UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Culture finder

[Github](#) - [Taiga](#) - [Drive](#) - [Project Record Track](#)

Cognom	Nom	Responsable	UPC e-mail	Taiga	GDrive	Github
Morón	Daniel	Management of services	daniel.moron.roces@estudiantat.upc.edu	danielmr_6	daniel.moron.roces@estudiantat.upc.edu	danielmr6
Risueño	Iván	Inception	ivan.risueno@estudiantat.upc.edu	ivan.risueno	ivan.risueno@estudiantat.upc.edu	ivan-risueno
Rodriguez	Marc	Sprint 2	marc.rodriguez.martin@estudiantat.upc.edu	marcrd11	marc.rodriguez.martin@estudiantat.upc.edu	MarcRd11
Duch	Marc	Sprint 1	marc.duch@estudiantat.upc.edu	marc.duch	marc.duch@estudiantat.upc.edu	Marcarrones
Moreno	Miguel	Final demo and closing documentation	miguel.moreno.alcaraz@estudiantat.upc.edu	MiguelMorenoAlcaraz	miguel.moreno.alcaraz@estudiantat.upc.edu	MiguelMorenoAlcaraz
Delgado	Òscar	Sprint 3	oscar.delgado.gomez@estudiantat.upc.edu	oscard14	oscar.delgado.gomez@estudiantat.upc.edu	oscard147

1. Sprint master report **4**

2. Updated “NOT” list **5**

3. List of stakeholders **6**

3.1. Joves que busquen activitats per a fer: 6

3.2. Famílies que volen fer activitats familiars: 6

3.3. Interessats en tipus d’esdeveniments concrets: 7

3.4. Organitzadors d’esdeveniments i activitats culturals: 7

3.5. Equips de PES: 7

3.6. Product owner: 7

3.7. La Generalitat: 7

3.8. Android: 7

3.9. Flutter: 7

3.10. Spring boot: 8

3.11. GitHub: 8

3.12. Google: 8

3.13. Amazon: 8

3.14. Docker: 8

3.15. Competidors: 8

4. Table of epics and user stories **9**

5. Critical mockups **12**

6. Architectural physical diagram **21**

7. UML class diagram **23**

8. Description of working methodology and technology **25**

8.1. Visió General 25

8.2. Gestió Projecte 25

8.3. Gestió versions 26

8.4. Comunicació d’equip 28

8.5. Frameworks i llenguatges 28

8.6. IDEs 29

8.7. Gestió de qualitat 29

8.8. Gestió de proves 30

8.9. Gestió de configuracions 30

8.10. Interacció amb altres grups 30

8.11. Convencions de codi 30

8.11.1. Variables 30

8.11.2. Funcions i mètodes 31

8.11.3. Classes 32

8.11.4. Fitxers i directoris 32

8.12. Gestió de bases de dades 33

9. Consumer and producer of services **34**

10. Adreces d'interès **35**

1. Sprint master report

Donem ja per acabada la segona fase d'Incepció. Aquest *report* serveix com a resum de com l'equip s'ha organitzat durant aquestes dues setmanes, les tasques que ha realitzat i com s'ha afrontat el treball.

El primer que vaig fer va ser crear les tasques pertinents al Trello, i parlar amb la resta del grup de treball per a assignar-les a cadascú de manera equitativa. El dia de la primera reunió presencial vam confeccionar les primeres històries d'usuari, i més tard les vaig passar a net dins d'aquest document i al Taiga, afegint els criteris d'acceptació.

Cadascú va començar amb les seves tasques, necessitant prèviament formació. Tant els mock-ups, com la part de la metodologia i les tecnologies, com els stakeholders i l'arquitectura de l'aplicació van requerir bastants hores de formació per a tots els membres de l'equip.

A la reunió *weekly* del diumenge vaig dir de posar en comú el progrés de les tasques fins a aquell moment, i així ho vam fer. Vam retocar una mica el diagrama UML i vam fer un *planning poker* per decidir els punts de les històries d'usuari (que posteriorment canviaríem a una reunió focal que vam organitzar).

A la segona setmana, hem anat acabant totes les tasques. Ens hem focalitzat en les nostres pròpies, però donat el moment en el qual algú acaba la seva feina es posa amb un company per a ajudar. Aquesta segona setmana ha sigut més dinàmica, hem parlat més entre nosaltres i espero que aquest sigui el ritme de treball que portem durant els *sprints*.

Un cop decidides les tecnologies i acabats els *mock-ups*, hem afegit més històries d'usuari, hem trencat algunes altres i hem reconsiderat els punts tenint en compte les hores prèvies de formació a les tasques, la implementació d'aquestes funcionalitats i els tests associats a cadascuna.

Tot plegat hem treballat de manera estructurada, organitzant i assignant les tasques que cadascú ha hagut de fer, i davant situacions que ho mereixien he organitzat reunions focals amb l'objectiu de posar en comú la feina feta i aclarir dubtes, a més de sincronitzar documentació prèvia per tal d'evitar obsolescències de funcionalitats i inconsistències entre parts del document.

Volem afrontar els *sprints* amb ganes, dedicació i actitud, i esperem que davant algunes dificultats ens puguem sortir sense massa problema.

2. Updated “NOT” list

A continuació mostrem la NOT-List actualitzada que pertany a la fase 2 de l’Inception.
En verd surten les funcionalitats que hem afegit a l’aplicació des d'UNRESOLVED, i en vermell les que hem decidit deixar-les fora.

IN SCOPE	OUT OF SCOPE
Geolocalització	Compra d’entrades
Localitzar esdeveniments (Vista Mapa)	Integració amb taxis o altres sistemes de transport
Guardar dades dels esdeveniments que ens proporciona la API, pero només els posteriors al 1/1/2023	Afegir nous esdeveniments
Calendari d’esdeveniments (Vista Calendari)	Perfils d’organitzadors
Esdeveniments mostrats en forma de llista (Vista Llista)	Modificar esdeveniments
Poder buscar i filtrar esdeveniments per diferents criteris (geogràficament, categories, data, nom, popularitat...)	Informació dels esdeveniments a temps real
Mostrar informació detallada dels esdeveniments (Vista Detall)	Mètodes d’iniciar sessió amb reconeixement facial o empremta dactilar
Sincronització de dades (Dades de l'API agenda cultural)	Comprovació de la veracitat i confiança dels esdeveniments
Sistema multi idioma (català, castellà, anglès) [Només menú]	Funcionament fora de Catalunya
Recomanació d’activitats segons els interessos dels usuarios	Possibilitat de promocionar activitat cultural
Compartir esdeveniments a través de xarxes socials o URL	
Valoració d’esdeveniments entre 1 i 5 punts	Xat amb els responsables dels esdeveniments
Creació, modificació i alta de perfils	Traçat del camí fins l’esdeveniment / direccions
Notificar assistència als esdeveniments	Sistema multiplataforma

Llista d'esdeveniments favorits	Fòrum de comentaris sobre els esdeveniments
Creació de llistes d'esdeveniments personalitzades	
Sistema per reportar errors d'esdeveniments	
Pàgina per administrador per consultar i respondre als reports dels usuaris	
Sincronització amb Google Calendar	
Apartat de dubtes freqüents i ajuda	
Sistema de notificacions per als usuaris sobre els esdeveniments als que han afegit a alguna llista, assistiran o els poden interessar	
Estadístiques de l'esdeveniment (número assistents, valoració mitjana...)	
Mostrar els esdeveniments més populars a través d'un mapa de calor	
UNRESOLVED	

3. List of stakeholders

En el nostre projecte, com a qualsevol, hi ha una llista de persones o entitats interessades. A continuació enumerarem les que considerem més importants per a Culture Finder.

3.1. Joves que busquen activitats per a fer:

Un dels clients objectius de l'aplicació són joves que vulguin anar a concerts, festes, i esdeveniments similars enfocats principalment a ells. A l'aplicació podran trobar aquests esdeveniments amb facilitat.

3.2. Famílies que volen fer activitats familiars:

Un altre client objectiu seran pares interessats en activitats per a tota la família, o infantils per portar els nens petits, que volen poder buscar-les amb facilitat.

3.3. Interessats en tipus d'esdeveniments concrets:

Aquells aficionats a un tipus d'esdeveniment més específic, com podria ser exposicions a museus, estarien interessats en l'aplicació, ja que els proporciona informació de tots els esdeveniments d'aquest tipus a Catalunya.

3.4. Organitzadors d'esdeveniments i activitats culturals:

Els organitzadors dels esdeveniments a l'aplicació voldran que l'esdeveniment es difongui quan més possible, i el nostre projecte fa justament això.

3.5. Equips de PES:

Ja que oferim un servei a un altre equip de PES i ens ofereixen un servei també, els altres equips de PES estan interessats en la nostra aplicació.

3.6. Product owner:

És el client principal, que inverteix en el nostre projecte i dona una guia general de com ha de ser l'aplicació. En el nostre cas el professor de PES seria el Product Owner, tot i que no guia l'aplicació com faria un de real.

3.7. La Generalitat:

La base de dades dels esdeveniments culturals que es mostraran a la nostra aplicació venen de la pàgina d'esdeveniments culturals de la Generalitat.

3.8. Android:

Culture Finder serà una aplicació mòbil per a dispositius Android. I farem servir el seu IDE: Android Studio

3.9. Flutter:

Utilitzarem Flutter com a framework per la part de front de l'aplicació.

3.10. Spring boot:

Utilitzarem Spring boot com a framework per a la part de backend de l'aplicació.

