\ tasca\ -\ hores\ reals\ de\ la\ tasca)\ *\ Cost\ estimat$

- **Desviació dels costos en recursos humans per tasca:**
  $(cost\ estimat\ -\ cost\ real)\ *\ hores\ reals$

- **Desviació dels costos en recursos materials:**
  $(cost\ estimat\ material\ -\ cost\ real\ material)$

- **Desviació dels costos indirectes:**
  $(cost\ estimat\ indirecte\ -\ cost\ real\ indirecte)$

- **Desviació dels costos imprevistos:**
  $(cost\ estimat\ imprevist\ -\ cost\ real\ imprevist)$

- **Desviació de les hores totals:**

  $(hores\ estimades\ -\ hores\ reals)$

12. Informe de sostenibilitat

La sostenibilitat, actualment, és un factor imprescindible a tenir en compte per a qualsevol projecte.
Per aquest motiu, és imprescindible fer un informe de sostenibilitat per a mesurar l'impacte sobre la sostenibilitat dels productes i serveis informàtics creats.

Abans d'haver d'analitzar la sostenibilitat del nostre projecte i en concret, abans de fer l'enquesta de sostenibilitat de la UPC, obviàvem molts factors que s'inclouen en el conjunt de la sostenibilitat.
Si bé és veritat que al llarg del Grau en Enginyeria Informàtica ens han ensenyat molt sobre totes les branques que engloben la sostenibilitat, sobretot la part que engloba el medi ambient i quin és l'impacte que hi té la nostra disciplina, no ens han ensenyat a utilitzar mètriques per mesurar l'impacte social, mediambiental i econòmic.

Crec que, personalment, coneixiem molt bé l'impacte social i mediambiental de la informàtica, fins i tot podria dir que ens interessava molt la disciplina social, el software lliure i el software com a ajuda a la societat. Però per altra banda, la part econòmica la desconeixiem, o la coneixiem molt vagament, cosa que feia que fallés en aquest àmbit. La realització d'aquest informe ens ha fet fixar molt més en les parts que dominavem menys i ens ha fet conscients de tots els àmbits que engloba.

Per a poder analitzar la sostenibilitat del projecte ens hem de fixar en tres àmbits: **l'ambiental**, **l'econòmic** i el **social**.
Abordarem cada àmbit amb dues preguntes, cadascuna referent a una part; el **Projecte Posat en Producció** (PPP), que inclou la planificació i desenvolupament i implementació del projecte i la **vida útil**, que inclou des que s'implanta el projecte fins que es desmantella.
A més, es tindran en compte els diferents **riscos**.

12.1.1 Ambiental

Fita Inicial:

*Respecte al **PPP**: Has estimat l'impacte ambiental que tindrà la realització del projecte? T'has plantejat minimitzar l'impacte, per exemple, reutilitzant recursos?*

El nostre projecte té un impacte mediambiental molt positiu, ja que pot ajudar, a partir de voluntariats, a què es tingui cura del medi ambient. Desde recollides de brosses fins a voluntariats d'ajuda als animals, aquest projecte impactarà positivament.
Tot i que hi ha un impacte negatiu, l'impacte ambiental negatiu és difícil d'estimar, ja que en el nostre cas, l'únic que impacta en el medi ambient és el hardware que utilitzem (com es recicla, que es fa del material, etc.) i sobretot, l'electricitat que gastem.
Com el hardware que utilitzarem només serà un ordinador de sobretaula i servidors per allotjar l'aplicació web, l'única cosa que podem fer per minimitzar l'impacte és utilitzar servidors existents, sense crear una planta de servidors exclusius per la nostra aplicació web.

Per altra banda, per estimar el cost de l'electricitat que gastem podríem tenir en compte el que gasta un ordinador de sobretaula, que segons *tarify*[47] és de 1,176kWh si tenim de mitja l'ordinador encès durant 5 hores al dia.
Per altra banda, un servidor com el que tindríem operatiu 24 hores gastaria 0.672kWh.

