



## Culture finder

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

| Cognom    | Nom    | Responsable                          | UPC e-mail                                | Taiga               | GDrive                                    | Github              |
|-----------|--------|--------------------------------------|---|---------------------|---|---------------------|
| Morón     | Daniel | Management services of               | 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           |

# Índex

|  |           |
|--|-----------|
| <b>1. Requisites</b>                                   | <b>4</b>  |
| <b>1.1. Concepció general del projecte</b>             | <b>4</b>  |
| <b>1.2. “NOT” list</b>                                 | <b>4</b>  |
| <b>1.3. Model conceptual de les dades</b>              | <b>7</b>  |
| <b>1.2. Resum del backlog del producte al document</b> | <b>8</b>  |
| 1.2.1. Sprint 1 Backlog                                | 8         |
| 1.2.2. Sprint 2 Backlog                                | 11        |
| 1.2.3. Sprint 3 Backlog                                | 16        |
| <b>1.4. Requisites no funcionals</b>                   | <b>22</b> |
| 1.5.1 Requisites de rendiment                          | 22        |
| 1.5.2 Requisites de preservació i suport               | 22        |
| 1.5.3 Requisites de capacitat d'ús i humanitat         | 23        |
| 1.5.3 Requisites de seguretat                          | 24        |
| <b>1.5. Tractament dels aspectes transversals</b>      | <b>26</b> |
| <b>1.6. Serveis de tercers</b>                         | <b>30</b> |
| 1.6.1 Repartiment de funcionalitats entre els equips   | 30        |
| 1.6.2 Comunicació amb la resta dels equips             | 31        |
| <b>2. Metodologia</b>                                  | <b>31</b> |
| 2.1. Visió General                                     | 31        |
| 2.1.2 Divisió de l'equip en Backend i Frontend         | 31        |
| 2.1.2 Scrum  | 32        |
| 2.2. Gestió Projecte                                   | 33        |
| 2.3. Gestió versions                                   | 35        |
| 2.4. Comunicació d'equip CultureFinder                 | 37        |
| 2.5. Frameworks i llenguatges                          | 37        |
| 2.6. Llibreries  | 39        |
| 2.6.1 Mapstruct  | 39        |
| 2.6.2 Jackson  | 40        |
| 2.6.3 Liquibase  | 40        |
| 2.6.4 Lombok   | 41        |
| 2.6.5 Openapi  | 41        |
| 2.6.6 Json Web Token                                   | 42        |
| 2.6.7 JSoup  | 43        |
| 2.6.8 Firebase   | 43        |
| 2.6.9 Shared Preferences                               | 43        |
| 2.6.10 Table Calendar                                  | 43        |
| 2.6.11 Riverpod  | 43        |
| 2.6.12 Google Maps for Flutter                         | 44        |
| 2.6.13 Google Maps Cluster                             | 44        |
| 2.6.14 Geolocator                                      | 44        |
| 2.6.15 Permission Handler                              | 44        |
| 2.6.16 Google Sign In                                  | 44        |

|  |           |
|--|-----------|
| 2.6.17 Image Picker                            | 44        |
| 2.7. IDEs                                      | 45        |
| 2.8. Gestió de qualitat                        | 45        |
| 2.9. Interacció amb altres grups               | 46        |
| 2.10. Convencions de codi                      | 46        |
| 2.10.1. Variables                              | 46        |
| 2.10.2. Funcions i mètodes                     | 47        |
| 2.10.3. Classes                                | 47        |
| 2.10.4. Fitxers i directoris                   | 48        |
| 2.11. Gestió de bases de dades                 | 48        |
| 2.12. Gestió de bugs                           | 49        |
| <b>3. Descripció tècnica</b>                   | <b>50</b> |
| <b>3.1. Concepte de l'arquitectura</b>         | <b>50</b> |
| <b>3.1.1. Arquitectura física</b>              | <b>50</b> |
| <b>3.1.2. Patrons d'arquitectura aplicats</b>  | <b>51</b> |
| <b>3.2. Capa de domini</b>                     | <b>52</b> |
| <b>3.2.1. Patrons de dissenys aplicats</b>     | <b>52</b> |
| <b>3.3. Diagrama de la base de dades (UML)</b> | <b>55</b> |
| <b>3.4. Llistat de les tecnologies</b>         | <b>55</b> |
| <b>3.4.1. Visió general dels servidors</b>     | <b>55</b> |
| <b>3.4.2. Configuració dels servidors</b>      | <b>56</b> |
| <b>3.4.3. Bases de dades</b>                   | <b>56</b> |
| 3.4.3.1. Backend                               | 56        |
| 3.4.3.2. Admin web                             | 56        |
| 3.4.3.3. Frontend                              | 56        |
| <b>3.4.4. Llenguatges utilitzats</b>           | <b>56</b> |
| 3.4.4.1. SQL (PostgreSQL)                      | 56        |
| 3.4.4.2. Dart (Flutter)                        | 57        |
| 3.4.4.3. Java (Spring boot)                    | 57        |
| 3.4.4.4. Kotlin(Gradle)                        | 58        |
| 3.4.4.5. JavaScript (NodeJS)                   | 58        |
| <b>3.5. APIs</b>                               | <b>59</b> |
| <b>3.5.1. API del nostre producte</b>          | <b>59</b> |
| 3.5.1.1. /users                                | 59        |
| 3.5.1.2. /lists                                | 60        |
| 3.5.1.3. /incidents                            | 61        |
| 3.5.1.4. /events                               | 62        |
| 3.5.1.5. /assistances                          | 64        |
| 3.5.1.6. Gestió de la seguretat                | 65        |
| <b>3.5.2. APIs externes</b>                    | <b>65</b> |
| 3.5.2.1. Google Maps API                       | 66        |
| 3.5.2.2. Firebase Auth API                     | 66        |
| 3.5.2.3. Geolocator API                        | 67        |
| 3.5.2.4. Permission handler API                | 67        |

|  |           |
|--|-----------|
| 3.5.2.5. Shared preferences API                          | 67        |
| <b>3.5.3. Consum del servidor</b>                        | <b>67</b> |
| <b>3.5.4. Subministrament del servidor</b>               | <b>68</b> |
| <b>3.5.5. Consum de Dades Obertes</b>                    | <b>70</b> |
| <b>3.6. Eines de desenvolupament i entorn de treball</b> | <b>71</b> |
| <b>3.6.1. Deployment</b>                                 | <b>71</b> |
| 3.6.1.1. Backend   | 71        |

# 1. Requisits

## 1.1. Concepció general del projecte

El nostre projecte consisteix en una aplicació nativa per a Android que recull tots els esdeveniments de tipus cultural que es fan a Catalunya, i permet a l'usuari buscar-los i organitzar-los de manera personalitzada i apuntar-se als que vulgui.

## 1.2. “NOT” list

A continuació mostrem la NOT-List actualitzada que anirem modificant a mesura que avancem en les fases del projecte. En **verd** surten les funcionalitats que estan actualment implementades, 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            |
| Recomanació d'activitats segons els interessos dels usuaris   | Possibilitat de promocionar activitat cultural                        |
| Valoració d'esdeveniments entre 1 i 5 punts   |   |

|  |  |
|--|--|
| 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 multi idioma (català, castellà, anglès) [Només menú] |
| Sistema per reportar errors d'esdeveniments  | Funcionament fora de Catalunya                               |
| Pàgina per administrador per consultar i respondre als reports dels usuaris.   | Compartir esdeveniments a través de xarxes socials o URL     |
| Estadístiques de l'esdeveniment (número assistents, valoració mitjana...)  | Xat amb els responsables dels esdeveniments                  |
| Sistema de notifikacions per als usuaris sobre els esdeveniments als que han afegit a alguna llista, assistiran o els poden interessar | Sincronització amb Google Calendar                           |
| Mostrar els esdeveniments més populars a través d'un mapa de calor   | Apartat de dubtes freqüents i ajuda                          |