3.11. GitHub:

A GitHub penjarem el front i back de l'aplicació, i l'utilitzarem per pujar actualitzacions del codi i gestionar les diferents versions i branques de cada membre.

3.12. Google:

Farem servir l'inici de sessió mitjançant Google com a alternativa a l'inici de sessió manual a l'aplicació. Amb aquest inici de sessió de Google, permetrà a un usuari sincronitzar el calendari de l'aplicació amb el de Google.

3.13. Amazon:

Utilitzarem Amazon Web Services, com a base de dades de l'aplicació.

3.14. Docker:

Utilitzarem Docker com a contenidor per a l'aplicació de Java.

3.15. Competidors:

Aplicacions similars com ara Fever, Agenda Cultural, Forsquare y Eventbite (les quals vam analitzar a l'estudi de mercat), seran stakeholders negatius, ja que la nostra aplicació seria competencia seva.

4. Table of epics and user stories

Èpica	Història d'usuari	Punts
Cerca i consulta d'esdeveniments	Consultar esdeveniments recomanats per categories	8
	Consultar esdeveniments recomanats per proximitat	8
	Consultar esdeveniments mostrats per llista	5.5
	Consultar la ubicació dels esdeveniments al mapa	7
	Consultar la meva ubicació al mapa	3.5
	Consultar la informació detallada d'un esdeveniment	7
	Cercar esdeveniments per nom	3.5
	Cercar esdeveniments filtrant per categoria	8
	Cercar esdeveniments filtrant per distància	7.5
	Cercar esdeveniments filtrant per rang de dates	6
	Cercar esdeveniments filtrant per preu	7.5
	Cercar esdeveniments filtrant per comarca	5
	Cercar esdeveniments filtrant per municipi	5
	Consultar esdeveniments mitjançant el calendari	10
	Consultar esdeveniments marcats com a favorits	6
	Consultar visualment els esdeveniments més populars al mapa	10
	Consultar ubicació en mapa d'un esdeveniment	3.5
	Veure suggeriments d'esdeveniments semblants	7.5
	Consultar assistents d'un esdeveniment	5
Compartir esdeveniments	Compartir esdeveniment per Whatsapp	8
	Compartir esdeveniment per Instagram	8
	Compartir esdeveniment per Twitter	8
	Compartir esdeveniment per Gmail	8
	Compartir esdeveniment per Outlook	8
	Mostrar informació d'un esdeveniment a l'obrir una URL	5

Gestió d'usuari	Crear Usuari	7.5
	Eliminar Usuari	5
	Consultar el meu perfil	7.5
	Canviar nom perfil	5
	Editar els meus interessos al meu perfil	7
	Canviar contrasenya	5
	Modificar foto de perfil	5.5
	Iniciar sessió	6
	Iniciar sessió amb Google	8
	Tancar sessió	4
Interaccions usuari - esdeveniments	Buscar llista	4
	Apuntar-me a un esdeveniment	4
	Desapuntar-me d'un esdeveniment	4
	Puntuar un esdeveniment	5
	Esborrar la meva puntuació d'un esdeveniment	5
	Afegir un esdeveniment a la llista de favorits	5.5
	Eliminar esdeveniments de la llista de favorits	4
	Consultar esdeveniments als que assistiré en llista	5.5
	Consultar esdeveniments als que assistiré en calendari	10
	Crear llista	5.5
	Eliminar llista	5
	Afegir esdeveniment a llista	4
	Eliminar esdeveniment de llista	4
	Editar nom de llista	5
	Editar descripció de llista	4
Incidències	Reportar error d'esdeveniment	10
	Veure totes les incidències	5
	Veure incidències obertes/tancades	5

	Canviar estat d'una incidència	8
	Respondre a una incidència	8
	Iniciar sessió a la pàgina d'administració	8
	Tancar sessió a la pàgina d'administració	5
Altres funcionalitats	Sincronitzar esdeveniments amb Google Calendar	11
	Canviar idioma	7
	Dades actualitzades	11
	Informació consistent	21
	Veure dubtes freqüents i ajuda	8
Notificacions	Configurar notificacions	8
	Notificar usuari de nous esdeveniments	10
	Notificar amb recordatori sobre esdeveniments als quals assistirà	10
	Notificar a l'usuari sobre canvis als esdeveniments marcats com a favorits	8

Els punts de cada història d'usuari representen la complexitat de cada història a implementar. Aquesta complexitat té en compte les hores que trigarem a programar-les, dependències amb altres històries i la formació requerida per a desenvolupar la solució. Els hem escollit entre tots fent un *planning poker*(https://planningpokeronline.com/).

Link al product backlog del nostre projecte a Taiga:
https://tree.taiga.io/project/marcduch-pes-agenda-cultural/backlog

5. Critical mockups

En aquest apartat mostrarem el conjunt de funcionalitats del nostre sistema a partir dels *mockups* que hem creat per tal d'entendre millor com funcionarà el nostre producte. Hem creat un prototip interactiu de l'aplicació amb Figma que es pot visualitzar en el següent enllaç:
https://www.figma.com/file/6B4fXSUHUUR2NKHxGgsDLu/CurltureFinder?node-id=915%3A769&t=ACd51XDHBj6Vk3HQ-1

Aquesta pantalla serà la primera que es veurà quan un usuari es descarregui l'aplicació i entri per primera vegada o directament tanqui la sessió.

Primerament, l'usuari pot entrar iniciant sessió indicant el correu i la seva contrasenya corresponent, amb la qual es pot activar un recordatori per no haver d'estar afegint-la constantment.

A més, a partir d'aquesta pantalla es pot fer l'inici de sessió de manera sincronitzada amb un compte de Google, i en el cas que l'usuari no tingui encara cap compte de CultureFinder, et permetrà redirigir-te a una altra pantalla per crear el seu compte.

Figura 1. Pantalla inici sessió

Aquesta és la pantalla per restablir la contrasenya. En el cas que l'usuari no se'n recordi de la seva contrasenya, donem la possibilitat de restablir-la afegint l'adreça del seu correu.

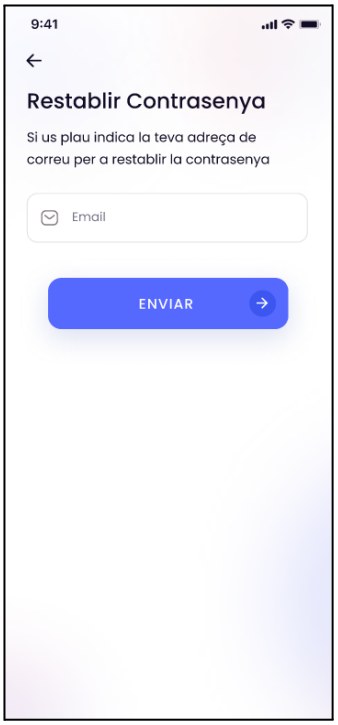
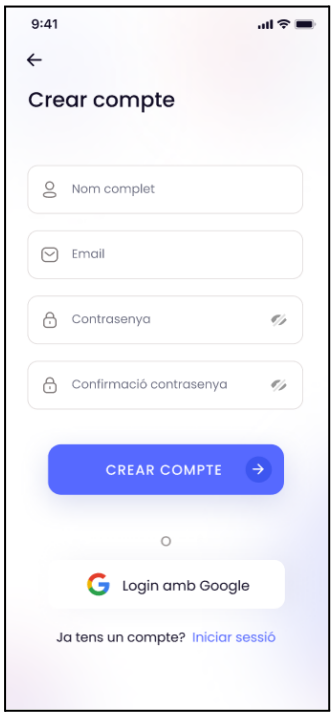


Figura 2. Pantalla restabliment contrasenya



A la pantalla de l'esquerra es redirigeix l'usuari quan selecciona l'opció de crear-se un compte. Com es pot veure s'ha d'introduir un nom d'usuari, un correu i una contrasenya. També permet l'opció de crear el compte connectant amb Google.

A l'altra pantalla s'afegeix la data de naixement, amb l'objectiu de revisar si l'usuari és major d'edat o no, l'edat mínima permesa serà de 14 anys amb la supervisió d'un adult.

Figura 3: Pantalla inicial de crear compte

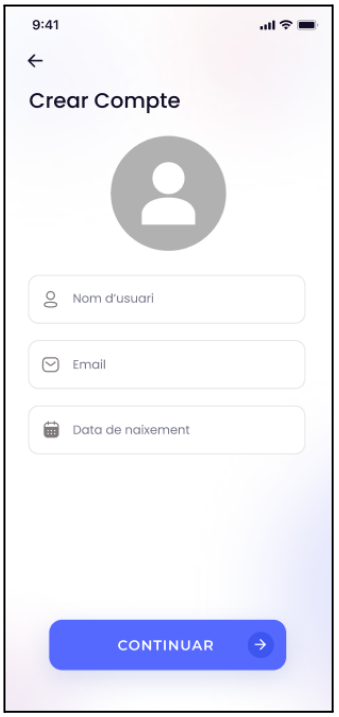


Figura 4: Pantalla següent de crear compte

En el cas que l'usuari s'hagi equivocat en seleccionar l'opció de crear compte i ja tingui un compte disponible, podrà seleccionar el botó de tornar enrere, altrament haurà de clicar en el botó de continuar.

Una vegada l'usuari ha afegit les seves dades, haurà de seleccionar els seus interessos. Gràcies a aquests filtres l'aplicació mostrarà certes recomanacions d'esdeveniments en funció de les categories seleccionades.

Com a mínim, l'usuari haurà de seleccionar 3 categories que l'interessin per ajudar al sistema a entendre millor els seus gustos personals.