*Respecte a la **vida útil** del projecte: Com es resol actualment el problema que vols abordar (estat de l'art)? En què millorarà ambientalment la teva solució a les existents?*

La meva solució serà positiva pel medi ambient, ja que inclourà dins d'ella moltes activitats mitjançant voluntariats o experiències la finalitat dels quals serà incidir i impactar positivament en aquest àmbit. Moltes solucions existents no inclouen aquest tipus de voluntariats, i les que les inclouen tenen un impacte similar o superior al de la meva solució, ja que s'han creat amb molt hardware d'una companyia, i aquest projecte es realitza des d'un sol ordinador, incidint molt menys i gastant menys en electricitat i hardware

Fita Final:

*Respecte al **PPP**: Has quantificat l'impacte ambiental de la realització del projecte? Quines mesures has pres per a reduir-lo? Has quantificat la reducció?*

No s'ha quantificat ja que hem desenvolupat aquest projecte íntegrament en un ordinador personal propi, pel que gasta la mateixa electricitat que ja gastaria de manera habitual. Sí que és veritat que per la realització del treball ha pogut augmentar una mica l'ús d'aquesta, però no significativament.
En tot cas, el cost ambiental no és significatiu, ja que no s'ha destinat cap recurs extra per a la realització d'aquest treball, per tant, ens serviria com a quantificació l'estimació feta a la Fase Inicial.
Tal com hem dit en aquell punt, el nostre projecte deriva directament en un impacte molt positiu al medi ambient ja que la realització de voluntariats d'aquest tipus l'afavoreixen.

*Respecte al **PPP**: Si fessis de nou el projecte, podries fer-lo amb menys recursos?*

Crec que no és possible utilitzar menys recursos que un ordinador per a desenvolupar aquest projecte.
S'ha utilitzat l'únic recurs d'un ordinador personal, pel que si volguessim utilitzar menys recursos elèctrics, hauriem de trobar un ordinador que en gastés menys, el que voldria dir un ordinador menys potent i, per tant, començaria a ser contraproduent per què com més ràpid desenvolupem, menys energia gastarem. Si un ordinador és menys potent, serà més lent per a nosaltres desenvolupar, i acabarem gastant la mateixa energia.

*Respecte a la **vida útil** del projecte: Quins recursos estimes que s'utilitzaran durant la vida útil del projecte? Quin serà el seu impacte ambiental?*

Si el projecte es puja a un servidor, la despesa elèctrica d'aquest servidor tindria un impacte ambiental clar ja que hauria d'estar encès totes les hores del dia.

A més, tots els dispositius en els que s'utilitzi l'aplicació tindran un impacte ambiental també per l'electricitat, però també per tots els minerals que necessiten per funcionar. Però aquest impacte no és produït directament per la nostra aplicació ja que sense ella també existirien.
*Respecte a la **vida útil** del projecte: El projecte permetrà reduir l'ús d'altres recursos? Globalment, l'ús del projecte millorarà o empitjorarà la petjada ecològica?*

Si fem un balanç, la nostra aplicació millorarà la petjada ambiental ja que la despesa elèctrica i impacte ambiental que produeix la seva implementació i ús, s'equilibra i passa a tenir un impacte molt més positiu a mesura que aquesta s'utilitzi, ja que es crearan voluntariats inspirats en el medi ambient i es faran accions que revertiran la petjada ecològica i faran un favor al medi ambient.

*Respecte als **riscos** del projecte: Podrien produir-se escenaris que fessin augmentar la petjada ecològica del projecte?*

Sí, l'augment de demanda sobre l'aplicació podria generar l'ús d'un major número de servidors per a repartir millor el trànsit depenent de la localització, i alhora, augmentar la despesa elèctrica, i alhora, augmentar la petjada ecològica.
Però pel mateix motiu que comentavem anteriorment, si augmenta la demanda d'ús de l'aplicació, es generarien més voluntariats ambientals, per estadística, i per aquesta raó la petjada ecològica disminuiria.

12.1.2 Econòmic

Fita Inicial:

*Respecte al **PPP**: Has estimat el cost de la realització del projecte (recursos materials i humans)?*

Sí, a la secció anterior he estimat el cost dels recursos humans (els sous dels qui treballen en el projecte), els dels recursos materials, costos indirectes, els imprevistos, les contingències i les potencials desviacions que ens podem trobar.

*Respecte a la **vida útil** del projecte: Com es resol actualment el problema que vols abordar (estat de l'art)? En què millorarà econòmicament la teva solució a les existents?*

La meva solució millora les existents per un motiu, actualment les existents han de mantenir la seva plataforma mitjançant subvencions o anuncis. La nostra plataforma econòmicament anirà a càrrec de les empreses que ofereixen experiències i voluntariats, per el qual no ens hem de preocupar pel manteniment econòmic de l'app, que normalment es fa mitjançant abusius anuncis a la plataforma.
A més, la plataforma s'ofereix gratuïtament als usuaris, sense la necessitat de pagar per inscriure's o per fer voluntariats.