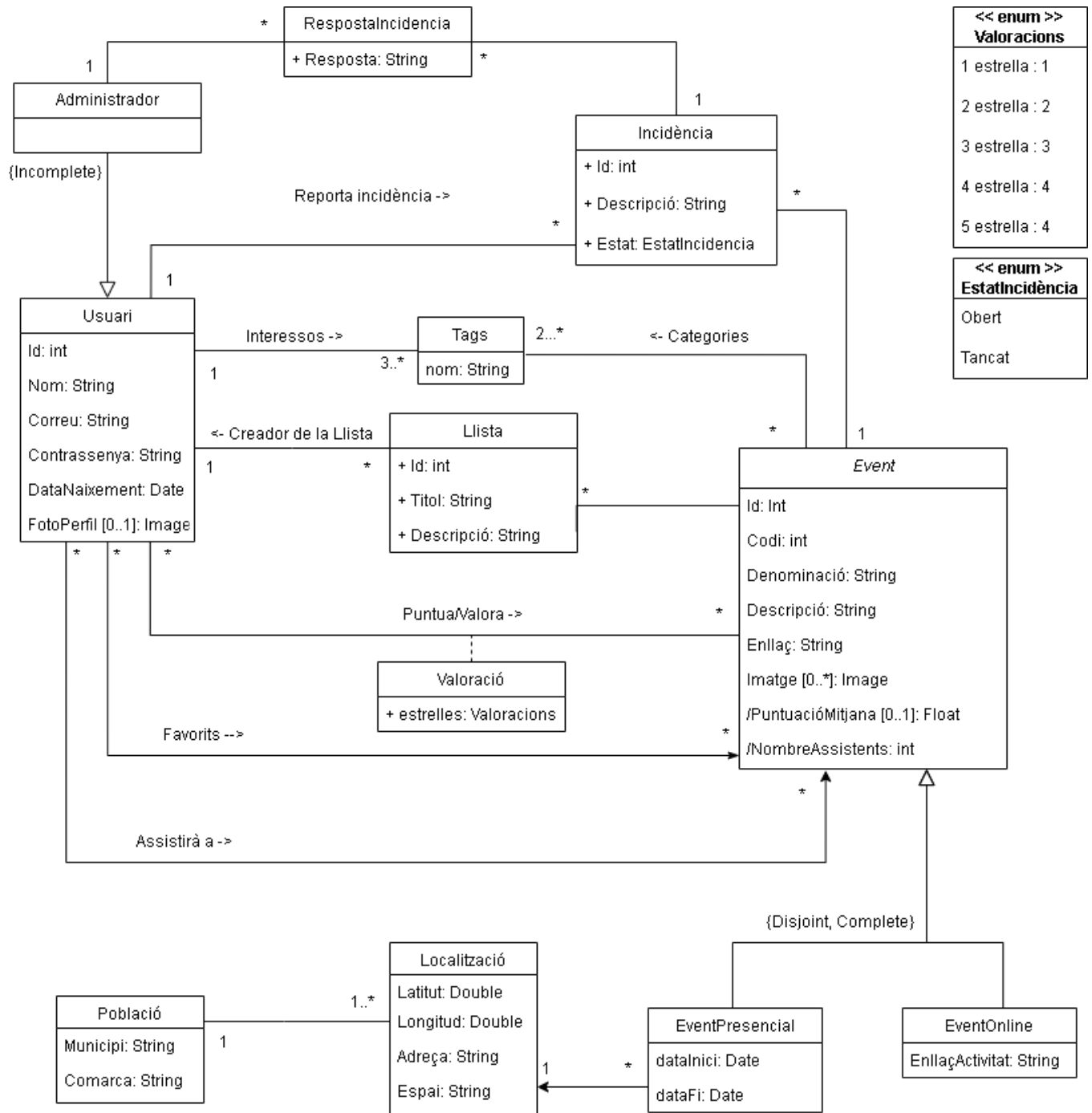
D'aquesta llista, primerament podem dir que no hem implementat cap funcionalitat inclosa en la part 'out of scope', ja que van quedar descartades a l'etapa d'incepció i a mesura que hem avançat el projecte hem confirmat que l'abast del nostre projecte és prou adequat. Aquestes funcionalitats estan marcades en color **vermell**.

Les funcionalitats que hem implementat a l'aplicació durant aquest esprint han estat les marcades en **blau** a la not list. Aquestes funcionalitats han estat completament finalitzades i funcionen completament. Incloent la part de Backend i Frontend. Aquestes funcionalitats són les més importants del sistema com les vistes principals (vista detallada, llista, calendari...), mostrar el mapa amb la localització pròpia i la dels esdeveniments i les funcionalitats més importants del Backend com guardar els esdeveniments a partir del 2023 a la base de dades i proporcionar l'API al Frontend.

Les funcionalitats que hem implementat a l'aplicació, però que no han estat finalitzades del tot, per problemes de temps o d'altres, estan marcades de color **groc**. Aquestes funcionalitats inclouen parts crítiques de l'aplicació que han portat una complexitat alta i d'altres que encara que no eren massa complexes, han sorgit altres problemes con canvis de prioritat o conflictes previs. Per exemple, per les estadístiques d'esdeveniments està implementada la valoració mitjana però falta per acabar d'afegir el nombre d'assistents. A més, la sincronització periòdica de dades ha tingut una complexitat més alta de l'esperada al primer sprint, però pel segon ja es va poder tancar. Les altres dues, sistema per reportar errors d'esdeveniments i la pàgina d'administració no s'han pogut acabar pel segon sprint. Pel que fa a aquestes en el segon sprint es van finalitzar la implementació de la lògica d'incidències, a falta de l'API Token. En addició, pel que fa a la pàgina d'administració encara queda terminar-la amb les crides a la nostra API. En el cas dels perfils, s'ha implementat la lògica de creació d'usuari i l'inici de sessió a la part del Backend, i després ja es van implementar les vistes del Frontend. Tampoc s'ha implementat la modificació de perfil.

Les funcionalitats que no hem implementat de la llista de la part de 'in scope' estan marcades de color **taronja**. Aquestes funcionalitats no han estat implementades, ja que no estaven previstes per l'sprint 1 ni pel sprint 2.

### 1.3. Model conceptual de les dades



## RESTRICCIONS TEXTUALS

### 1. Claus Primaries:

**Usuari** (Id); **Tags** (nom); **Event** (Id); **Llista** (Id + Usuari.Id); **Incidència** (Id); **Localització** (Latitud + Longitud + Espai); **Població** (Municipi + Comarca)

2. La dataInici d'un EventPresencial no pot ser posterior 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 PuntuacióMitjana d'un Event es calcula segons la mitjana d'estrelles de Valoració

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



## 1.2. Resum del backlog del producte al document

### 1.2.1. Sprint 1 Backlog

| Èpica |                       | Història d'usuari |                       |       | Tasques  | Front/Back |
|-------|-----------------------|-------------------|-----------------------|-------|--|------------|
| #     | Títol                 | #                 | Títol                 | Punts | #num_tasca: Nom tasca frontend                                     | F          |
|       |                       |                   |                       |       | #num_tasca: Nom tasca backend                                      | B          |
| -     | -                     | -                 | Storyless tasks       | -     | #125: Infraestructura del domini                                   | B          |
|       |                       |                   |                       |       | #111: Implementar infraestructura per a la traducció dels elements | F          |
| #5    | Altres Funcionalitats | #66               | Informació consistent | 22.5  | #82: Setup de la base de dades                                     | B          |
|       |                       |                   |                       |       | #83: Cridar l'Api de l'Agenda Cultural                             | B          |
|       |                       |                   |                       |       | #85: Processar preu  | B          |
|       |                       |                   |                       |       | #86: Processar Comarca i Municipi                                  | B          |
|       |                       |                   |                       |       | #96: Processament de dades per data inicial/final                  | B          |
|       |                       |                   |                       |       | #114: Processar Tags:Àmbits/Categories/Altres_Categories           | B          |
|       |                       |                   |                       |       | #113: Processar Descripcions                                       | B          |
|       |                       |                   |                       |       | #115: Processar Localització                                       | B          |

|    |                                |     |  |     |   |   |
|----|--------------------------------|-----|--|-----|---|---|
| #1 | Cerca i consulta esdeveniments | #10 | Consultar esdeveniments en llista                | 8   | #110: Vista llista esdeveniments                    | F |
|    |                                |     |  |     | #118: Component element de llista                   | F |
|    |                                |     |  |     | #117: Exposar endpoint API                          | B |
|    |                                | #13 | Consultar informació detallada d'un esdeveniment | 8   | #104: Mostrar dades d'event                         | F |
|    |                                |     |  |     | #119: Exposar endpoint API                          | B |
|    |                                | #15 | Cercar esdeveniments filtrant per categoria      | 10  | #109: Mostrar esdeveniments filtrats                | F |
|    |                                |     |  |     | #121: Crear endpoint API amb paràmetres per filtrar | B |
|    |                                |     |  |     | #122: Crear endpoint per llistar possibles tags     | B |
|    |                                | #14 | Cercar esdeveniments per nom                     | 4.5 | #99: Filtratge per nom                              | B |
|    |                                |     |  |     | #106: Mostrar events del filtratge                  | F |
|    |                                | #12 | Consulta la meua ubicació al mapa                | 4.5 | #123: Integrar API Google Maps                      | F |
|    |                                |     |  |     | #108: Mostrar Mapa                                  | F |
|    |                                |     |  |     | #126: Cridar API localització dispositiu            | F |
|    |                                | #11 | Consulta la ubicació dels esdeveniments al mapa  | 11  | #93: Mostrar ubicació al mapa                       | F |
|    |                                | #17 | Cercar esdeveniments filtrant per rang de dates  | 8   | #101: Filtratge per data (API)                      | B |
|    |                                |     |  |     | #124: Afegir Selector de dates                      | F |

|         |                       |     |   |      |   |   |
|---------|-----------------------|-----|---|------|---|---|
|         |                       | #21 | Consultar esdeveniments mitjançant el calendari | 12   | #112: Mostrar esdeveniments en vista calendari          | F |
| #3      | Gestió Usuari         | #27 | Crear Usuari                                    | 9    | #98: Crear vista creació d'usuari                       | F |
|         |                       |     |   |      | #100 Implementar formulari inicial                      | F |
|         |                       |     |   |      | #102: Implementar formulari d'interessos                | F |
|         |                       |     |   |      | #103: Guardar usuari a la base de dades                 | B |
|         |                       | #41 | Iniciar Sessió                                  | 8    | #91: Autenticar usuari (API)                            | B |
|         |                       |     |   |      | #90: Enviar credencials i rebre resposta                | F |
|         |                       |     |   |      | #89: Crear vista d'iniciar sessió                       | F |
|         |                       | #42 | Tancar Sessió                                   | 6    | #94: Crear botó/event/vista per tancar sessió           | F |
| #6<br>3 | Incidències           | #39 | Reportar error d'esdeveniment                   | 13   | #84 Creació de la taula d'incidències (BD + API)        | B |
|         |                       |     |   |      | #107: Crear vista d'incidència                          | F |
| #5      | Altres Funcionalitats | #64 | Dades actualitzades                             | 13.5 | #95: Crida periòdica a Dades Obertes (sync)             | B |
| -       | -                     | -   | Storyless Task                                  | -    | #97: Implementar navegabilitat entre pantalles (navbar) | F |

En total ens han sortit 136.5 story points dels 190 que haurien d'haver sortit si haguéssim distribuït les històries de forma equitativa. Aquests 136.5 punts no inclouen les 3 tasques (#97, #11, #125) que no estan assignades a cap història d'usuari ni la feina “extra” d'estructurar el codi desde zero.