Figura 5: Pantalla de seleccionar aficions

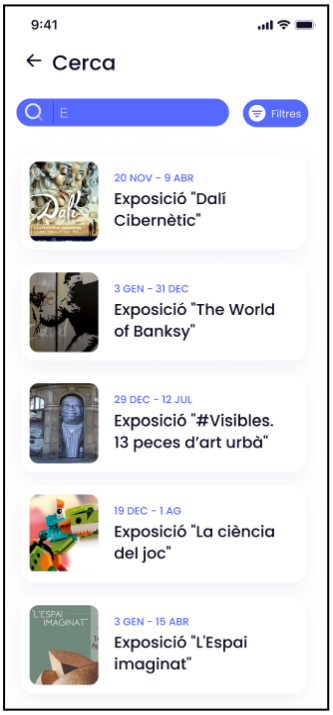


Figura 6: Pantalla de cerca

Aquesta és la pantalla que es mostra quan s'afegeix algun paràmetre de cerca a la barra de cerca que hi ha a la part superior de l'aplicació. Per exemple, en aquesta situació, es pot observar que surten tots els esdeveniments que comencen per una lletra en concret, tot i que en dates diferents.

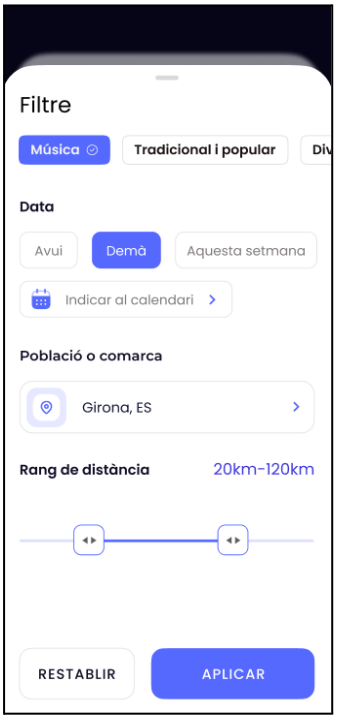


Figura 7: Pantalla de filtres

Aquesta és la pantalla que es visualitza quan l'usuari vol indicar els tipus de filtres que li interessen per la cerca corresponent. Existeix un *scroll* lateral per tal de visualitzar el conjunt de filtres esmentat. A més es pot indicar el filtre especificant la data via el calendari del sistema.

A la dreta del tot podem veure que el resultat de la cerca ha canviat, i mostra només l'exposició de l'endemà.

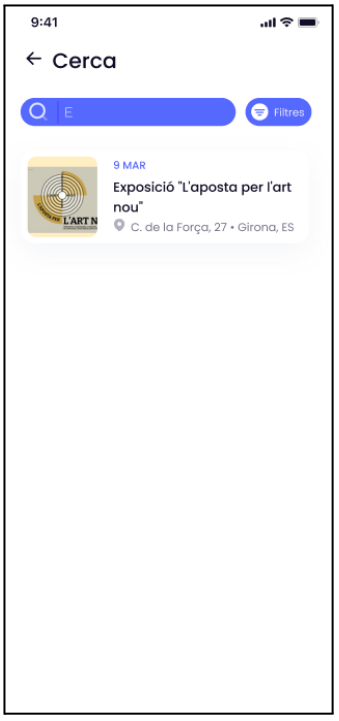
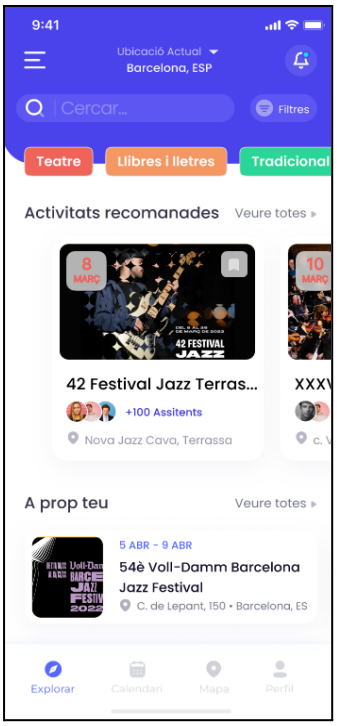


Figura 8: Pantalla de cerca amb filtres indicats



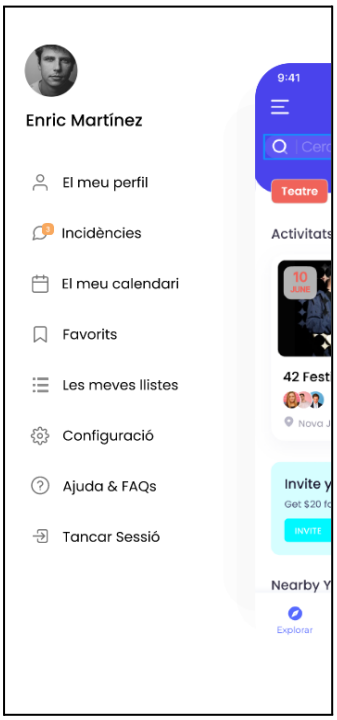
Aquesta pantalla és la primera que es mostra quan un usuari registrat entra a l'aplicació. És la pantalla principal on es mostren suggeriments d'esdeveniments als quals pot assistir l'usuari. Les activitats mostrades varien entre usuaris segons els seus interessos, cerques anteriors i localització.

A la part superior esquerra es troba la icona del menú lateral, que en ser seleccionada es desplega al costat esquerre de la pantalla. Al seu costat hi ha un desplegable des d'on es pot canviar la ubicació actual, fent que també canviïn els suggeriments mostrats per pantalla. A la seva dreta està el botó que obre les notificacions que ha rebut l'usuari.

A sota hi ha un cercador per buscar esdeveniments per nom i un botó que mostra els diferents filtres que es poden aplicar a la cerca. Just a sota es troben botons seleccionables de les categories d'esdeveniments més usades pels usuaris quan fan una cerca.

Figura 9: Pantalla d'inici de l'aplicació

A la part inferior es troben les diferents pantalles secundàries com són la cerca d'esdeveniments per calendari, per mapa i la configuració de perfil.



Aquesta pantalla mostrà diferents funcionalitats a les quals pot accedir l'usuari en un menú lateral. La primera és accedir al seu perfil, on pot veure i editar la informació d'aquest. La següent és incidències, on es mostra un *log* de les incidències reportades per l'usuari i les respostes que ha obtingut dels administradors. A sota està El meu calendari, on es mostra un calendari, que es pot sincronitzar amb Google Calendar, amb tots els esdeveniments als quals ha assistit i assistirà l'usuari.
Seguidament, està l'opció de Favorits, la qual mostra la llista d'esdeveniments que han estat marcats com a favorits. Després anant a Les meves llistes es poden veure, editar i eliminar les llistes d'esdeveniments creades per l'usuari. Just a sota està l'opció de Configuració, que permet escollir l'idioma, seleccionar quin tipus de notificacions rebre i configurar aspectes visuals de l'aplicació. A sota en Ajuda & FAQ es poden veure els dubtes més freqüents dels usuaris sobre l'aplicació, tutorials de com usar-la i informació de contacte amb els administradors.

Figura 10: Pantalla d'opcions laterals

Per últim, està l'opció de tancar sessió per així poder entrar amb un altre compte per exemple.

Aquesta és la pantalla que es mostra quan s'obre un esdeveniment seleccionat. Permet anar a la pantalla des de la qual s'ha anat seleccionant la fletxa. A la part superior dreta hi ha un botó per afegir l'esdeveniment a la llista de favorits. Si es manté pressionat el botó, es mostra l'opció d'afegir l'esdeveniment a una llista existent o a una nova. En la part superior també es pot veure una imatge de l'esdeveniment i el nombre de persones que han indicat que hi assistiran. Just al costat d'aquest número hi ha un botó per compartir l'esdeveniment per xarxes socials o correu.

Figura 11: Pantalla detallada d'esdeveniment

A partir d'aquí es mostrà tota la informació que es té de l'activitat com la data, horari, ubicació, valoració dels usuaris, descripció, categories de les quals forma part, enllaços a la seva web, preu, imatges, vídeos i telèfon de contacte. Des d'aquesta pantalla també es pot reportar si es troba algun error en la consistència de les dades de l'esdeveniment.
En la part inferior hi ha un botó per indicar que s'assistirà a l'esdeveniment.

Aquí es mostren les diferents opcions a través de les quals es pot compartir un esdeveniment. A part de les xarxes socials i els correus electrònics, també es pot copiar l'enllaç a la informació de l'esdeveniment.

Figura 12: Pantalla per compartir esdeveniment

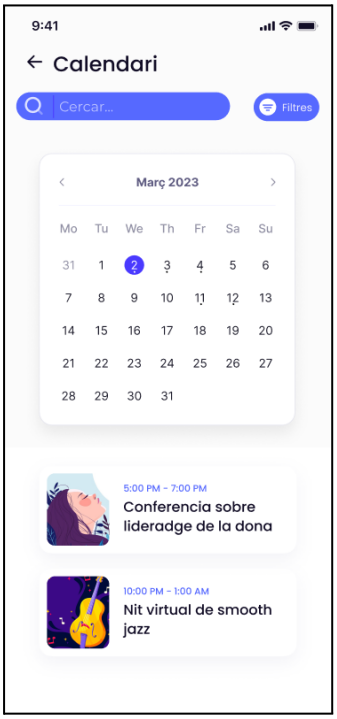


Figura 14: Pantalla de calendari

En aquesta pantalla es poden cercar i visualitzar els esdeveniments que compleixen els filtres en format de calendari. Si un dia té un punt a sota significa que hi ha algun esdeveniment aquell dia que compleix els filtres de la cerca que ha fet l'usuari. Si es selecciona un dia es mostren tots aquests esdeveniments ordenats cronològicament. A la part superior esquerra hi ha una fletxa que al ser seleccionada envia a l'usuari a la pantalla principal.