Fita Final:

*Respecte al **PPP**: Has quantificat el cost (recursos humans i materials) de la realització del projecte? Quines decisions has pres per a reduir-ne el cost? Has quantificat aquest estalvi?*

Com el projecte l'he realitzat en solitari, no he quantificat el cost dels recursos humans ja que no ho necessitava fer més allà de l'estimació donada a l'apartat anterior. De la mateixa manera que amb els materials, ja que he utilitzat un ordinador personal sense cap despesa extra i expressa per al projecte.
Per a que el cost fos el mínim vaig triar aquesta manera de realitzar el treball, amb recursos materials que ja tenia i una única persona com a recursos humans sense comptar al director.
Per aquesta raó no crec que es pugui reduir més el cost del projecte.

*Respecte al **PPP**: S'ha ajustat el cost previst al cost final? Has justificat les diferències?*

Com no hem quantificat el cost final, podem donar com bona l'estimació, ja que no tenim manera de quantificar-lo.
A més, hem utilitzat exactament els mateixos recursos que al calcular l'estimació, i donat que l'estudi que es va fer va ser el més específic i contrastat possible, podem confiar en els resultats.

*Respecte a la **vida útil** del projecte: Quin cost estimes que tindrà el projecte durant la seva vida útil? Es podria reduir aquest cost per fer-lo encara més viable?*

Tindrà sobretot un cost material de manteniment de servidors, que es podria reduir utilitzant només els necessaris i fent un estudi de mercat per a saber quins tenen la millor qualitat en relació al preu més baix.
En lo referent als costos en recursos humans és diferent, ja que s'hauria de mantenir almenys un sou mensual d'un desenvolupador el qual s'encarregui de mantenir l'aplicació. D'aquesta manera només necessitariem gastar en un recurs humà i estaria el cost reduït al màxim.

*Respecte a la **vida útil** del projecte: S'ha tingut en compte el cost dels ajustos / actualitzacions / reparacions durant la vida útil del projecte?*

Sí, a l'apartat anterior hem estimat com a contingències tots aquests ajustos / actualitzacions / reparacions referents a l'aplicació, per així estar coberts en cas que haguem fet un pressupost i sense necessitat d'augmentar-lo

*Respecte als **riscos** del projecte: Podrien produir-se escenaris que perjudiquessin la viabilitat del projecte?*

Qualsevol problema amb l'operativitat de l'aplicació podria generar un efecte negatiu als usuaris de l'aplicació i fer que la deixessin d'utilitzar per desconfiança. En aquest punt la viabilitat del projecte seria greument perjudicada ja que sense usuaris, no hi ha aplicació.

12.1.3 Social

Fita Inicial:

*Respecte al **PPP**: Que creus que t'aportarà a nivell personal la realització del projecte?*

Aquest projecte m'aportarà experiència en un àmbit en el qual no era expert, el qual impacta a la societat d'una manera molt positiva.
Per aquest motiu, m'aportarà satisfacció i felicitat, ja que estic implantant una solució que socialment és necessària i perquè estic treballant per una causa molt bonica que pot ajudar als altres de diferents formes; ajudant a qui ho necessita i a qui vol ajudar.

*Respecte a la **vida útil** del projecte: Com es resol actualment el problema que vols abordar (estat de l'art)? En que millorarà socialment (qualitat de vida) la teva solució a les existents?*

La meva solució millorarà en qualitat de vida a tota la societat.

En primer lloc, ajudarà a fer que tot funcioni millor, ja que els voluntariats venen a resoldre necessitats de la societat que no estan cobertes per una incapacitat governamental o de qui administra.
En segon lloc, ajudarà a les persones a sentir-se útils i bones persones, i això augmentarà la seva qualitat de vida i la seva forma de ser. A més a més, aquestes persones voluntàries podran gaudir d'experiències que augmentaran la seva qualitat de vida.
En tercer lloc, segons quins voluntariats, ajudaran a les persones més desfavorides a viure millor millorant la seva qualitat de vida.
I finalment, ajudarà a les empreses i al seu departament de Responsabilitat Social Corporativa.