### 1.2.2. Sprint 2 Backlog

| Èpica |                                | Història d'usuari |   |       | Tasques   | Front/<br>Back |
|-------|--------------------------------|-------------------|---|-------|---|----------------|
| #     | Títol                          | #                 | Títol   | Punts | #num_tasca: Nom tasca frontend  | F              |
|       |                                |                   |   |       | #num_tasca: Nom tasca backend   | B              |
| #5    | Altres Funcionalitats          | #170              | Servei de llista esdeveniments                    | 10    | #171: Afegir endpoint a la API per a rebre esdeveniments en un radi d'amplitud i un rang de dades | B              |
|       |                                | #64               | Dades actualitzades                               | 13.5  | #95: Crida periòdica a Dades Obertes (sync)   | B              |
| #1    | Cerca i consulta esdeveniments | #14               | Cercar esdeveniments per nom                      | 4.5   | #99: Filtratge per nom  | B              |
|       |                                |                   |   |       | #106: Mostrar events del filtratge  | F              |
|       |                                | #16               | Cercar esdeveniments filtrant per distància       | 9.5   | #157: Afegir endpoint a la API per a rebre esdeveniments en un radi d'amplitud                    | B              |
|       |                                | #11               | Consultar la ubicació dels esdeveniments al mapa  | 11    | #93: Mostrar ubicació al mapa   | F              |
|       |                                |                   |   |       | #144: Afegir endpoint a la API per a rebre esdeveniments en un radi d'amplitud                    | B              |
|       |                                | #9                | Consultar esdeveniments recomanats per proximitat | 10    | #143: Afegir endpoint a la API per a rebre esdeveniments en un radi d'amplitud                    | B              |

|    |               |     |   |    |   |   |
|----|---------------|-----|---|----|---|---|
|    |               | #15 | Cercar esdeveniments filtrant per categoria       | 10 | #109: Mostrar esdeveniments filtrats                | F |
|    |               |     |   |    | #121: Crear endpoint API amb paràmetres per filtrar | B |
|    |               |     |   |    | #122: Crear endpoint per llistar possibles tags     | B |
|    |               | #8  | Consultar esdeveniments recomanats per categories | 10 | #142: Desenvolupar algorisme de recomanació         | B |
|    |               | #17 | Cercar esdeveniments filtrant per rang de dates   | 8  | #101: Filtratge per data (API)                      | B |
|    |               |     |   |    | #124: Afegir Selector de dates                      | F |
|    |               |     |   |    | #132: Crear vista de filtres                        | F |
| #3 | Gestió Usuari | #27 | Crear Usuari                                      | 9  | #98: Crear vista creació d'usuari                   | F |
|    |               |     |   |    | #100 Implementar formulari inicial                  | F |
|    |               |     |   |    | #102: Implementar formulari d'interessos            | F |
|    |               |     |   |    | #133: Guardar usuari a la base de dades             | B |
|    |               | #41 | Iniciar Sessió                                    | 8  | #91: Autenticar usuari (API)                        | B |
|    |               |     |   |    | #90: Enviar credencials i rebre resposta            | F |
|    |               |     |   |    | #89: Crear vista d'iniciar sessió                   | F |
|    |               |     |   |    | #103: Guardar usuari a la base de dades             | B |

|     |               |     |  |     |   |   |
|-----|---------------|-----|--|-----|---|---|
|     |               | #51 | Iniciar Sessió amb Google                | 11  | #158: Afegir botó per iniciar sessió amb Google                   | F |
|     |               |     |  |     | #159: Autenticació amb API de Google                              | B |
|     |               | #42 | Tancar Sessió                            | 6   | #94: Crear botó/event/vista per tancar sessió                     | F |
| #63 | Incidències   | #39 | Reportar error d'esdeveniment            | 13  | #84 Creació de la taula d'incidències (BD + API)                  | B |
|     |               |     |  |     | #107: Crear vista d'incidència                                    | F |
| #3  | Gestió Usuari | #29 | Consultar el meu perfil                  | 8.5 | #160: Afegir vista per consultar perfil                           | F |
|     |               |     |  |     | #161: Afegir endpoint a la API per consultar les dades del perfil | B |
|     |               | #42 | Tancar Sessió                            | 6   | #94: Crear botó/event/vista per tancar sessió                     | F |
|     |               | #28 | Eliminar Usuari                          | 7   | #162: Crear botó per eliminar usuari                              | F |
|     |               |     |  |     | #163: Afegir endpoint a la API per eliminar usuari                | B |
|     |               | #30 | Editar els meus interessos al meu perfil | 9   | #141: Refactoritzar els usuaris per a afegir categories d'interés | B |
|     |               | #56 | Modificar foto de perfil                 | 6.5 | #164: Afegir vista per modificar foto de perfil                   | F |
|     |               |     |  |     | #165: Afegir nova foto de perfil a la BD                          | B |
|     |               | #57 | Canviar nom perfil                       | 7   | #166: Afegir vista per canviar nom perfil                         | F |
|     |               |     |  |     | #167: Canviar el nom perfil a la BD                               | B |
|     |               | #52 | Canviar contrasenya                      | 7   | #168: Afegir vista per canviar contrasenya                        | F |

|     |  |     |  |      |  |   |
|-----|--|-----|--|------|--|---|
|     |  |     |  |      | #169: Canviar la contrasenya de l'usuari a la BD                               | B |
| #4  | Interacció<br>usuari-esdevenime<br>nts | #35 | Apuntar-me a un<br>esdeveniment                | 5.5  | #154: Crear una taula amb assistències   | B |
|     |  |     |  |      | #155: Afegir endpoint a la API per a apuntar un usuari a un esdeveniment       | B |
|     |  |     |  |      | #156: Afegir endpoint a la API per a desapuntar un usuari d'un esdeveniment    | B |
|     |  | #32 | Afegir un esdeveniment a la llista de favorits | 6.5  | #153: Afegir endpoint a la API per a afegir un esdeveniment a una llista       | B |
|     |  | #37 | Puntuar un esdeveniment                        | 6.5  | #151: Afegir endpoint a la API per a valorar un esdeveniment                   | B |
|     |  | #38 | Esborrar la meva puntuació d'un esdeveniment   | 6.5  | #152: Afegir endpoint a la API per a esborrar una valoració                    | B |
|     |  | #43 | Crear llista                                   | 6.5  | #149: Afegir llistes a la BD   | B |
|     |  | #45 | Afegir esdeveniment a la llista                | 5.5  | #150: Afegir endpoint a la API per a afegir un esdeveniment a una llista       | B |
| #63 | Incidències                            | #54 | Iniciar sessió a la pàgina d'administració     | 10.5 | #138: Afegir el formulari d'iniciar sessió                                     | B |
|     |  |     |  |      | #139: Refactoritzar els usuaris per a afegir administradors                    | B |
|     |  |     |  |      | #140: Afegir endpoint a la API per a autenticació a la pàgina d'administradors | B |
|     |  | #58 | Tancar sessió a la pàgina                      | 6.5  | #137: Afegir el botó de tancar sessió  | B |

|   |   |     |                                    |     |   |   |
|---|---|-----|------------------------------------|-----|---|---|
|   |   |     | d'administració                    |     |   |   |
|   |   | #49 | Veure incidències obertes/tancades | 6.5 | #148: Crear pagina estàtica amb les incidències   | B |
|   |   | #48 | Veure totes les incidències        | 6.5 | #81: Crear pagina estatica amb les incidències  | B |
|   |   | #50 | Cambiar l'estat d'una incidència   | 9.5 | #147: Afegir el formulari de la incidència  | B |
|   |   | #53 | Respondre incidència               | 9.5 | #146: Afegir el formulari de la incidència  | B |
| - | - | -   | Storyless Tasks                    | -   | #97: Implementar la navegabilitat entre pantalles   | F |
|   |   |     |                                    |     | #111: Implementar la infraestructura necessària per a convertir etiquetes/títols de la UI a strings | F |
|   |   |     |                                    |     | #127: Infraestructura MVVM + Repository   | F |
|   |   |     |                                    |     | #135: Implementar API Repository  | F |
|   |   |     |                                    |     | #145: Desenvolupar el servei a oferir   | B |
|   |   |     |                                    |     | #174: Afegir esquelet per al tractament d'excepcions  | B |
|   |   |     |                                    |     | #175: Afegir tractament d'excepcions per al controlador d'usuaris                                   | B |
|   |   |     |                                    |     | #177: Afegir tractament d'excepcions per al controlador d'assistències                              | B |
|   |   |     |                                    |     | #183: Afegir filtrat avançat per a les incidències  | B |
|   |   |     |                                    |     | #200: Securitzar crida d'incidències amb API Token  | B |



|  |  |  |  |  |   |   |
|--|--|--|--|--|---|---|
|  |  |  |  |  | #145: Desenvolupar servei a oferir                                      | B |
|  |  |  |  |  | #189: Afegir DTOs per a les assistències                                | B |
|  |  |  |  |  | #176: Afegir tractament d'excepcions per al controlador d'incidències   | B |
|  |  |  |  |  | #188: Afegir DTOs per a les incidències                                 | B |
|  |  |  |  |  | #173: Afegir seguretat a la API   | B |
|  |  |  |  |  | #178: Afegir tractament d'excepcions per al controlador d'esdeveniments | B |