La pantalla de El meu calendari es molt similar, amb l'única diferència que mostra només els esdeveniments als quals ha assistit o assistirà l'usuari. També hi ha un toggle per sincronitzar amb Google Calendar.

Aquesta pàgina mostra per defecte el mapa amb tots els esdeveniments dels pròxims 15 dies. A la part superior es pot cercar un esdeveniment per nom i seleccionant el botó situat a baix a la dreta es poden afegir filtres pels esdeveniments que es mostren en el mapa. A dalt a la dreta hi ha un botó per centrar el mapa a la ubicació de l'usuari, que es mostra en el mapa com un punt blau. Just a sota es troben botons seleccionables de les categories d'esdeveniments més usades pels usuaris quan fan una cerca. En seleccionar categories es van filtrant les xinxetes d'esdeveniments mostrats al mapa.

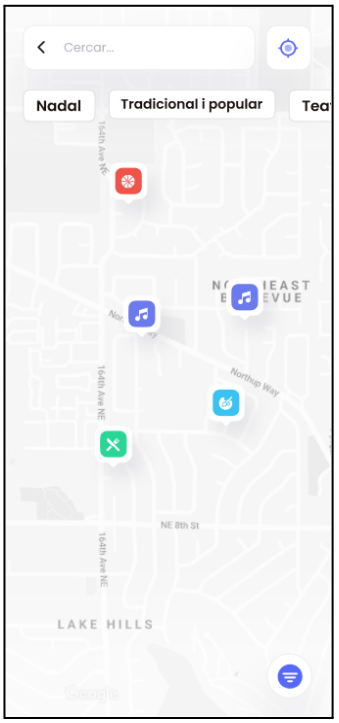


Figura 15: Pantalla de mapa

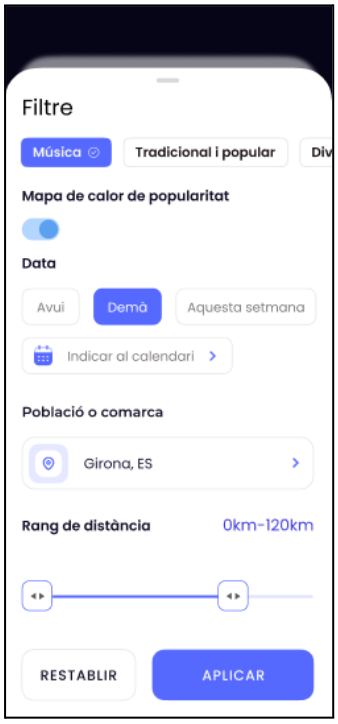


Figura 16: Pantalla de filtres en mapa

Aquesta pantalla és molt semblant a la pantalla de filtres descrita anteriorment, amb l'única diferència que aquesta inclou un toggle per indicar si es vol visualitzar el mapa de calor dels esdeveniments més populars en la data indicada al filtre.

Aquesta és la pantalla del mapa amb el filtre de mapa de calor aplicat. També es pot veure que s'han aplicat filtres de categories i s'ha seleccionat un esdeveniment. Quan es selecciona la xinxeta d'un esdeveniment, aquesta es fa més gran i es mostra certa informació important d'aquest en la part inferior de la pantalla.

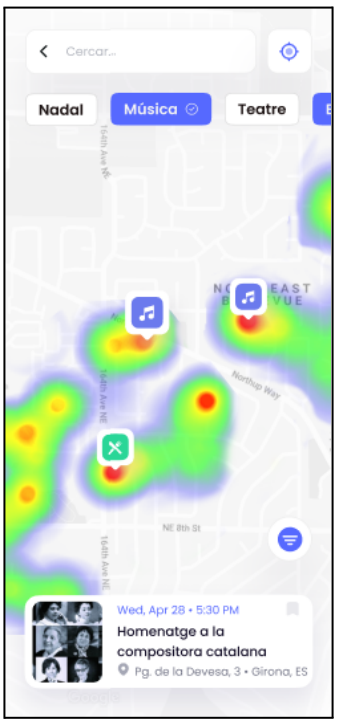


Figura 17: Pantalla de mapa de calor

En aquesta pantalla mostrem el perfil de l'usuari, on prèviament s'ha d'haver clicat al botó pertinent al menú desplegable de l'esquerra que conté la pàgina inicial.

A la part superior de la captura del mockup podem veure la imatge de perfil de l'usuari juntament amb el seu nom. Si es fa clic en el botó de canviar, es podrien modificar tots els camps relacionats amb el perfil, com poden ser l'username, el correu o la mateixa fotografia.

En canvi, a la part de sota, podem trobar els diferents interessos principals que té l'usuari. Gràcies a això, es pot tenir una idea clara de les aficions de l'usuari i si és compatible amb qualsevol altre perfil.

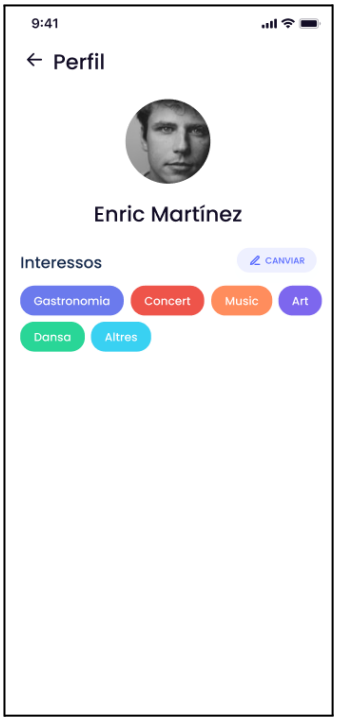
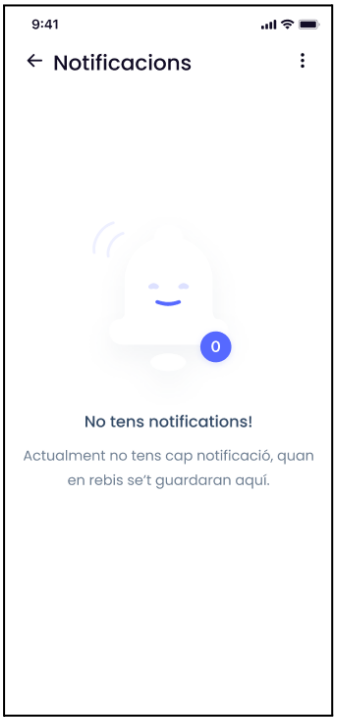
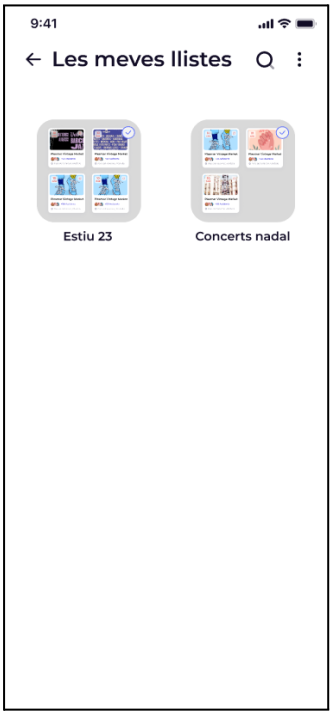


Figura 18: Pantalla de perfil



Com el mateix nom indica, en aquesta pantalla es mostren a temps real les notificacions que té l'usuari, en aquest cas les llistes de favorits que hagi confeccionat en qualsevol moment.

Figura 19: Pantalla de notificacions



Aquesta és la pantalla que mostra les llistes corresponents que ha creat l'usuari. Amb el botó de dalt de tot a la dreta, es pot seleccionar un subconjunt de les llistes existents, i esborrar les que estiguin seleccionades.

A més a més, es poden cercar pel nom de la llista utilitzant el botó de la lupa que està just a la part superior de la imatge.

Figura 20: Pantalla de les meves llistes

Aquesta és la pantalla amb la qual es poden veure els diferents esdeveniments que pertanyen a una llista creada per l'usuari. Igual que en la pantalla anterior, es poden cercar pel nom de l'esdeveniment i es pot seleccionar un subconjunt per tal d'esborrar els elements que interessin.