*Respecte a la **vida útil** del projecte: Existeix una necessitat real del producte?*

Sí, existeix una necessitat real d'unificar en una plataforma diferents voluntariats de forma que sigui més fàcil poder ser voluntari. També, poder arribar a fer un voluntariat des de la mateixa plataforma amb facilitat.
També existeix la necessitat des de les empreses d'ampliar el sentiment de marca cap a elles. Això ho soluciona la plataforma.
A més, la gamificació dels voluntariats és molt cridanera per als usuaris.

Fita Final:

*Respecte al **PPP**: La realització del projecte ha implicat reflexions significatives a nivell personal, professional o ètic de les persones que han intervingut?*

Sí, personalment he reflexionat molt sobre la utilitat dels voluntariats i sobre el perquè són necessaris en aquest sistema. Sent aquests en moltes ocasions un pegat per les deficiències sistemàtiques de la societat, cosa que m'ha generat contradiccions ideològiques que he pogut resoldre finalment.
Però també he acabat concluent que mentre existeixi aquest sistema, han d'existir per a pal·liar algunes de les injustícies amb les nostres mans.

*Respecte a la **vida útil** del projecte: Qui es beneficiarà de l'ús del projecte? Hi ha algun col·lectiu que pugui veure's perjudicat pel projecte? En quina mesura?*

Principalment, els voluntaris que vulguin fer voluntariats i trobin aquesta plataforma que centralitza els voluntariats i que es veuràn recompensats per xapes i podràn bescanviar-les per experiències.
Alhora, les organitzacions que necessitin ampliar el seu sentiment de marca i ajudar la societat d'una forma més directa.

Cap col·lectiu es veurà perjudicat pel projecte, ja que aquest projecte és purament positiu per a la societat i qui la integra.
Qui es pot veure perjudicat és la competència tecnològica de la nostra aplicació, a qui podem prendre-li usuaris.

*Respecte a la **vida útil** del projecte: En quina mesura soluciona el projecte el problema plantejat inicialment?*

Aquest projecte soluciona, com comentarem en les conclusions, perfectament el problema plantejat inicialment, ja que s'ha creat una aplicació que satisfà totes les problemàtiques inicials i permet fer tot tipus de voluntariats, incloure un sistema de gamificació, obtenir mètriques i permet als dos tipus d'usuari obtenir en forma d'aplicació web totes les funcionalitats que es buscaven en un principi

*Respecte als **riscos** del projecte: Podrien produir-se escenaris que fessin que el projecte fos perjudicial per algun segment particular de la població?*

No, en la meva opinió, no hi ha cap factor que produeixi que la nostra aplicació, fortament encarada a solucionar problemàtiques socials, fos perjudicial per algun segment de la població. Ans el contrari, servirà per a que la gent més perjudicada per aquest sistema, acabi sent menys castigada.

*Respecte als **riscos** del projecte: Podria crear el projecte algun tipus de dependència que deixés als usuaris en posició de debilitat?*

No hauria de fer-ho. A primera vista només hi ha una possibilitat de crear alguna dependència amb la nostra aplicació i és la gamificació.
La voluntat d'acumular xapes per a bescanviar-les per les màximes experiències possibles pot portar algú a desenvolupar algun tipus d'addicció, però com aquest procés d'obtenció de xapes és un procés llarg i en el que has de fer un voluntariat de per mig, això dificulta que es generi aquesta dependència ja que el procés és costós.

13. Conclusions

En aquest punt, i per poder donar per finalitzat el treball, hem d'analitzar tota la feina feta per extreure conclusions.

Primer de tot, necessitem analitzar si els objectius que proposàvem a l'Abast del projecte han estat satisfets de manera satisfactòria.

Referent al primer objectiu, hem creat un sistema de comunicació en temps real entre les organitzacions i els voluntaris, a partir de notificacions en temps real que les organitzacions poden enviar a tots aquells voluntaris que hagin escanejat el QR per a confirmar assistència en aquell voluntariat específic.
Hem resolt, doncs, el primer propòsit de manera satisfactòria utilitzant l'API de firebase *Cloud Messaging* per a trametre i rebre notificacions *push* web.
Quant al segon objectiu, hem satisfet l'objectiu de la gamificació. Un voluntari pot arribar a fer una experiència mitjançant la participació en voluntariats si ha obtingut les xapes suficients, de la manera tal com ens proposàvem.
Pel que fa al tercer, gràcies al qual per nosaltres és un correcte disseny de la capa del domini, hem pogut obtenir mètriques tals com la nota mitjana dels voluntariats / experiències de cada organització i totes les opinions dels voluntaris en cada voluntariat, o totes les opinions de cada voluntari en general, satisfent així l'objectiu marcat.
Per acabar, no hem pogut completar l'últim objectiu proposat, per falta de servidor.
Es podria dir que l'únic que ens cal per acabar d'assolir l'abast del projecte és que algú ens proporcioni un servidor al que poder pujar el projecte, ja que no n'hem pogut tenir un des de la universitat.