### 1.2.3. Sprint 3 Backlog

| Èpica |                       | Història d'usuari |  |       | Tasques   | Front/<br>Back |
|-------|-----------------------|-------------------|--|-------|---|----------------|
| #     | Títol                 | #                 | Títol  | Punts | #num_tasca: Nom tasca frontend                    | F              |
|       |                       |                   |  |       | #num_tasca: Nom tasca backend                     | B              |
| #5    | Altres Funcionalitats | #62               | Sincronitzar esdeveniments amb Google Calendar | 13.5  | #231: Configurar integració amb Google Calendar   | F              |
|       |                       |                   |  |       | #233: Implementar comunicació amb Google calendar | B              |

|    |                                |      |   |     |  |   |
|----|--------------------------------|------|---|-----|--|---|
|    |                                | #251 | Integrar el servei de carregadors                       |     | #252Mostrar carregadors mes propers a la ubicació d'un esdeveniment                            | F |
| #1 | Cerca i consulta esdeveniments | #74  | Consultar visualment els events més populars en el mapa | 15  | #208: Refactor del plugin de mapa de claor de Google Maps                                      | F |
|    |                                |      |   |     | #209: Implementar el mapa de calor en base a l'assistencia d'esdeveniments                     | F |
|    |                                | #20  | Cercar esdeveniments filtrant per municipi              | 6.5 | #234: Implementar crida a la API donat un filtre de municipi                                   | F |
|    |                                |      |   |     | #235: Crear endpoint de consulta per municipi  | B |
|    |                                | #11  | Consultar la ubicació dels esdeveniments al mapa        | 11  | #202: Mostrar marcadors associats als Events a la posició marcada per la seva latitud/longitud | F |
|    |                                |      |   |     | #217: Fer cluster de marcadors depenent del zoom del mapa                                      | F |
|    |                                | #9   | Consultar esdeveniments recomanats per proximitat       | 10  | #203: Afegir Carrousel d'events a la Homepage  | F |
|    |                                | #88  | Veure suggerencies d'esdeveniments segons interessos    | 10  | #248: Mostrar esdeveniments a la vista principal   | F |

|    |               |     |  |     |  |   |
|----|---------------|-----|--|-----|--|---|
|    |               | #8  | Consultar esdeveniments<br>recomanats per categories | 10  | #199: Crear/especificar 'Theme' per la app                         | F |
|    |               |     |  |     | #197: Crear vista principal "Homepage"                             | F |
|    |               | #31 | Consultar esdeveniments<br>marcats com a favorits    | 8   | #250: Filtrar i enviar llista d'esdeveniments favorits d'un usuari | B |
|    |               |     |  |     | #249: Implememntar vista amb llista d'esdeveniments favorits       | F |
|    |               | #79 | Consultar assistents a un<br>esdeveniment            |     | #237: Mostrar número aproximat s'assistents d'un esdeveniment      | F |
|    |               |     |  |     | #236: Actualitzar assistents a la BD                               | B |
| #3 | Gestió Usuari | #27 | Crear Usuari   | 9   | #102: Implementar formulari d'interessos                           | F |
|    |               | #29 | Consultar el meu perfil                              | 8.5 | #160: Afegir vista per consultar perfil                            | F |
|    |               |     |  |     | #210: Implementar edició de camps de perfil                        | F |
|    |               |     |  |     | #211: Implementar canvi d'imatge de perfil                         | F |
|    |               | #28 | Eliminar Usuari                                      | 7   | #162: Crear botó per eliminar usuari                               | F |
|    |               | #30 | Editar els meus interessos al<br>meu perfil          | 9   | #204: Afegir vista per seleccionar les categories interesades      | F |
|    |               | #56 | Modificar foto de perfil                             | 6.5 | #164: Afegir vista per modificar foto de perfil                    | F |

|    |  |     |  |     |   |   |
|----|--|-----|--|-----|---|---|
|    |  |     |  |     | #165: Afegir nova foto de perfil a la BD                          | B |
|    |  | #57 | Canviar nom perfil   | 7   | #166: Afegir vista per canviar nom perfil                         | F |
|    |  | #52 | Canviar contrasenya  | 7   | #168: Afegir vista per canviar contrasenya                        | F |
| #4 | Interacció<br>usuari-esdevenime<br>nts | #35 | Apuntar-me a un esdeveniment                               | 5.5 | #207: Cridar endpoint API   | F |
|    |  |     |  |     | #218: Implementar algoritme de popularitat                        | B |
|    |  | #32 | Afegir un esdeveniment a la llista de favorits             | 6.5 | #214: Crear vista amb llista d'esdeveniments favorits             | F |
|    |  | #47 | Editar nom de llista                                       | 6.5 | #223: Crear formulari per editar el nom                           | B |
|    |  |     |  |     | #222: Editar llista a la BD                                       | F |
|    |  | #37 | Puntuar un esdeveniment                                    | 6.5 | #219 Implementar la votació interactiva am estrelles              | F |
|    |  | #38 | Esborrar la meva puntuació d'un esdeveniment               | 6.5 | #220 Afegir el borrat de valoració a la UI                        | F |
|    |  | #46 | Eliminar esdeveniment de llista                            | 5.5 | #225: Afegir botó d'eliminar esdeveniment                         | F |
|    |  | #44 | Eliminar llista  | 6.5 | #226: Afegir botó d'eliminar llista                               | F |
|    |  | #36 | Desapuntar-me d'un esdeveniment                            | 5.5 | #228: Afegir botó per deixar de ser assistent                     | F |
|    |  |     |  |     | #229: Borrar a l'usuari com a assistent d'un esdeveniment a la BD | B |
|    |  | #60 | Consultar els esdeveniments als que assistiré en calendari | 11  | #230: Crear calendari d'usuari                                    | F |

|     |                         |     |  |      |   |   |
|-----|-------------------------|-----|--|------|---|---|
| #63 | Incidències             | #54 | Iniciar sessió a la pàgina d'administració           | 10.5 | #138: Afegir el formulari d'iniciar sessió                                    | B |
|     |                         | #49 | Veure incidències obertes/tancades                   | 6.5  | #148: Crear pagina estàtica amb les incidències                               | B |
|     |                         | #48 | Veure totes les incidències                          | 6.5  | #81: Crear pagina estatica amb les incidències                                | B |
|     |                         | #50 | Cambiar l'estat d'una incidència                     | 9.5  | c   | B |
|     |                         | #53 | Respondre incidència                                 | 9.5  | #146: Afegir el formulari de la incidència                                    | B |
| #2  | Compartir esdeveniments | #75 | Mostrar informació d'un esdeveniment al obrir un URL | 8    | #232: Redireccionar a la vista detallada d'un esdeveniment a partir d'una URL | F |
|     |                         | #22 | Compartir esdeveniment per Whatsapp                  | 10   | #240: Redirigir a vista detallada d'esdeveniment                              | F |
|     |                         |     |  |      | #239: Implementar crides a la API   | F |
|     |                         |     |  |      | #238: Configurar integració amb API   | F |
|     |                         | #23 | Compartir esdeveniment per Instagram                 | 10   | #240: Redirigir a vista detallada d'esdeveniment                              | F |
|     |                         |     |  |      | #239: Implementar crides a la API   | F |
|     |                         |     |  |      | #238: Configurar integració amb API   | F |
|     |                         | #24 | Compartir esdeveniment per Twitter                   | 10   | #240: Redirigir a vista detallada d'esdeveniment                              | F |
|     |                         |     |  |      | #239: Implementar crides a la API   | F |
|     |                         |     |  |      | #238: Configurar integració amb API   | F |
|     |                         |     |  |      | #97: Implementar la navegabilitat entre pantalles                             | F |
|     |                         |     |  |      |   |   |

Storyless Task

|  |  |  |  |  |   |   |
|--|--|--|--|--|---|---|
|  |  |  |  |  | #111: Implementar la infraestructura necessària per a convertir etiquetes/títols de la UI a strings | F |
|  |  |  |  |  | #127: Infraestructura MVVM + Repository   | F |
|  |  |  |  |  | #135: Implementar API Repository  | F |
|  |  |  |  |  | #145: Desenvolupar el servei a oferir   | B |

## 1.4. Requisits no funcionals

En aquesta secció es descriu el conjunt de requisits no funcionals que conté el nostre sistema, és a dir, tots aquests que mencionen i expliquen les propietats del nostre producte, sense posar èmfasi en les funcions principals que ha de poder fer la nostra aplicació. Aquests requisits estan identificats per un número i segueixen la classificació utilitzada en la plantilla de *Volere*:

### 1.5.1 Requisits de rendiment

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | <b>1</b>  |
| <b>Tipus de requisit</b>         | 15a. Requisits de longevitat  |
| <b>Descripció</b>                | El producte s'ha de poder mantenir funcionalment en qualsevol moment.   |
| <b>Justificació del requisit</b> | Cal tenir de manera activa el sistema durant un gran període de temps per tal de poder assolir els objectius proposats per <i>CultureFinder</i> . |
| <b>Condicció de satisfacció</b>  | ❖ L'aplicació romandrà funcional i activa fins al darrer dia en el qual acabi el projecte de l'assignatura.                                       |

### 1.5.2 Requisits de preservació i suport

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | <b>2</b>  |
| <b>Tipus de requisit</b>         | 14b. Requisits de suport  |
| <b>Descripció</b>                | Especifica el nivell de suport que el sistema necessita.  |
| <b>Justificació del requisit</b> | Cal tenir la seguretat de que el nostre sistema proporciona suport als usuaris quan troben problemes relacionats amb l'aplicació. |
| <b>Condicció de satisfacció</b>  | ❖ S'ofereix la possibilitat d'afegir incidències relacionades amb els esdeveniments de l'aplicació.                               |

### 1.5.3 Requisits de capacitat d'ús i humanitat

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | <b>3</b>  |
| <b>Tipus de requisit</b>         | 11a. Requisits de facilitat d'ús  |
| <b>Descripció</b>                | L'aplicació ha de ser fàcil d'usar i de navegar per a qualsevol usuari, independentment del seu nivell d'experiència o coneixements tècnics. Ha d'estar dissenyada de manera clara i intuïtiva, amb una interfície d'usuari simple i fàcil d'entendre.  |
| <b>Justificació del requisit</b> | La facilitat d'ús és un requisit important per a qualsevol aplicació, ja que pot afectar directament la satisfacció de l'usuari i l'adopció de l'aplicació. Si l'aplicació és difícil d'usar o de navegar, els usuaris poden sentir frustració i no usar-la, la qual cosa pot afectar negativament l'èxit del projecte. |
| <b>Condicció de satisfacció</b>  | ❖ El percentatge d'usuaris que consideren l'aplicació fàcil d'usar i navegar ha de ser superior al 80%. A més, qualsevol problema d'usabilitat que es detecti durant les proves ha de ser corregit abans del llançament de l'aplicació.   |

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | <b>4</b>  |
| <b>Tipus de requisit</b>         | 11b. Requisits de personalització i internacionalització  |
| <b>Descripció</b>                | Especifica com l'aplicació pot modificar-se per adaptar-se a les necessitats específiques dels usuaris. Això significa que els usuaris han de poder personalitzar l'aplicació d'acord amb les seves preferències personals. |
| <b>Justificació del requisit</b> | Cal tenir clara la intenció de ser utilitzada en diferents mercats i per diferents usuaris amb diferents necessitats i preferències. Si l'aplicació no es pot personalitzar, l'usuari pot tenir una pitjor                  |



|                                |   |
|--------------------------------|---|
|                                | visió respecte al producte.   |
| <b>Condició de satisfacció</b> | ❖ Els esdeveniments recomanats per l'usuari estan basats en els seus interessos principals indicats a l'hora de registrar-se a l'aplicació. |

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | <b>5</b>  |
| <b>Tipus de requisit</b>         | 11c. Requisits d'aprenentatge   |
| <b>Descripció</b>                | Especifica el nivell de facilitat que ha de tenir l'aplicació per poder ser utilitzada.   |
| <b>Justificació del requisit</b> | És de vital importància que als usuaris de la nostra aplicació no els hi sigui complicat aprendre el seu funcionament, és a dir, ha de ser intuïtiva. Tanmateix, podrien frustrar-se i perdre l'interés en aquesta. |
| <b>Condició de satisfacció</b>   | ❖ No es requereix cap curs ni tutorial previ per tal de fer ús de la nostra aplicació. Això es validarà tenint en compte l'opinió d'un conjunt d'usuaris per rebre el feedback necessari.                           |

### 1.5.3 Requisits de seguretat

|                          |   |
|--------------------------|---|
| <b>Número</b>            | <b>6</b>  |
| <b>Tipus de requisit</b> | 15c. Requisits de privacitat  |
| <b>Descripció</b>        | Especifica com el sistema pot garantir la privadesa de la informació que s'emmagatzema dels usuaris. Per tant, és fonamental tenir en compte les lleis relacionades amb privacitat sobre les dades. |

|                                  |   |
|----------------------------------|---|
| <b>Justificació del requisit</b> | Un dels punts més destacables de la nostre producte és el compromís de fer unes bones pràctiques i mantenir la informació protegida per tal d'evitar atacs de seguretat que comprometin les dades.                  |
| <b>Condicció de satisfacció</b>  | <ul style="list-style-type: none"> <li>❖ L'aplicació ha de protegir la informació privada de cada client.</li> <li>❖ No es mostrarà tota la informació que pugui ser sensible, com correus dels usuaris.</li> </ul> |

|                                  |   |
|----------------------------------|---|
| <b>Número</b>                    | 7   |
| <b>Tipus de requisit</b>         | 15a. Requisits d'accés  |
| <b>Descripció</b>                | Especifica les persones autoritzades per accedir al producte i a les crides corresponents, és a dir, les diverses funcionalitat com l'accés a les dades de l'aplicació. Depenent de quin context i tipus d'usuari accedeixi, es mostraran unes dades o unes altres, sempre tenint en compte els privilegis de cadascun. |
| <b>Justificació del requisit</b> | És necessari que totes les dades introduïdes pels usuaris siguin correctament guardades al nostre producte.   |
| <b>Condicció de satisfacció</b>  | <ul style="list-style-type: none"> <li>❖ El sistema valida les dades introduïdes pels usuaris.</li> </ul>   |

Una vegada ja tenim una arquitectura ben definida i una estructura ben desenvolupada de la nostra aplicació, tant per la part del backend com per la part del frontend, podem mostrar algunes de les tasques que es veuen influenciades en els requisits no funcionals.

| Número RNF | Número Volere | Tipus de requisit | Història d'usuari | Estat |
|------------|---------------|-------------------|-------------------|-------|
|------------|---------------|-------------------|-------------------|-------|

|   |      |                 |   |           |
|---|------|-----------------|---|-----------|
| 1 | 15a. | Longevitat      | #64   | Completed |
| 2 | 14b. | Suport          | #48, #49, #50, #54  | Completed |
| 3 | 11a. | Facilitat d'ús  | -   | Completed |
| 4 | 11b. | Personalització | #8, #20, #80  | Completed |
| 5 | 11c. | Aprenentatge    | -   | Completed |
| 6 | 15c. | Privacitat      | StoryLess Task (Tasques de backend relacionades amb securització) | Completed |
| 7 | 15a. | Accés           | StoryLess Task (Tasques de backend relacionades amb securització) | Completed |

Pel que fa als últims dos punts no es poden verificar amb històries d'usuaris concretes, ja que al Taiga ho hem indicat d'aquesta manera les millores que anàvem realitzant relacionades amb Refactor i Securitació.

## 1.5. Tractament dels aspectes transversals

Sprint 1:

**Geolocalització:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

**Xarxes socials:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

**Xat:** No tenim pensat implementar aquest aspecte transversal en la nostra aplicació.

**Gamificació:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

**Stakeholders reals:** Un cop finalitzat el primer sprint vam fer una petita demostració a un familiar, però ens vam adonar que encara quedaven bastants funcionalitats per ser usable de manera adequada.

**Refutació:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

**Calendari:** El calendari encara no mostra del tot bé els esdeveniments.

**Web-app admin:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

**Multiidioma:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 1.

| Aspecte transversal   | Estatus actual |
|-----------------------|----------------|
| Geolocalització       | ×              |
| Xarxes socials        | ×              |
| Xat                   | ×              |
| Gamificació           | ×              |
| Stakeholders<br>reals | ✓              |
| Refutació             | ×              |
| Calendari             | ×              |
| Web-app admin         | ×              |
| Multiidioma           | ×              |

Sprint 2:

**Geolocalització:** En aquest segon sprint hem estat implementat aquest aspecte tenint en compte totes les històries d'usuari relacionades amb l'ubicació del client que utilitza l'aplicació fent ús del mapa.

**Xarxes socials:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 2.

**Xat:** No tenim pensat implementar aquest aspecte transversal en la nostra aplicació.

**Gamificació:** Les valoracions estan implementades correctament a data de fi de l'Sprint 2.

**Stakeholders reals:** Un cop finalitzat el segon sprint farem una demostració a un familiar o amic, per tal de rebre feedback en aspectes com usabilitat o rendiment.

**Refutació:** Aquest aspecte transversal no ha estat implementat al acabar l'Sprint 2.

**Calendari:** S'ha implementat un calendari per tal de tenir una de les funcionalitats principals del nostre projecte que és poder cercar per filtratge i calendari els esdeveniments de l'aplicació.

**Web-app admin:** En aquest segon sprint s'ha començat a implementar la pàgina web d'administració fent la interfície principal, tenint en compte les crides a l'API que hem anat implementat relacionades amb incidències.

**Multiidioma:** Aquest aspecte transversal finalment no ha estat implementat.

| Aspecte transversal   | Estatus actual |
|-----------------------|----------------|
| Geolocalització       | ✓              |
| Xarxes socials        | ✗              |
| Xat                   | ✗              |
| Gamificació           | ✓              |
| Stakeholders<br>reals | ✓              |
| Refutació             | ✗              |
| Calendari             | ✓              |
| Web-app admin         | ✓              |
| Multiidioma           | ✗              |

### Sprint 3:

**Geolocalització:** La vista de mapa es centra en la ubicació actual del usuari. Apart d'això, aquesta vista també fa crides a la API del back tenint en compte aquesta localització.