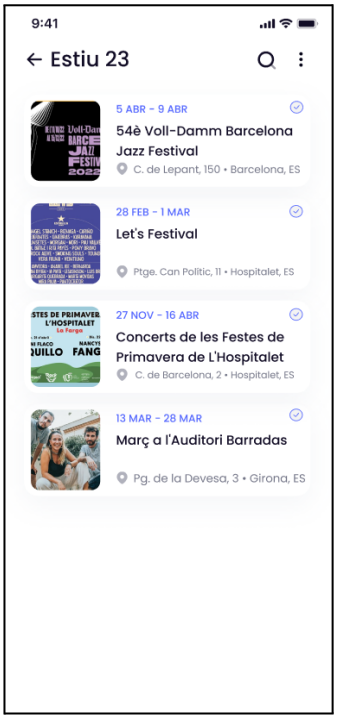


Figura 21: Pantalla d'esdeveniments en llista

6. Architectural physical diagram

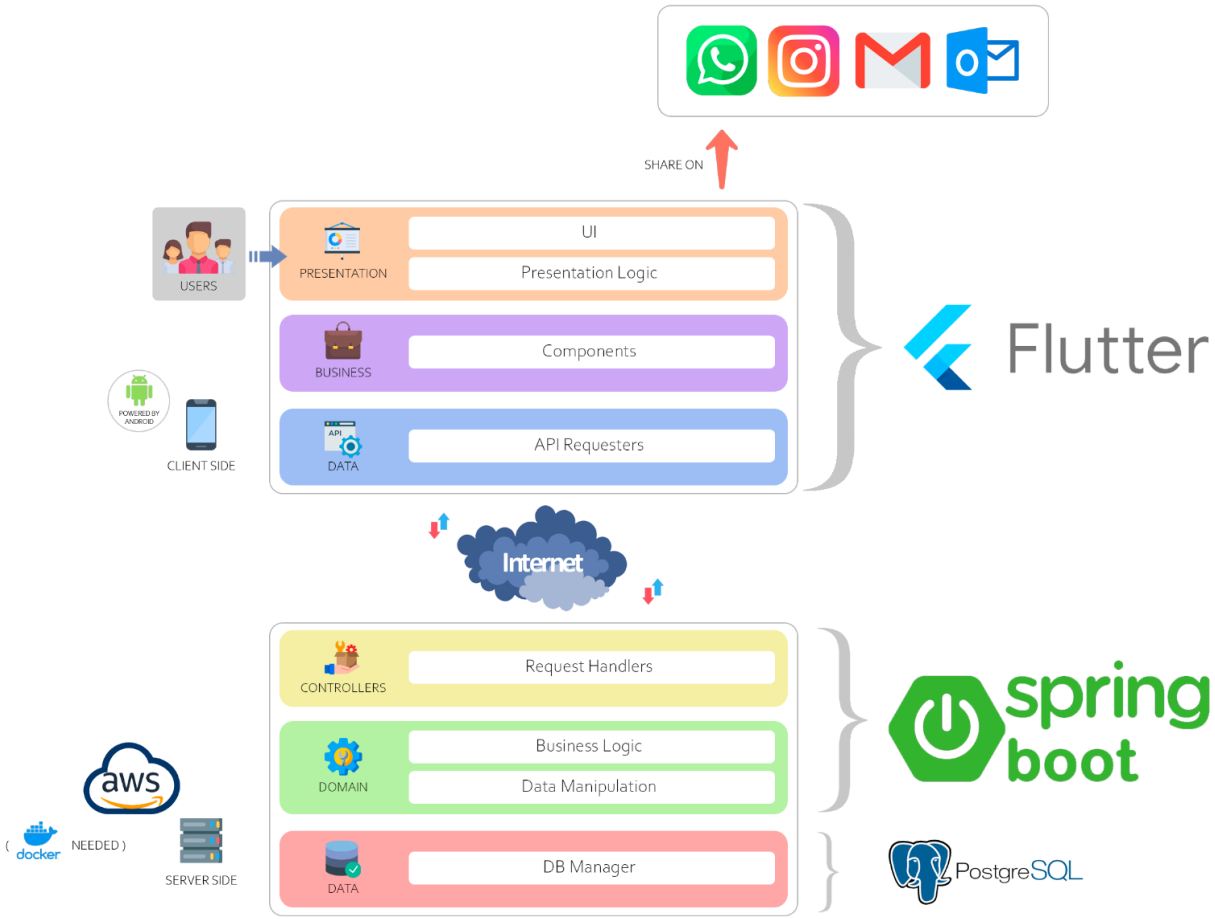


Figura 22: Diagrama d'arquitectura física

Hem decidit dividir l'aplicació en un *backend* i un *frontend*. L'arquitectura de cadascun estarà dividida en tres capes, i interaccionaran només amb les seves adjacents (i.e. la capa de domini del *backend* amb la capa de dades i els controladors).

7.1. Frontend

Per fer l'app nativa dissenyarem i implementarem un *frontend* que sigui capaç d'obtenir dades de la nostra API, muntar els components que renderitzarem a la interfície d'usuari i fer el *display* en concret. El dividim en tres capes:

- Presentation Layer: Encarregada de mostrar la interfície d'usuari (UI) i de renderitzar les vistes i els components, aquesta capa només mostra la informació de manera estructurada.

- Business Layer: La seva funcionalitat és definir els components que utilitzarem a la UI, és a dir, la seva forma, funció, i informació que mostraran.

- Data Layer: Ens servirà per a fer peticions a la nostra API *Rest*, i enviarem les dades a la Business Layer perquè els components puguin contenir la informació pertinent.

El *frontend* constituirà una aplicació per a Android i estarà programada utilitzant el *framework* Flutter. Hem estat pensant davant de les diferentes opcions que teníem: Flutter, Compose, React Native, Xamarin, Ionic i Cordova. Les últimes dues les vam descartar perquè són *frameworks* híbrids que s'utilitzen per a fer aplicacions web o *Progressive Web Apps*. Hem escollit Flutter, ja que React Native considerem que és molt poc estructurat i depèn molt de llibreries externes, Xamarin quasi no s'utilitza i Compose només servia per a Android. Les raons de pes de per què ens vam decidir per Flutter són les següents:

- És un framework molt estructurat, i permet fer quasi qualsevol cosa a Android, ja que el propietari és Google. Considerem que s'ajusta molt bé a les nostres necessitats i ens permetrà desenvolupar el nostre *frontend* de forma senzilla.

- És molt popular, i ens interessa aprendre una tecnologia que s'utilitza per tot el món. A més, es programa amb Dart, un llenguatge *open source* desenvolupat per Google que rep influència de *C#*, *Java* i *JavaScript*.

- Permet desenvolupar per a Android i per a iOS. Malgrat que només tinguem pensat fer la nostra app per a Android, la veritat és que al principi vam considerar fer-la també per a iOS. Amb tot, ens agrada la idea de poder ampliar el nostre *target* en un futur i vam considerar aquesta possibilitat com a un avantatge.

A sobre surten les icones de Whatsapp, Instagram, Gmail i Outlook. Ja que podrem compartir esdeveniments mitjançant aquestes aplicacions, hem considerat afegir-les al diagrama de l'arquitectura.

7.2. Backend

L'aplicació que constituirà el *Backend* tindrà la nostra base de dades amb el *dataset* que ens proporciona la Generalitat de Catalunya, la gestió i neteja de les dades i la nostra *API Rest*, la qual servirà de manera directa al *Frontend* i a altre grup de treball del nostre subgrup de PES. La dividirem en tres capes:

- *Controller Layer*: S'encarregarà de fer d'handler per a les peticions a l'API *Rest* de banda del *frontend*. Transformarà les dades a format *.json* i les enviarà, i delegarà responsabilitats a la Domain Layer.

- *Domain Layer*: Aquesta capa té com a funció allotjar tota la lògica de negoci i tractament de les dades del nostre sistema. Cridarà a la Data Layer del *backend* per a guardar informació i obtenir-la de la BD.

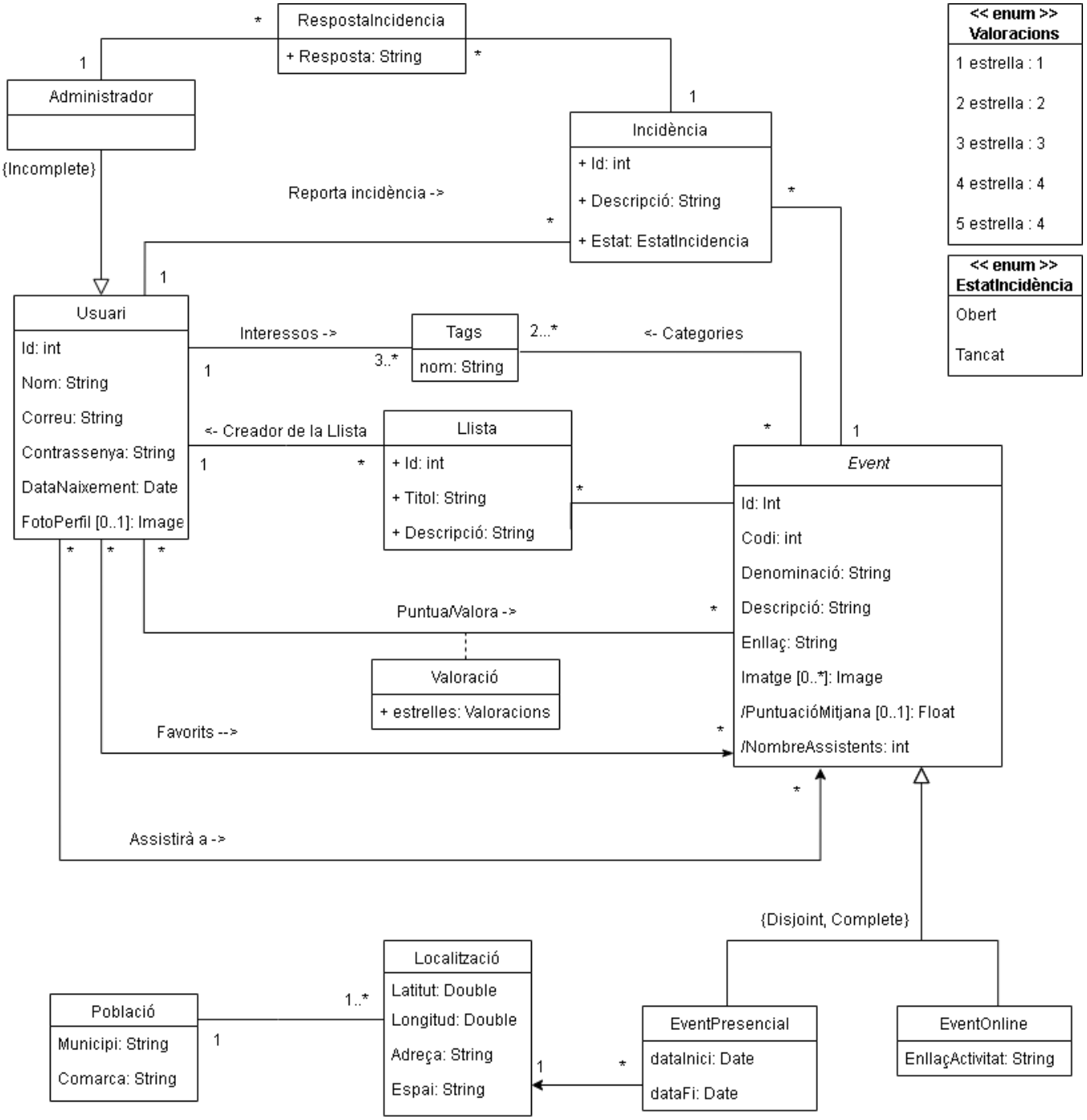
- *Data Layer*: Contindrà funcions que interactuïn directament amb la BD, fent *Inserts*, *Updates*, i *Deletes* a les nostres taules.