A més a més, com aquest treball és un treball d'Enginyeria del Software, hi ha certes competències d'aquesta que hem assolit.
Podem concloure que aquest projecte ha tractat i ha solucionat un problema o repte atorgat per l'empresa DAMM i compartit pel mercat actual.
Per aquest projecte, hem hagut de definir juntament amb DAMM els requisits del sistema software i els hem hagut de gestionar per a transformar-los en Històries d'Usuari, ja que actuàvem com a Proxy Product Owner.
Seguidament, i tenint en compte tots aquests requisits, ens hem encarregat de dissenyar la solució software adequada per a cada capa (Presentació, Domini, Persistència), utilitzant mètodes d'Enginyeria del Software.
En aquestes solucions, hem prioritzat totes aquelles tecnologies ètiques properes al software lliure o al codi obert. Per exemple, la BBDD és PostgreSQL, el Front-End és React i el Back-End és *Spring*, de codi obert. Així mateix, com aquesta solució és causada per un problema social, hem tingut en compte els aspectes més socials del món tecnològic.
Un cop dissenyada la solució, ens hem dedicat a desenvolupar el sistema software complex amb el qual ha de comptar aquesta per a poder dur-se a terme. Aquest software complex ha estat avaluat en diferents iteracions (ja que hem usat metodologies *Àgils*) per a millorar o mantenir aquest sistema de la forma adequada.
Com en qualsevol software de l'estil de *VoluntariApp*, necessitem emmagatzemar totes les dades que han d'aparèixer en aquesta aplicació, i que aquestes siguin segures i fàcils

d'accedir. Per aquest motiu hem hagut d'especificar conforme als requisits la nostra Base de dades, i dissenyar-la i implementar-la de manera que funcionés adequadament dins d'un sistema software de les característiques d'aquest projecte.
I, per tant, al tenir una BBDD, hem hagut d'administrar-la per assegurar-nos de què les connexions no fallin, que tot funcioni de manera correcta i conforme a com hauria de funcionar perquè el software complet sigui consistent, administrar les creacions i crides a aquesta…
Per especificar més, aquest sistema software s'ha decidit desenvolupar en forma d'aplicació web, amb una arquitectura en tres capes centrada en els serveis, per tant, d'acord amb l'Enginyeria del Software, hem desenvolupat serveis que s'invoquen mitjançant una aplicació multimèdia i ho hem fet en profunditat, ja que el gruix de treball d'aquest projecte no ha estat tant en la memòria com en el desenvolupament de l'aplicació.
També, i per respectar tots aquells principis de l'Enginyeria del Software, hem dissenyat proves per a la producció correcta de software i d'aquesta manera n'hem controlat la qualitat, assegurant-nos que no teníem cap imprevist que no ens havíem plantejat, i alhora, comprovant que l'aplicació fes el que volíem que fes.
Aquestes proves s'han realitzat en tot el domini i a través de tests unitaris que ens asseguren el funcionament satisfactori de cada part del codi.

Sobre les possibles millores de l'aplicació, podem concloure que durant el desenvolupament del projecte, han anat sorgint idees que no estaven contemplades en un principi i s'haurien pogut convertir en funcionalitats, però que per falta de temps o d'eines no s'han pogut implementar. Per tant, hi ha millores interessants que es poden dur a terme en un futur i les anomenarem.
En primer lloc, el pas natural d'aquest projecte per a millorar-lo, és desenvolupar una adaptació mòbil de l'aplicació web existent. D'aquesta manera es podrien implementar notificacions natives als dispositius mòbils i els usuaris tindrien una solució personalitzada si utilitzen el mòbil. Dur a terme aquesta millora seria tan senzill com convertir el front-end de *React* a *React Native*. Com els dos llenguatges fan servir *React*, es pot convertir amb certa facilitat[48].
En segon lloc, la següent millora lògica és un següent pas i consisteix a pujar l'aplicació web a un servidor per a poder accedir-hi des de qualsevol dispositiu. És un pas necessari per a poder fer-la servir.
En tercer lloc, una altra possible millora podria ser afegir la localització en temps real dels usuaris en els voluntariats perquè les organitzacions puguin localitzar el grup de voluntaris que hi participen, i aquest es realitzi de la forma com estava dissenyat.
En darrer lloc, es podria afegir un apartat de recomanacions de voluntariats i experiències basat en algun dels algorismes existents per a recomanacions.