**Xarxes socials:** Teniem pensat afegir l'opció de compartir esdeveniments, pero finalment no hem pogut.

**Xat:** No tenim pensat implementar aquest aspecte transversal en la nostra aplicació.

**Gamificació:** El mapa de calor mostra de forma interactiva, la popularitat dels events

**Stakeholders reals:** Vam ensenyar l'aplicació a altres amics per tal de que ens donessin feedback per tal de millorar la experiencia d'usuari.

**Refutació:** Aquest aspecte transversal no ha estat implementa.

**Calendari:** S'ha implementat un calendari per tal de tenir una de les funcionalitats principals del nostre projecte que és poder cercar per filtratge i calendari els esdeveniments de l'aplicació.

**Web-app admin:** En el tercer sprint la pàgina ha quedat a mitges, visualment esta finalitzada, pero vam tenir problemes a la hora d'implementar les crides a la API.

**Multiidioma:** Aquest aspecte transversal finalment no ha estat implementat.

| Aspecte transversal | Estatus actual |
|---------------------|----------------|
| Geolocalització     | ✓              |
| Xarxes socials      | ✗              |
| Xat                 | ✗              |
| Gamificació         | ✓              |
| Stakeholders reals  | ✓              |
| Refutació           | ✗              |
| Calendari           | ✓              |
| Web-app admin       | ✓              |
| Multiidioma         | ✗              |

## 1.6. Serveis de tercers

### 1.6.1 Repartiment de funcionalitats entre els equips

Durant la segona fase de l'incepció, vam realitzar 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 Donat un esdeveniment, es podran mostrar les estacions de càrrega per a estan al voltant del seu entorn.

Així doncs, la funcionalitat que rebem per part del grup de Mobilitat Sostenible és la següent:  
vehicles elèctrics a Catalunya més properes a aquesta activitat.

#### **Llicitacions i adjudicacions en curs**

En relació amb la negociació amb el grup de llicitacions 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 donat un radi mencionat, una posició (latitud i longitud) i un rang de dates concretes tal de poder observar les activitats culturals de Catalunya.

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

- Donat un interval de dates d'inici i fi, una latitud i longitud i un radi en kilòmetres, proporcionarem els esdeveniments que tenen lloc entre aquestes dates a la localització especificada dins del radi especificat.

### 1.6.2 Comunicació amb la resta dels equips

Pel que fa a la interacció amb els altres equips per als serveis que hem hagut d'implementar i oferir entre nosaltres l'hem gestionat a través de dos canals. Per una banda, el nostre responsable ha aprofitat 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.

## 2. Metodologia

### 2.1. Visió General

#### 2.1.2 Divisió de l'equip en Backend i Frontend

Per tal de poder treballar de millor forma, i així poder estar més centrats en les tasques de cadascun, vam pensar tal i com es produeix en equips que realitzen projectes grans, que la millor opció seria subdividir l'equip en dos: tenir tres membres de l'equip de Backend i tres membres de l'equip de Frontend.

En primer lloc, l'especialització permet un enfocament més profund i concentrat en cada aspecte del desenvolupament. L'equip de backend s'encarrega de la lògica empresarial, la gestió de dades i la implementació de la funcionalitat principal del sistema, mentre que l'equip de frontend es concentra en l'experiència de l'usuari, la interfície d'usuari i la presentació visual del producte. Aquesta separació permet a cada equip desenvolupar la seva experiència i coneixement en la seva àrea específica, millorant la qualitat i eficiència del seu treball.

En segon lloc, la divisió d'equips agilitza el procés de desenvolupament. En treballar en paral·lel, l'equip de backend i l'equip de frontend poden avançar simultàniament en les seves respectives tasques, la qual cosa redueix el temps total de desenvolupament. Això permet un lliurament més ràpid del producte final i una major capacitat de resposta als canvis i requeriments del client.



A més, la divisió d'equips promou una comunicació més clara i eficient. Cada equip pot utilitzar el llenguatge i les eines adequades per a la seva àrea d'especialització, la qual cosa facilita la comunicació interna i la resolució de problemes. A més, en definir interfícies clares entre el backend i el frontend, s'estableixen punts de connexió ben definits, la qual cosa facilita la integració entre els dos components i redueix els possibles conflictes.

### 2.1.2 Scrum

Per a organitzar-nos hem decidit seguir la metodologia *Scrum*. Però, entrant ja en matèria, què és exactament Scrum?

La metodologia Scrum és un procés per a dur a terme un conjunt de tasques de forma regular amb l'objectiu principal de treballar de manera col·laborativa, és a dir, per a fomentar el treball en equip.

Amb aquest mètode de treball el que es pretén és aconseguir el millor resultat d'un projecte determinat. Les pràctiques que s'apliquen amb la metodologia Scrum es retroalimenten les unes amb les altres i la integració de les mateixes té el seu origen en un estudi de com cal coordinar als equips per a ser potencialment competitius.

#### **Product Backlog**

El Product Backlog és la fase en la qual s'estableixen les tasques prioritàries i on s'obté informació breu i detallada sobre el projecte que es desenvoluparà.

Amb el mètode Scrum no és necessari definir tots els objectius al començament del projecte. El Product Owner, de manera conjunta amb l'equip de treball comencen a llistar el més important per al Product Backlog.

El Product Backlog és necessari per a poder arrencar amb el primer sprint, té permès canviar i créixer tantes vegades com sigui necessari en funció de l'aprenentatge adquirit en el desenvolupament del producte.

#### **Sprint**

Dins del mètode Scrum, l'Sprint és el cor, un interval de temps que com a màxim té una durada d'un mes i on es produeix el desenvolupament d'un producte que és lliurable potencialment.

Per a entendre-ho millor, si el Product Owner sol·licita el producte es requereix un mínim esforç per al seu lliurament al client.

També es pot definir l'Sprint com un mini projecte on l'equip de treball es focalitza en el desenvolupament de tasques per a aconseguir l'objectiu que s'ha definit prèviament en l'Sprint planning.

## Burn Down

El Burn Down és la fase en la qual es mesura el progrés d'un determinat projecte Scrum. En ella, el Scrum Màster serà l'encarregat d'actualitzar els gràfics quan es finalitzi cadascun dels Sprints.

A manera de resum, la metodologia Scrum se centra principalment en el treball en equip, seguint una sèrie de criteris establerts per a obtenir els millors resultats en un projecte.

## Aplicació de la metodologia

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 dimegnes 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 (tot i que algunes històries creades durant el projecte no es van definir amb els criteris com hauria d'haver sigut), les tasques associades i els *story points* que té assignats, desglossat en Front, Back i UX.

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

## 2.2. 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.

Adicionalment, hem decidit que per tenir clara la **prioritat** i els objectius principals de cada sprint, a l'hora de repartir les tasques al board ho hem organitzat afegint a la part més superior les històries d'usuari més importants, i a mesura que es va baixant pel board la prioritat corresponent va decreixent. D'aquesta manera, el desenvolupador de cada equip té clara quina és la tasca que ha de començar una vegada ha acabat la que tenia en procés.

A més, com hem mencionat anteriorment al principi de la secció de [Metodologia](#), per tal de dur a terme amb èxit el projecte ens hem dividit en dos subequips, backend i frontend. Per aquest mateix motiu, cada tasca té assignada un *tag* amb el qual es pot observar quin subequip serà l'encarregat de realitzar aquella funcionalitat. Per evitar confusions, hem decidit utilitzar un *tag* anomenat **back** i un altre anomenat *front*.

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 5 fases, les quals definim més endavant a partir d'un problema que vam tenir al primer sprint.

Per qüestions de temps hem decidit no fer *testing*. Sabem i considerem la importància d'aquesta pràctica a l'hora de produir *software* de qualitat, però no hem trobat la manera d'incorporar-lo al nostre projecte per aquestes raons.

Un dels aspectes discutits a la primera reunió de retrospectiva va ser la ambigua Definition of Done que havíem definit anteriorment tant per les històries d'usuari com les tasques que les formen. Per aquest motiu, vam tornar a redefinir el nostre concepte de DoD:

- **[New]** - Història d'usuari creada, en aquest estat no té encara cap criteri d'acceptació.
- **[Ready]** - La història d'usuari té definits un conjunt de criteris d'acceptació. Aquests criteris defineixen, en part, quan aquesta història passa a estar acabada.

- **[In Progress]** - Història d'usuari ha estat assignada a un sprint.
- **[Closed]**- La historia d'usuari (i per tant les seves tasques) està implementada i funciona correctament.

En quant a les *fases de cada tasca*, el canvi principal que hem incorporat al primer sprint és l'eliminació de la fase intermitja de Ready for Test.

- **[New]** - Nova tasca creada al sprint
- **[In Progress]** - La tasca ha estat assignada a un membre de l'equip
- **[Undocumented]** - Un cop en aquesta fase, el codi que pertany a la tasca ha estat implementat i no hauria de crear problemes a la hora de compilar (dintre de la seva branca de feature)
- **[Closed]** - La tasca ha estat implementada i documentada.
  - En el nostre cas, el tema de documentar és flexible. La documentació hauria d'ajudar als altres membres de l'equip a entendre quina funció té el codi/clase/mètode i no cal que sigui exhaustiva, tot i que si que vam definir una estructura per aquells mètodes que potser requereixen més informació, sobretot si formen part d'un component que farà servir la resta de l'equip.