Hem decidit allotjar el *Backend* i la base de dades a un mateix servei de *hosting*. Concretament, ens hem decantat finalment per Amazon Web Services. Com a opcions hem considerat Virtech, Heroku, AWS, i Render. Vam descartar Virtech perquè considerem un desavantatge treballar sempre amb VPN, respecte a Heroku ho vam descartar perquè des de fa uns mesos és de pagament i Render perquè malgrat que hi ha molta documentació, ens ofereix menys recursos de manera gratuïta.

Pel que fa al *framework*, farem servir Spring Boot. Com a opcions inicials ens van cridar l'atenció Django, Ruby on Rails i Spring Boot. Vam decidir Spring Boot perquè és en un llenguatge molt semblant a Java (utilitzarem Kotlin), i considerem que Python (utilitzat per Django) i Ruby (utilitzat per Ruby on Rails) no s'adeqüen bé als requisits no funcionals que volem que tingui la nostra aplicació, i no ens permeten aplicar segons quins patrons i principis de disseny que tenim present implementar de cara al començament del primer sprint.

7. UML class diagram

A continuació tenim el diagrama general de clases en UML amb les seves restriccions textuals. Aquest diagrama mostra tots els elements del nostre sistema i les relacions que existeixen entre elles.



RESTRICCIONS TEXTUALS

1.Claus Primaries:

Usuari (Id); **Tags** (nom); **Event** (id); **Llista** (id + Usuari:id); **Incidencia** (id); **Localització**(Latitud + Longitud + Espai); **Població** (Municipi + Comarca)

2. La dataInici d'un EventPresencial no pot ser posteriorr a la dataFi

3. La dataNaixement d'un Usuari no pot ser posterior a la data actual

4. Un Usuari no pot Puntuar/Valorar un Event al que no Assistirà

5. L'atribut PuntuacioMitjana d'un Event es calcula segons la mitjana d'estrelles de Valoració

6. L'atribut NombreAssitens d'un Event és el nombre d'Usuaris que *assistiran* al Event

Figura 23: Diagrama de classes UML

8. Description of working methodology and technology

8.1. Visió General

Per a organitzar-nos hem decidit seguir la metodologia agile *Scrum*. Seguint aquesta metodologia, hem dividit el projecte en dues fases d'*Inception* per definir, entre altres coses, les diferents funcionalitats que tindrà la nostra aplicació i tres *Sprints* de tres setmanes. Cadascun dels Sprints té associat un membre com a '*Sprint master*' que serà l'encarregat de dirigir les retrospectives i reunions setmanals, verificar les tasques del Project Record Track, entre altres responsabilitats. També tenim a un membre dedicat a relacionar-se amb la resta de grups de cara al servei que haurem d'integrar i oferir a un dels grups.

Actualment, tenim dedicats 3 dies per a reunir-nos, les classes de dimarts i dijous, i els diumenges a la tarda per tal de plantejar les tasques de la setmana vinent o per fer les retrospectives. Com a grup ens comuniquem mitjançant un grup de Whatsapp per notificar-nos sobre canvis recents o importants que afectin el grup en general i un servidor de Discord amb canals de veu per fer les reunions, canals de text per enviar-nos recursos que poden ser d'utilitat (#link), canals dedicats per les reunions o per enviar-nos referències (mockups, aplicacions semblants, etc.) que ens puguin ajudar.

Finalment, per a les històries d'usuari estem fent servir Taiga, allà tenim definits els criteris d'acceptació de cada història, les tasques associades i els *story points* que té assignats, desglossat en Front, Back, Testing i UX.

En els següents apartats, entrarem més en detall sobre les tecnologies i metodologies escollides per al projecte.

8.2. Gestió Projecte

Com s'ha esmentat anteriorment, la principal eina de gestió que fem servir és Taiga, la qual ens permet definir històries d'usuari i agrupar-les en èpiques. A la descripció de les històries és on tenim definit els diferents criteris d'acceptació de cada història i el seu pes en *story points* desglossat en diferents apartats.

Aquests story points no són necessàriament fixes i som conscients que hi ha tasques que a priori haurem puntuat malament, sigui a l'alta o a la baixa. La idea es anar refinant aquesta puntuació després de cada sprint amb les reunions retrospectives, per tal de distribuir-nos la feina de forma equitativa.

Abans d'assignar els story points a cada tasca vam concretar que un story point equivaldria a una hora de treball, aquesta hora no inclou el temps de formació directament, però, sí que té en compte la complexitat de la tasca i, per tant, la necessitat de formació.

Amb la definició del valor d'un story point, i amb les històries d'usuari definides, vam realitzar un planning poker amb tot l'equip per tal de decidir el valor final de cada tasca. Per acabar de decidir, els membres amb els valors més alts i més petits de cada tasca van explicar el perquè havien decidit aquests valors, i conjuntament vam decidir la puntuació final, tenint en compte la mitja que ens va donar el planning poker per a cada tasca.

A part del Taiga, també estem fent servir el Trello per poder assignar tasques que no tenen una relació directa amb les històries d'usuari. Això ens permet veure quines tasques queden pendents fora de l'àmbit del desenvolupament, en particular en aquestes primeres fases d'inception.

Finalment, hem acordat el *Definition of Done* (*DoD*) i el cicle de vida de cada tasca.
El DoD de cada història l'hem definit en 4 fases, la primera consisteix en la redacció dels criteris d'acceptació i les tasques que formen part de la història, a continuació, tenim la implementació de la història d'usuari, evidentment, sense errors.

Algunes de les històries d'usuari, les que creiem més importants o crítiques tenen assignades unes etiquetes indicant quin tipus de test s'han de crear per tal de garantir que no introduïm nous problemes a mesura que anem avançant el projecte. La granularitat dels tests està explicada amb més detall més endavant. Independentment de si la història té tests o no, per passar a la següent fase, és indispensable que passin els tests d'integració abans de documentar el codi, donant així per acabada la història.

Pel que fa al cicle d'història, fem servir el sistema que determina el Taiga, la qual pot estar en un de 4 estats:

- [Ready] L'estat per defecte de les històries d'un *sprint*, aquestes històries han d'estar redactades prèviament.
- [In Progress] Un cop iniciat un sprint, quan la història estigui assignada a un membre de l'equip.
- [Ready For Testing] La història està implementada i el codi compila sense errors, però les noves funcionalitats no han estat testejades. En aquest cas, els tests poden fer referència als tests d'integració o als tests unitaris que s'esmenten més endavant.
- [Tested - Undocumented] La història d'usuari ha passat els tests d'integració, però no ha estat documentada. Arribat aquest punt, el codi de la història es pot posar en producció.
- [Done] La història d'usuari està implementada, sense errors, ha passat tots els tests i està documentada.

8.3. Gestió versions

Per tal de gestionar les versions del codi, hem decidit que farem servir GitHub com a servidor a on guardar els repositoris. Hem creat l'organització PES-Agenda-Cultural, aquesta organització té dos repositoris, un per al codi del frontend i l'altre per al codi del backend.

Per facilitar el procés de generar releases i la resolució de possibles conflictes a l'hora de fer merge entre branques, hem decidit definir dos nous rols, Lead Backend i Lead Frontend. Les seves responsabilitats respecte a la gestió de versions consistirà principalment a solucionar conflictes a l'hora de fer merge entre les branques i mantenir els convenis estipulats més endavant.

La gestió dels dos repositoris seguirà la mateixa metodologia de gestió de versió, el model de control de branques anomenat **Gitflow**. Aquest model consisteix en dues branques principals, una branca main amb el codi de **producció** i una segona branca de **desenvolupament** amb el codi estable més recent.

La branca de **principal (main)** conté el codi preparat per producció i serveix com a branca històrica on cada commit representa una nova versió, principal o intermèdia per arreglar bugs que no hàgim captat durant el desenvolupament. En el cas del back end, aquesta és la branca que clonarà el contenedor de Docker. Qualsevol canvi efectuat en aquesta branca, s'haurà de replicar a la branca de dev (merge main -> dev), principalment en el cas que es trobés un bug un cop fet el merge.