Per acabar, i a forma de reflexió final, estic content i satisfet amb els resultat obtinguts, sobretot en la part de l'aplicació web, ja que hem sabut crear un resultat molt vistós aprofitant tecnologies existents.
Hem pogut satisfer tots els requisits i hem aconseguit un resultat operatiu i provat.
En addició, i pel que també estic satisfet, és que hem pogut presentar a la DAMM una demostració íntegra de tota l'aplicació i l'han validat i valorat de manera molt positiva, el que ens afegeix un valor al projecte, ja que els que consideràvem d'alguna manera com al nostre client, han sortit contents al veure el resultat assolit.

14. Bibliografia

[1] EQUIP PAE DAMM Q2 20/21 (2021) Enquesta per als voluntaris [en línia] Google Forms.
[Consultat: 24 setembre 2021] Disponible a Internet <https://docs.google.com/forms/d/1TkUhrygunMJh6duW7ly4ysxlglyIg0GXV0O7_GmArSg>

[2] AGENCIA EFE (2018) Welever, la empresa española que quiere cambiar el mundo [en línia] Agencia EFE.
[Consultat: 24 setembre 2021] Disponible a Internet <https://www.efe.com/efe/espana/efeempresas/welever-la-empresa-espanola-que-quiere-cambiar-el-mundo/50000908-3813585>

[3] HACES FALTA (2021) Haces Falta. Fundación Hazlo Posible [en línia] Haces Falta. [Consultat: 28 setembre 2021] Disponible a Internet <https://www.hacesfalta.org/>

[4] VOLUNTEER WORLD (2021) Wildlife Volunteering [en línia] Volunteer World. [Consultat: 28 setembre 2021] Disponible a Internet <https://www.volunteerworld.com/en/volunteer-abroad/wildlife-conservation>

[5] MOVILIZA-T (2021) Proyecto Moviliza-T. Quienes somos [en línia] Moviliza-T, [Consultat: 28 setembre 2021] Disponible a Internet <https://www.movilizat.org/es/13-Proyecto-Moviliza-T/35-Quienes-Somos>

[6] GIVINGWAY (2021) Driving local impact, globally [en línia] GivingWay. [Consultat: 28 setembre 2021] Disponible a Internet <https://www.givingway.com/>

[7] MORENO, O (2019) Pruebas unitarias: imprescindibles para programar [en línia]. [Consultat: 03 octubre 2021] Disponible a Internet <https://www.yeeply.com/blog/que-son-pruebas-unitarias/>

[8] FACEBOOK OPEN SOURCE(2022) React - Una biblioteca de JavaScript para construir interfaces de usuario
[en línia] Facebook Open Source.
[Consultat: 16 gener 2022] Disponible a Internet <https://es.reactjs.org/>

[9] FACEBOOK OPEN SOURCE (2022) Presentando JSX - React [en línia] Facebook Open Source.
[Consultat: 16 gener 2022] Disponible a Internet <https://es.reactjs.org/docs/introducing-jsx.html>

[10] UNIVERSITAT D'ALACANT (2022) DOM (Document Object Model) - Servicio de Informática XHTML y CSS. [en línia] Universitat d'Alacant.
[Consultat: 16 gener 2022] Disponible a Internet <https://si.ua.es/es/documentacion/xhtml-css/dom-document-object-model.html#:~:text=El%20modelo%20de%20objeto%20de,su%20estructura%2C%20estilo%20y%20contenido>

[11] FRANCH,XAVIER I GÓMEZ, CRISTINA (2020) Presentation Layer. Arquitectura del Software. Facultat d'Informàtica de Barcelona.
[Consultat: 16 gener 2022]