## 2.3. 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 tres repositoris, un per al codi del frontend, altre per al codi del backend i un últim afegit a l'*Sprint 2* per al codi de la pàgina web d'administració.

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 i de cara al primer sprint, la inicialització i creació.

La gestió dels dos repositoris seguirà la mateixa metodologia de gestió de versió, el model de control de branques anomenat **Gitflow**, però, amb una petita variació. 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 haguem captat durant el desenvolupament. En el cas del back end, aquesta és la branca que clonarà el servidor. 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/solucionar el pull request de **dev** -> **main** dels seus repositoris respectius.

En quant a la variació de Gitflow, en comptes de fer servir branques '**feature**' per a cada història d'usuari, farem servir branques '**task**'. Tot el desenvolupament relacionat a una tasca es farà dins d'aquesta branca.

El nom de les noves branques seguirà el següent conveni:

- <# num\_tasca>:<nom\_tasca>

Exemple: #99:Filtratge\_per\_nom

On *num\_tasca* i *nom\_tasca* fan referència al número al títol que té la tasca al Taiga respectivament.

Aquestes branques es crearan a partir de la branca *dev* i un cop acabades es farà un pull request a la branca *dev* (**task** -> **dev**).

Tot i que les històries d'usuari han estat redactades per tal de maximitzar la seva independència, certes funcionalitats requereixen d'una certa base per tal de funcionar correctament, si aquesta funcionalitat és present en una altra branca de *task*, hem decidit que per tal d'agilitzar el desenvolupament, podem crear branques temporals en les que treballar amb les dos *features* inacabades. Idealment, aquesta branca només existeix per agafar codi d'una tasca o història d'usuari del qual la implementació ja està acabada. En qualsevol cas, aquestes branques es tancaran un cop acabada la integració.

Pel que fa a la correcció d'errors, farem servir una metodologia semblant a les branques '*task*'; creant 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 '*task*' 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.

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

- issue/<#num\_issue>:<títol>

Exemple: issue/#123:Error\_Login\_Google

## · Commits

Finalment, els comentaris dels commits a les branques de '*task*' tindran un missatge clar i entenedor de l'estat d'aquella mateixa tasca, com per exemple:

- "Tasca finalitzada".
- "Tasca en procés. Queda afegir el codi del repositori."

## · Pull Requests (PR)

Un altre procés que va sorgir de forma espontània al primer sprint, però que volem incorporar a les nostres convencions de manera continuada, és l'ús dels *pull requests* a

l'hora d'ajuntar les diferents branques/features amb la branca de producció (**dev**) o **main** en acabar l'sprint corresponent.

Amb aquest procés assignarem un reviewer (o més) per a comprovar quins són els canvis que hi entren i resoldre els conflictes del merge, només en el cas de fer un pull request de **dev** a **main**. Els *pull requests* de branques relacionades amb desenvolupament de tasques a **dev** no caldrà assignar cap reviewer.

Els pull requests també ens permeten assignar-hi accions a executar un cop acceptada la request que executi de forma automàtica un conjunt de test per tal de garantir que el merge no ha trencat alguna altra funcionalitat.

## 2.4. Comunicació d'equip CultureFinder

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.

Apart de les hores de classe, també quedem cada diumenge per la tarda per fer una pre-reunió per parlar sobre el que s'ha fet durant la setmana, com van les tasques de cadascú i quins objectius tenim de cara a la setmana vinent. Aquestes reunions tendeixen a ser amb tot l'equip, per normalment ens acabem dividint en canals separats per tal de parlar sobre els aspectes més tècnics i ajudar-nos mútuament amb possibles problemes que haguem trobat.

Per aquestes reunions fem servir el servidor de Discord on tenim canals de veu que ens permeten compartir la pantalla. Canals de text per al front, el back, reunions, millores i consells que ens dona el profe entre d'altres.

## 2.5. 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.
- **Ràpida:** 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 **Java** com a llenguatge de programació. Java és un llenguatge de programació dissenyat el 1990 per James Gosling amb altres companys de Sun Microsystems a partir del llenguatge C. Des del seu naixement fou pensat com un llenguatge orientat a objectes. Tot i això, a l'inici vam decidir utilitzar Kotlin, un llenguatge similar a Java amb el qual tot l'equip està familiaritzat, però està més enfocat al desenvolupament d'aplicacions Android.

El principal motiu pel qual hem modificat el llenguatge del Backend a Java és perquè ens vam trobar en la situació en la qual no trobàvem molta documentació relacionada amb Kotlin i tots els membres de l'equip del Backend estàvem molt més familiaritzats amb Java.

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 tenia 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.

A més del *frontend* i el *backend* també contem amb una **pàgina web** on els usuaris que son administradors poden gestionar les incidències que creen els usuaris de la nostra aplicació.

Les tecnologies utilitzades a aquesta pàgina web son HTML, Javascript i CSS. A més fem servir l'arquitectura model-view-controller que separa la lògica de l'aplicació en tres components: el model, la vista i el controlador. El model representa les dades i la lògica de de l'aplicació, la vista és la interfície d'usuari i el controlador maneja les interaccions entre el model i la vista.

HTML (Hypertext Markup Language) és el llenguatge de marcatge estàndard utilitzat per crear l'estructura i el contingut d'una pàgina web. És molt utilitzat en el desenvolupament d'aplicacions web a causa de la seva facilitat d'ús i el seu ampli suport pels navegadors web.

JavaScript, d'altra banda, és un llenguatge de programació dinàmic que s'utilitza per afegir interactivitat a les pàgines web. És molt popular en el desenvolupament web perquè pot executar-se al navegador de l'usuari i és compatible amb la majoria dels navegadors.

CSS (Cascading Style Sheets) és un llenguatge de fulls d'estil que s'utilitza per definir l'aparença visual d'una pàgina web. Permet separar la presentació d'una pàgina web del contingut, el que facilita el seu manteniment i actualització.

Hem escollit aquestes tecnologies per la seva àmplia adopció en el desenvolupament web, la seva facilitat d'ús i la seva capacitat per proporcionar una experiència d'usuari efectiva i eficient.

## 2.6. Llibreries

### 2.6.1 Mapstruct

Les aplicacions multicapa sovint requereixen un mapatge entre diferents models d'objectes (per exemple, entitats i DTOs). A l'hora de desenvolupar el codi específic per poder fer el mapeig, com es menciona a la secció d'arquitectura, vam utilitzar el framework **MapStruct**, ja que escriure aquest codi d'assignació és una tasca tediosa i propensa a errors.

MapStruct és una llibreria que ens permet fer mapeig d'objectes sense haver d'escriure tot el codi a mà, simplement amb una interfície, en la qual li indicarem l'objecte d'entrada i l'objecte de sortida.

MapStruct el que ens permet és, mitjançant aquesta llibreria, crear la interfície, i amb l'etiqueta `@Mapper` simplement generar el mètode, definint l'objecte que volem retornar i l'objecte que li volem passar per paràmetres per a poder convertir-lo. Una vegada fet, ens implementarà aquesta interfície una vegada compili l'aplicació, sense nosaltres haver de fer feina extra.



A nivell d'usabilitat és bastant fàcil, el únic punt que se'ns va complicar una mica al principi va ser el tema de la configuració de dependències, però finalment vam poder utilitzar-la sense més complicacions.

### 2.6.2 Jackson

Jackson és una llibreria de Java que s'utilitza àmpliament en el desenvolupament de Spring Boot per gestionar la conversió d'objectes Java a formats JSON (JavaScript Object Notation) i viceversa. Jackson proporciona eines per a l'anàlisi i la generació de JSON, el que fa que sigui fàcil treballar amb dades JSON en una aplicació Spring Boot.

Concretament, utilitzem dues funcionalitats que ofereix Jackson:

- Deserialització: Jackson permet convertir dades JSON en objectes Java utilitzant el procés de deserialització. Aquest procés implica llegir un JSON i mapejar-lo automàticament a les propietats corresponents d'un objecte Java.
- Anotacions: Jackson proporciona un conjunt d'anotacions que es poden utilitzar en les classes Java per personalitzar el procés de conversió. Per exemple, pots utilitzar l'anotació `@JsonIgnore` per ignorar una propietat en la serialització o deserialització, o l'anotació `@JsonProperty` per canviar el nom de la propietat JSON associada a una propietat Java.

A més, també vam pensar en utilitzar la serialització per a convertir objectes Java a JSON i enviar-los com a resposta a les crides de la API, però Spring Boot et permet enviar objectes directament que es converteixen automàticament a JSON.

### 2.6.3 Liquibase

Liquibase és una eina que s'utilitza en el desenvolupament de Spring Boot per gestionar els canvis en la estructura de la base de dades. Proporciona un sistema de control de versions per a la base de dades, permetent als desenvolupadors gestionar i aplicar canvis de manera controlada i reproducible.

Utilitzem Liquibase principalment per a fer migracions de la base de dades sense pèrdua de dades, i addicionalment per a tenir un control de versions de la base de dades durant el temps(aquesta llibreria la hem afegir a l'sprint 3 però considerem que és una bona pràctica tenir-la). El funcionament és molt senzill, i és que mitjançant l'IntelliJ amb el plugin JPA Buddy podem generar un changelog automàticament de la base de dades respecte la antiga versió, i indicar a l'arxiu `application.properties` que l'última versió de la

base de dades correspon a aquest changelog. D'aquesta manera, si es veu un canvi respecte al model que presenta el changelog, liquibase automàticament fa la migració a l'estat que l'arxiu presenta.