La branca de **desenvolupament (dev)** mantindrà el codi estable més recent. Aquesta branca serà la branca de la qual es bifurquen les altres branques que explicarem a continuació. Un cop finalitzat un sprint, el Lead Backend i Lead Front End seran els encarregats de fer el merge de **dev -> main** dels seus repositoris respectius.

Seguint el model de Gitflow, farem servir branques **'feature'** per a cada història d'usuari. Tot el desenvolupament relacionat a la història d'usuari i les seves tasques es contindrà dins de la seva pròpia branca, incloent-hi els tests que facin falta.

El nom de les noves branques seguira el següent conveni:

- feature/<# num_història>:<nom_historia>
  Exemple: feature/#41:Iniciar_Sessio

On *num_historia* i *nom_historia* fan referència al número al títol que té la història d'usuari al Taiga respectivament.

Aquestes branques es crearan a partir de la branca *dev* i un cop acabades es tornaran a fer merge a la branca *dev* **(feature -> dev)**.

Tot i que les històries d'usuari han estat redactades per tal de maximitzar la seva independència, si es donés el cas que una història necessites una funcionalitat present en una altra branca de *feature*, el procediment consistirà a fer merge de la branca *feature*/xx de la qual depèn a *dev* **(feature -> dev)** i de dev merge a la *feature*/yy que tenia la dependència **(dev -> feature)**. Amb aquest procediment la idea és evitar costi el que costi el merge entre branques de tipus feature per tal d'evitar conflictes o discrepàncies al codi més endavant.

Pel que fa a la correcció d'errors, farem servir una metodologia semblant a les branques '*feature*', una nova branca **'issue'** per a cada error que trobem. A diferència de les històries d'usuari, farem servir l'apartat d'issues del mateix Github per especificar els detalls de cada bug. A diferència de les branques '*feature*' aquestes noves branques es poden crear a partir de **main** si es trobés l'error després de fer el release, en aquest cas, un cop solucionat el problema, la branca tornaria a ajuntar-se amb main. Aquestes branques han de contenir els canvis mínims per solucionar el problema i a poder ser, un nou test per evitar tornar a introduir-lo més endavant.

El nom d'aquestes branques farà servir la següent convenció:

- issue/<#num_isssue>:<titol>
  Exemple: issue/#123:Error_Login_Google

· Commits

Finalment, per tal de millorar la llegibilitat, a la hora de fer els commits el missatge tindrà el número de la tasca que s'està resolent seguit d'una petita descripció explicant els canvis fets. La estructura per tant seria tal que:

- <#num_tasca>:<descripció>

Exemple: #81: Afegits botons per filtrar per ubicació

8.4. Comunicació d'equip

Amb l'objectiu de tenir la millor comunicació possible, tenim reunions presencials a classe els dimarts i dijous de 8:00 a 10:00. A més, com s'ha esmentat anteriorment, tenim dos canals de comunicació. Un grup de Whatsapp per notificar a tot l'equip sobre novetats, canvis, enviar petits missatges o concretar hores de reunió i un servidor de Discord amb diferents canals de text especialitzats per enviar enllaços d'interès, parlar de Backend, Frontend, fer sessions de pluja d'idees sense haver de notificar a tot l'equip.

A Discord també tenim canals de veu on podem compartir la pantalla i fer les reunions que generalment són els diumenges. D'aquesta manera afegim una reunió extra perquè no hi hagi tanta distància entre la reunió de dijous i la del dimarts de la setmana següent.

8.5. Frameworks i llenguatges

Com s'ha esmentat anteriorment, el nostre codi està dividit en dues parts, el *frontend* i el *backend*, aquests tenen els seus llenguatges, frameworks i repositoris diferents que s'expliquen a continuació.

Per la part de Backend de la nostra aplicació utilitzarem el framework Spring Boot.
Hi ha diversos motius pels quals hem escollit Spring Boot:

- **Flexibilitat:** Spring inclou un set d'extensions i llibreries per ajudar a desenvolupar qualsevol tipus d'aplicació.
- **Rapidesa:** Spring facilita entre altres coses la creació del projecte permetent inicialitzar un nou projecte de Spring en segons amb l'ajuda del 'Spring Initializer'.
- **Suport:** Spring té una comunitat global amb tutorials des de principiants fins a professionals. En conseqüència, no serà difícil buscar material d'ajuda.
- Spring està a tot arreu. Spring ha estat emprat per companyies com Amazon, Google o Microsoft i, per tant, és un al·licient a aprendre a fer-lo servir.

Conjuntament amb Spring Boot farem servir Kotlin com a llenguatge de programació. Kotlin és un llenguatge similar a Java amb el qual tot l'equip està familiaritzat, però està més enfocat al desenvolupament d'aplicacions Android. Alguns avantatges de Kotlin són la simplicitat, la flexibilitat i la reducció del anomenat *boilerplate* el que augmenta la productivitat per exemple a l'hora de declarar classes, a Kotlin són unes poques línies i els setters i getters es creen automàticament.

Per la part del frontend farem servir el framework desenvolupat per Google, Flutter. L'aplicació mòbil estarà programada per a dispositius Android tot i que Flutter com a framework permet crear aplicacions multiplataforma, he decidit que no donarem suport per a dispositius iOS, ja que per programar-los és indispensable tenir un Mac i cap membre de l'equip en té.

Respecte al perquè de Flutter, es redueix a dos motius principals. Per una banda, vam veure que en l'àmbit del desenvolupament mòbil, Flutter és un dels llenguatges més populars i més demanats per diferents empreses i anuncis de feina, i com a grup vam concretar que posats a aprendre un nou framework i llenguatge, estaria bé aprendre'n un que ens obri

més portes en un futur. Tot i ser prou nou com a framework, la seva documentació és robusta i hem trobat molts recursos per ajudar-nos a aprendre'l al llarg del projecte, entre aquests recursos destaca un curs introductori gratuït a YouTube de més de 37h de duració.

Una altra gran raó per la qual vam escollir Flutter és el llenguatge que fa servir, Dart. També va estar desenvolupat per Google, originalment com a alternativa a JavaScript pel desenvolupament de pàgines web, el llenguatge no va tenir molt d'èxit fins a la creació de Flutter uns anys més tard.
Tot i que cap membre de l'equip d'experiència prèvia amb Dart, un gran avantatge del llenguatge és la seva sintaxi, la qual és molt semblant a Java i JavaScript, i finalment, tot i ser relativament nou com a llenguatge, Google ofereix molta documentació sobre com funciona Dart, les seves similituds a Java i algun dels aspectes únics en relació amb el desenvolupament per a Android.

8.6. IDEs

Per facilitar el desenvolupament, hem escollit dos IDEs (entorns de desenvolupament integrat) que ens proporcionen plugins per ressaltar errors en el codi, breakpoints per debugar, snippets per agilitzar el desenvolupament, i la més important, un entorn consistent per als desenvolupadors. D'aquesta manera podem evitar problemes amb versions inconsistents de llibreries o altres dependències. Un altre avantatge del IDEs escollits és el fet que contenen eines per a compilar i executar el codi amb les seves dependències sense haver de fer tota la configuració de forma manual.

Per al frontend, en estar fent una aplicació nativa per a Android, hem decidit fer servir Android Studio com a IDE, el qual té un plugin per desenvolupar aplicacions amb Dart.

Per al backend, farem servir l'IDE IntelliJ de JetBrains. Com ja hem estipulat anteriorment, aquest IDE ens permet ancorar una versió específica del JDK al projecte, evitat així possibles problemes amb versions antigues presents en els ordinadors dels desenvolupadors.

8.7. Gestió de qualitat

Per a provar la qualitat del nostre *software* utilitzarem les eines que ens ofereixen els nostres IDE's.

Per la part de Frontend utilitzarem Android Studio que inclou un emulador d'Android. Aquest emulador ens ofereix coses com comprovar la interfície a diferents dispositius amb diferent dimensió de pantalla. També podem falsificar les coordenades del GPS per comprovar les funcions de geolocalització, rotar la pantalla... d'aquesta manera podem comprovar que la nostra aplicació funcioni correctament a tots els dispositius. Android Studio també ofereix la possibilitat de connectar un dispositiu per executar la nostra aplicació. Això permet comprovar coses que no permet l'emulació com l'experiència d'usuari i la velocitat de resposta. Com també podem utilitzar l'eina 'scrcpy' per mostrar i controlar el nostre dispositiu Android a través del nostre ordinador.

També usarem linters per ajudar a detectar errors de programació comuns, estilístics i per estandarditzar el codi, en particular al Frontend, ja que Dart té un linter integrat.

8.8. Gestió de proves

Per al nostre projecte farem servir tests unitaris i tests d'integració. De cara al primer sprint, l'execució dels tests la farem de forma manual, però estem buscant i informant-nos sobre formes d'automatitzar l'execució dels tests (en particular els d'integració) cada cop que es fa un commit a les branques ***main*** o ***dev***.

En el cas de *Flutter*, també ofereix l'opció de testejar components o *widgets* de forma aïllada, molt útil per tal de poder maximitzar la reusabilitat del codi.

Finalment, a l'hora de resoldre bugs, hem decidit que farem servir la pràctica de TDD (*test driven development*), és a dir, començarem escrivint els tests necessaris per determinar que el bug s'ha resolt i a partir d'aquests tests resoldrem l'error en qüestió. La idea és que amb aquesta metodologia, podem augmentar el nombre de tests de forma orgànica sense haver de pensar des d'un principi tots els casos extrems per a cada funcionalitat.