[12] VAN WELIE, MARTIJN (2008) Faceted Navigation - Interaction Pattern Library. [en línia] Welie.
[Consultat: 16 gener 2022] Disponible a Internet <http://welie.com/patterns/showPattern.php?patternID=main-navigation>

[13] VAN WELIE, MARTIJN (2008) Main Navigation - Interaction Pattern Library. [en línia] Welie.
[Consultat: 16 gener 2022] Disponible a Internet <http://welie.com/patterns/showPattern.php?patternID=faceted-navigation>

[14] Spring (2021) Spring | Home. [en línia] Spring.
[Consultat: 16 gener 2022] Disponible a Internet <https://spring.io/>

[15] Spring Boot (2021) Spring Boot. [en línia] Spring.
[Consultat: 16 gener 2022] Disponible a Internet <https://spring.io/projects/spring-boot#overview>

[16] Wikipedia (2022) Java (lenguaje de programación). [en línia] Wikipedia.
[Consultat: 16 gener 2022] Disponible a Internet <https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)>

[17] IONOS (2022) CRUD: la base de la gestión de datos. [en línia] IONOS Cloud S.L.U.
[Consultat: 16 gener 2022] Disponible a Internet <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/crud-las-principales-operaciones-de-bases-de-datos/>

[18] SMARTBEAR (2021) API Endpoints - What Are They? Why Do They Matter [en línia] SmartBear Software.
[Consultat: 16 gener 2022] Disponible a Internet <https://smartbear.com/learn/performance-monitoring/api-endpoints/>

[19] SMARTBEAR (2021) API Development for Everyone [en línia] SmartBear Software.
[Consultat: 16 gener 2022] Disponible a Internet <https://swagger.io/>

[20] REFACTORING GURU(2022) Factory Method [en línia] SmartBear Software.
[Consultat: 16 gener 2022] Disponible a Internet <https://refactoring.guru/design-patterns/factory-method>

[21] SENDGRID (2022) Email Delivery Service [en línia] SendGrid.
[Consultat: 16 gener 2022] Disponible a Internet <https://sendgrid.com/>

[22] HIBERNATE (2022) Hibernate. Everything data. [en línia] Hibernate.
[Consultat: 16 gener 2022] Disponible a Internet <https://hibernate.org/>

[23] EL BLOG DE AGGYLON (2018)Metodologías ágiles de desarrollo de software [en línia] Aggylon.
[Consultat: 28 setembre 2021] Disponible a Internet <https://www.aggylon.es/es/articulo/0/0/todo/51/metodologas-giles-de-desarrollo-de-software>

[24] SCRUM (2021) ¿Qué es Scrum? [en línia] Scrum.
[Consultat: 28 setembre 2021] Disponible a Internet <https://www.scrum.org/resources/blog/que-es-scrum>

[25] GARCÍA, R (2019) ¿Qué es el backlog? [en línia] Muy Agile.
[Consultat: 28 setembre 2021] Disponible a Internet <https://muyagile.com/que-es-el-backlog/>

[26] REQUENA, A (2018) Que es un Sprint de Scrum [en línia] Open Webinars.
[Consultat: 28 setembre 2021] Disponible a Internet <https://openwebinars.net/blog/que-es-un-sprint-scrum/>

[27] BALLARIN, A (2017) Proxy Product Owner: ¿porque? ¿como alcanzar un auténtico PO? [en línia] IT Nove.
[Consultat: 28 setembre 2021] Disponible a Internet <https://itnove.com/blog/scrum/product-owner/proxy-product-owner-como-alcanzar-un-autentico-product-owner/>

[28] DELOITTE (2021 ¿Cuál es la metodología más adecuada para tu proyecto? [en línia] Deloitte.
[Consultat: 28 setembre 2021] Disponible a Internet <https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html>

[29] SINNAPSIS (2020) SCRUM: ÉPICA, HISTORIA Y TAREA [en línia] Sinnapsis.
[Consultat: 28 setembre 2021] Disponible a Internet <https://www.sinnaps.com/blog-gestion-proyectos/scrum-epica>

[30] TAIGA (2021) Taiga: Your open source agile project management software [en línia] Taiga.
[Consultat: 28 setembre 2021] Disponible a Internet <https://www.taiga.io/>

[31] MARTIN, A (2018) Scrum Board para visualizar la situación del Sprint [en línia] Urtanta.
[Consultat: 28 setembre 2021] Disponible a Internet <https://urtanta.com/scrum-board/>

[32] GIT (2021) Git [en línia] Git.
[Consultat: 28 setembre 2021] Disponible a Internet <https://git-scm.com/>

[33] GITHUB (2021) GitHub [en línia] GitHub.
[Consultat: 28 setembre 2021] Disponible a Internet <https://github.com/>

[34] DE SOUZA, I (2020) Front-end y back.end: ¿qué son y en qué se distinguen? [en línia] Rock Content.