8.9. Gestió de configuracions

Per tal de fer els desplegaments a AWS farem servir Github Actions. Aquest servei de Github es pot automatitzar per tal que cada vegada que hi hagi un push s'executin els tests de CI/CD i es pugui desplegar l'aplicació amb més seguretat.

També farem ús d'AWS CodeDeploy. Aquest és un servei que automatitza els desplegaments de software. D'aquesta manera s'elimina la necessitat de fer operacions manuals que són propenses a errors i s'evita perdre temps innecessari.

8.10. Interacció amb altres grups

Pel que fa a la interacció amb els altres equips per als serveis que haurem d'implementar i oferir entre nosaltres l'estem gestionant a través de dos canals. Per una banda, el nostre responsable aprofita les hores dels dimarts i els dijous per anar als grups i preguntar al grup sencer quines funcionalitats estan buscant o en el cas del servei que hem d'integrar, debatre i explicar els nostres requisits amb relació al que ens poden oferir.

Per l'altra banda, també disposem d'un grup de whatsapp amb tots els membres dels dos equips per tal de facilitar la comunicació durant la resta de la setmana. Els missatges d'aquest grup també serveixen com a punt de referència en sobre quines funcionalitats s'han pactat a l'hora d'implementar el servei i per resoldre qualsevol problema o incongruència que hi pogués haver.

8.11. Convencions de codi

Per tal de facilitar la llegibilitat del codi i mantenir un cert nivell de coherencia, hem establert una sèrie de convencion a l'hora d'escriure el codi tant del Frontend com el del Backend, la primera norma és que el codi (funcions, variables, classes, etc) estarà escrit en anglès, l'única excepció seran els comentaris els quals poden estar escrits en l'idioma que prefereix l'autor, però procurarem que aquest sigui en català.

8.11.1. Variables

Les variables estaran majoritàriament escrites en *camelCase* i s'intentarà que siguin tan descriptives com sigui possible evitant abreviacions i lletres úniques. Les úniques

excepcions a aquesta norma seran els iteradors de bucles o índexs per a arrays. Per les assignacions, deixarem un espai entre el nom, el símbol '=' i l'assignació.

El nom de les variables constants s'escriuran en majúscules amb '_' indicant els espais (en el cas de ser més d'una paraula).

EX: **String** API_ROOT_URL = 'www.exemple.com/api/';

8.11.2. Funcions i mètodes

Respecte a les funcions, per norma general serà semblant al de les variables, camelCase i amb noms en anglès els més descriptius possibles, i en cas de fer una acció, el nom de la funció haurà de començar amb l'acció (verb).

Per exemple: *getAllEvents*();

Per ajudar-nos a entendre el codi de cada company, procurarem documentar les funcions de la següent manera:

- **Descripció** breu del que fa la funció o perquè existeix/s'utilitza.
- <**nom paràmetre**> : Si no fos prou evident, una petita descripció i/o consideracions a tenir en compte (si el valor es modifica, si pot ser Null, rang de valors min/max, etc)
- <**excepcions**> : Perquè salten i si són crítiques o no.
- **Resultat**: Descripció del quin és el resultat i consideracions a tenir en compte (semblant als paràmetres, si el resultat pot ser Null, si té resultats que denominen un estat d'error (ex: un -1), etc...

Les funcions més trivials com els getters o els setters no caldrà documentar-les. També cal remarcar que no serem molt estrictes a l'hora de documentar el codi, ja que considerem que en aquest context, és una eina per ajudar-nos a entendre el codi dels altres, i en el cas que tothom entengués una funció/mètode/classe no caldrà necessàriament documentar-la. Tot i això, la idea és documentar (lleugerament) el màxim de codi possible.

Sobre l'estilització dels mètodes, bucles o blocs condicionals (o qualsevol element que creï una indentació denominada amb '{ }'), els dos curly braces estaran en línies noves, l'única excepció serà la seqüència if - else (if), la qual podrà estar en la mateixa línia que el bràquet que tanca el bloc anterior i en cas de ser un else, el bràquet que obre el bloc de l'else també podrà estar en línia:

EX if - else:

```
if (a == b)
{
        //Codi
} else if (a < b)
{
        //Codi
} else {
        // Codi
}
```

EX for:

```
for (i = 0; i < 10; i++)
{
        // Codi
```

}

8.11.3. Classes

El nom de les classes s'escriurà fent servir UpperCamelCase on la primera lletra de cada paraula (inclosa la primera) sigui majúscula. El nom de la classe haurà de ser el mateix que el del fitxer que el conté, excepte en el cas que un fitxer contingui més d'una classe on el nom del fitxer serà el de la classe més important que contingui, o si les classes formen part d'una jerarquia, el nom del fitxer serà el de la superclasse. Si la classe existeix com a element d'un patró s'intentarà que el nom reflecteixi el patró que s'està emprant. EX: **class** *UserFactory*

L'ordre dels elements de les classes serà el següent:

1. Propietats (variables d'una classe)
2. Constructora
3. Destructora (si calgués crear-ne una específica)
4. Getters
5. Setters
6. Altres mètodes públics
7. Altres mètodes privats

En el cas del backend, les classes relacionades amb els endpoints de la API, seguiran el següent ordre:

1. Get
2. Post
3. Put
4. Delete

El nom de les propietats (variables) d'una classe que siguin privades aniran marcades amb un '_' al principi.

- Ex: **int** _userId;

En canvi, les propietats o mètodes que siguin públics faran servir la convenció de *UpperCamelCase* per tal de fer més fàcil la distinció entre element públics i privats d'una classe.

- Ex: **User** GetUserById(int id);

Els mètodes privats o protected faran servir el mateix conveni que s'ha esmentat anteriorment *camelCase*

Sobre la documentació, cada classe haurà de tenir una petita descripció explicant el que representa i/o la seva utilitat, per a les propietats públiques també caldrà escriure un petit comentari a sobre de la variable explicant.

8.11.4. Fitxers i directoris

Sobre els fitxers (que no són classes), farem servir camelCase i pels directoris UpperCamelCase

Ex: UnDirectori/UnAltreDirectori/apiKeys.json

Per a qualsevol altre element o en cas que no s'hagi estipulat en aquest apartat farem servir camelCase per tal de mantenir la consistència.

8.12. Gestió de bases de dades

Per la part de la base de dades farem servir PostreSQL. La desplegarem a Amazon Web Services de la mateixa manera que el backend. Hem escollit PostgreSQL perquè és un servei amb el qual tots estem familiaritzats. A l'hora de configurar la base de dades a AWS seleccionarem les opcions més bàsiques que són les que estan incloses de manera gratuïta. També seleccionarem l'opció d'accés públic per tal de poder accedir des dels nostres ordinadors i no utilitzar cap mena de VPN.

9. Consumer and producer of services

Durant aquesta segona fase de l'incepció, hem realitzat una primera toma de contacte en l'àmbit de negoci entre els diferents equips per tal de conèixer de primera mà quins projectes tenen pensats i de quina manera tindran impacte en el nostre producte.

Primerament, ens van comentar que havíem de rebre i donar funcionalitats destacables amb la resta de grups, per tant, vam estudiar quines opcions eren les més interessants, entre les que proporcionaven els companys, de cara a afegir-les a la nostra aplicació.

Mobilitat sostenible

D'aquesta manera, amb l'objectiu de tenir el millor producte possible ens vam centrar en quines propostes ens interessaven enviar i quines rebre, i així ho vam fer. Com a la nostra aplicació el mapa serà una de les funcionalitats més principals, vam estar negociant amb el grup de Mobilitat Sostenible i es va arribar a la conclusió que aquest grup ens oferirà la possibilitat d'integrar els punts de càrrega al nostre mapa interactiu. Gràcies a això la nostra aplicació tindrà més valor, ja que a més de poder cercar esdeveniments, l'usuari podrà tenir coneixement dels punts de càrrega que estan al voltant del seu entorn.

Així doncs, la funcionalitat que rebem per part del grup de Mobilitat Sostenible és la següent:

- Donada una ubicació concreta, en el mapa es podran mostrar les estacions de càrrega per a vehicles elèctrics a Catalunya.

Licitacions i adjudicacions en curs

En relació amb la negociació amb el grup de licitacions i adjudicacions en curs, tot va començar comentant amb la gestora dels serveis de l'altra grup quines funcionalitats tenien present i els hi podien ser d'utilitat. La primera idea que va sorgir va ser relacionada amb el mapa, però com és una funcionalitat fonamental pels dos productes, es va descartar ràpidament. Addicionalment, nosaltres vam aportar la idea de proporcionar el calendari per tal de poder observar segons alguns filtres de dates les contractacions públiques de Catalunya.

Com a conclusió, la funcionalitat que proporcionem al grup de licitacions és la següent:

- Donat un conjunt de dades, oferir la funcionalitat de calendari perquè es pugui adaptar a les crides de la seva API.

10. Adreces d'interès

[1] Integrating with GitHub Actions – CI/CD pipeline to deploy a Web App to Amazon EC2

[2] Planning poker online | Planning poker CultureFinder

[3] Google Calendar API overview | Google Developers

[4] Create and Connect to a PostgreSQL Database with Amazon RDS

[5] Quality Assurance Toolkit: Using Android Studio IDE

[6] ¿Qué es un linter en programación? | KeepCoding Tech School

[7] Servicio de Cloud docente de la FIB: Virtech

[8] The Six Most Popular Cross-Platform App Development Frameworks | Kotlin Documentation

[9] Top 5 Best Frameworks for Android App Development in 2022 (tateeda.com)

[10] share_plus | Flutter Package (pub.dev)