[Consultat: 03 octubre 2021] Disponible a Internet <https://rockcontent.com/es/blog/front-end-y-back-end/>

[35] GOOGLE (2021) Google Meet [en línia] Google. [Consultat: 02 octubre 2021] Disponible a Internet <https://meet.google.com/>

[36] GOOGLE (2021) Google Chrome [en línia] Google. [Consultat: 02 octubre 2021] Disponible a Internet <https://www.google.com/intl/ca_ES/chrome/>

[37] GOOGLE (2021) Google Drive [en línia] Google. [Consultat: 02 octubre 2021] Disponible a Internet <https://www.google.es/intl/ca/drive/>

[38] JET BRAINS (2021) IntelliJ IDEA: The Capable & Ergonomic Java IDE [en línia] JetBrains s.r.o. [Consultat: 02 octubre 2021] Disponible a Internet <https://www.jetbrains.com/idea/>

[39] MICROSOFT (2021) Visual Studio Code - Code Editing [en línia] Seattle: Microsoft. [Consultat: 01 octubre 2021] Disponible a Internet <https://code.visualstudio.com/>

[40] PGADMIN (2021) pgAdmin - PostgreSQL Tools [en línia]. [Consultat: 02 octubre 2021] Disponible a Internet <https://www.pgadmin.org/>

[41] THOMAS, A (2021) Gantt Project: Free Project Management Application [en línia] Praga: Bard Software s.r.o. [Consultat: 01 octubre 2021] Disponible a Internet <https://www.ganttproject.biz/>

[42] GLASSDOOR (2021) ¿Cuánto gana un Jefe de proyecto? [en línia]. [Consultat: 10 octubre 2021] Disponible a Internet <https://www.glassdoor.es/Sueldos/jefe-de-proyecto-sueldo-SRCH_KO0,16.htm>

[43] GLASSDOOR (2021) ¿Cuánto gana un Proxy Product Owner? [en línia]. [Consultat: 10 octubre 2021] Disponible a Internet <https://www.glassdoor.es/Sueldos/proxy-product-owner-sueldo-SRCH_KO0,19.htm>

[44] GLASSDOOR (2021) ¿Cuánto gana un Developer? [en línia]. [Consultat: 10 octubre 2021] Disponible a Internet <https://www.glassdoor.es/Sueldos/barcelona-developer-sueldo-SRCH_IL.0,9_IM1015_KO10,19.htm>

[45] CO23 (2021) Co23 - Espai Coworking a Vilafranca del Penedès [en línia]. [Consultat: 08 octubre 2021] Disponible a Internet <https://co23.cat/>

[46] CRONOSHARE (2020) ¿Cuánto cuesta reparar un ordenador? [en línia] Cronoshare. [Consultat: 08 octubre 2021] Disponible a Internet <https://www.cronoshare.com/cuanto-cuesta/reparar-ordenador>

[47] TARIFY (2020) Cuál es el consumo de un ordenador de sobremesa [en línia] tarify.
[Consultat: 08 octubre 2021] Disponible a Internet
<https://tarify.es/noticias/cual-consumo-ordenador-sobremesa>

[48] ZUEV, ANTON (2022) Cómo convertir una app de React en una app de React Native [en línia] Adictos al trabajo.
[Consultat: 16 gener 2022] Disponible a Internet
<https://tarify.es/noticias/cual-consumo-ordenador-sobremesa>

[49] ÁLVAREZ CAULES, CECILIO (2020) Spring @Autowired y la inyección de dependencias [en línia] Arquitectura Java.
[Consultat: 16 gener 2022] Disponible a Internet
<https://tarify.es/noticias/cual-consumo-ordenador-sobremesa>

[50] BARRIOS, EDUARDO (2020) Patrón Repositorio (Repository Pattern) [en línia] Dev.to.
[Consultat: 16 gener 2022] Disponible a Internet
<https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0-5goj>