#### 2.6.4 Lombok

Lombok és un projecte que va néixer l'any 2009 que, mitjançant contribucions, ha anat guanyant en riquesa i varietat de recursos. És una llibreria per a Java que a través d'anotacions ens redueix el codi que codifiquem, és a dir, ens estalvia temps i millora la llegibilitat d'aquest. Les transformacions de codi que realitza es fan en temps de compilació.

Per mencionar un exemple, a l'hora d'implementar les entitats de domini, utilitzant les anotacions `@Getter` i `@Setter` per a generar automàticament els mètodes corresponents. Els mètodes `Get` i `Set` seran públics tret que s'especifiqui passant-li un paràmetre a l'anotació a través de la classe `AccessLevel`.

#### 2.6.5 Openapi

OpenAPI és una especificació que defineix un estàndard per a la descripció d'APIs RESTful. Proporciona un conjunt de convencions i formats per descriure els recursos, les operacions, els paràmetres, les respostes i altres aspectes d'una API. L'objectiu principal d'OpenAPI és facilitar la comprensió i integració de les API per part de desenvolupadors, equips de treball i eines.

En el context de Spring Boot, la llibreria OpenAPI permet generar una documentació detallada i interactiva de l'API basada en les anotacions i configuracions dels controladors de Spring. Aquesta documentació es pot visualitzar utilitzant Swagger UI, una interfície web que permet explorar i provar l'API directament des del navegador.

Les anotacions com "`@Api`", "`@ApiOperation`", "`@ApiParam`", "`@ApiResponse`", entre d'altres, s'utilitzen per descriure les classes i mètodes dels controladors i els seus paràmetres, respostes i altres detalls. En temps d'execució la llibreria d'Openapi genera un arxiu `.yaml` o `.xml` que conté tota la especificació de la nostra API seguint aquest estàndard, i que es pot visualitzar mitjançant la url <http://nattech.fib.upc.edu:40380/api-docs>.

Amb Swagger UI, que s'integra amb Springfox Swagger, es pot accedir a la documentació generada visitant una URL específica de l'aplicació(<http://nattech.fib.upc.edu:40380/swagger-ui/index.html>). Aquesta interfície web permet explorar l'API, veure els endpoints disponibles, els paràmetres, les

respostes, els models de dades i fins i tot provar els endpoints directament des del navegador.

#### 2.6.6 Json Web Token

La llibreria JSON Web Token (JWT) per a Spring Boot és una eina que facilita la generació, verificació i gestió de tokens JWT en una aplicació basada en Spring Boot.

JSON Web Token és un estàndard obert (RFC 7519) que defineix un format compact i segur per a representar reclamacions entre dues parts en forma de token. Un token JWT consisteix en una cadena de caràcters que està format per tres parts separades per punts: la capçalera (header), les reclamacions (claims) i la firma (signature).

Aquesta tecnologia és àmpliament utilitzada en arquitectures d'autenticació i autorització basades en tokens. Un cas d'ús típic és l'autenticació d'usuari i l'enviament de tokens JWT com a mitjà de prova d'autenticació en cada sol·licitud a l'API.

Concretament, nosaltres fem servir dues funcionalitats:

- Generació de tokens JWT: La llibreria facilita la creació de tokens JWT signats mitjançant una clau secreta. Aquesta clau secreta s'utilitza per assegurar la integritat del token i verificar la seva autenticitat. El token pot contenir informació específica (reclamacions) com ara l'identificador de l'usuari, els seus rols, el temps d'expiració, etc.
- Decodificació de tokens JWT: Aquesta llibreria també facilita la decodificació de tokens JWT sempre i quan s'utilitzi el mateix algorisme de codificació i clau secreta. Podem verificar la seguretat ja que els tokens són nostres, així que podem decodificar-los sense cap problema i extreure la informació que guarden, que, en el nostre cas, és l'identificador dels usuaris.

#### 2.6.7 JSoup

Jsoup es una llibreria utilitzada al backend que es fa servir per parsejar les dades que ens arriben de la API de Dades Obertes, particularment per el camp de la descripció, el qual en molts dels events conté etiquetes HTML i altres artefactes.

#### 2.6.8 Firebase

La llibreria de Firebase per a Flutter és un conjunt d'eines i serveis que permeten als desenvolupadors integrar Firebase en les seves aplicacions mòbils desenvolupades amb Flutter.

Firebase ofereix una àmplia gamma de serveis, i la llibreria de Flutter proporciona enllaços i API específiques per a utilitzar aquests serveis des d'una aplicació Flutter de manera senzilla. Concretament per la nostra aplicació utilitzem els següents serveis, que descrivim en detall a l'apartat d'APIs:

- Firebase Authentication: Permet autenticar usuaris utilitzant mètodes com a correu electrònic/contrasenya, autenticació amb Google, Facebook, Twitter, entre altres.
- Firebase Cloud Messaging (FCM): Permet enviar notifikacions push als dispositius mòbils i mantenir als usuaris actualitzats amb informació rellevant.
- Firebase Storage: Proporciona emmagatzematge en el núvol per a arxius multimèdia com a imatges, vídeos i documents.

Aquesta llibreria ens simplifica la integració d'aquests serveis proporcionant-nos un conjunt de widgets, mètodes i classes que ens faciliten el seu ús.

#### 2.6.9 Shared Preferences

Shared Preferences és un plugin que permet emmagatzemar dades de manera persistent en una plataforma. Permet guardar tipus de dades senzills, concretament int, double, bool, String i List<String>. És compatible amb diversos sistemes operatius, com Android a partir de la versió SDK 16. Ens va interessar per això, ja que treballem per un sistema Android.

Hem utilitzat Shared Preferences per guardar dades de l'usuari, concretament l'API-token de cada usuari quan aquest fa login.

#### 2.6.10 Table Calendar

Table Calendar es un paquet de flutter que conté un widget de calendari molt customitzable i amb un gran número de funcions per calendaris. Permet mostrar el widget del calendari en diversos formats, com mes a mes o setmana a setmana. Té funcions per seleccionar dies, i per mostrar dades relacionades amb un dia concret.

Hem utilitzat el Table Calendar per implementar els dos calendaris que tenim a l'aplicació. Hem fet servir el widget de calendari, mostrant tot un mes, i la selecció de dies, començant amb el dia actual. Per mostrar els esdeveniments hem aprofitat la funcionalitat de mostrar dades d'un dia.

Vam escollir aquest paquet de calendari entre d'altres perquè permetia molta customització i és senzill d'utilitzar.

### 2.6.11 Riverpod

Aquest package és una de les solucions de maneig d'estat que existeixen per Flutter. Quan utilitzar correctament, permet compartir “estats” entre widgets a diferents punts de “l'arbre” de widgets amb el qual s'ordena Flutter. Tot i que *Riverpod* és una solució molt completa per manejar estat i comunicar-se entre vistes, no la vam fer servir molt ja que afegia molta complexitat, sobretot per algunes de les funcionalitats més aïllades.

Aquest package el vam acabar fent servir principalment a la vista de la llista d'esdeveniments, ja que facilitava les crides per actualitzar la UI.

### 2.6.12 Google Maps for Flutter

És un plugin que permet obtenir el widget del mapa de Google Maps. Aquesta llibreria utilitza crides a la API de Google Maps, que hem explicat a l'apartat 3.5.2.1. Google Maps API.

L'utilitzem perquè facilita un widget de mapa i a més té funcionalitats de Google Maps, i dona suport per a Android a partir de la versió SDK 20. Utilitzem el widget per mostrar el mapa amb events de la nostra aplicació.

### 2.6.13 Google Maps Cluster

Aquesta llibreria té funcions que permeten fer clustering d'items en un mapa de Google Maps. Requereix un widget de Google Maps, per tant depèn de la llibreria Google Maps for Cluster, explicada a l'apartat anterior (2.6.13. Google Maps for Cluster).

Utilitzem aquesta llibreria per fer clustering dels events que apareixen en el mapa, agrupant-los quan es fa zoom out per tal de que es mostrin de manera més organitzada i visible.

### 2.6.14 Geolocator

Aquesta llibreria permet cridar a la API de Geolocator, per obtenir l'ubicació del dispositiu. L'utilitzem per mostrar l'ubicació actual al mapa.

Hem explicat millor la API a l'apartat 3.5.2.3 Geolocator API.

### 2.6.15 Permission Handler

Aquesta llibreria ens permet gestionar els permisos en aplicacions mòbils de manera senzilla. Concretament, en el nostre cas, ens permet sol·licitar i verificar fàcilment permisos d'ubicació a Android. També s'encarrega de mostrar els quadres de diàleg de sol·licitud de permisos nadius i ofereix opcions de configuració.

Resulta útil per garantir una experiència d'usuari adequada i complir amb les polítiques de privacitat i seguretat. L'hem utilitzada per poder accedir a la ubicació del dispositiu per la funcionalitat del mapa.

#### 2.6.16 Google Sign In

És un plugin que permet accedir a Google Sign In. Dona suport a Android a partir de la versió SDK 16. Necessita alguna API per cridar a les funcionalitats de Google Sign In; en el nostre cas, utilitzem la API de Firebase Authentication.

Hem utilitzat el Google Sign In per poder fer login a l'aplicació mitjançant Google.

#### 2.6.17 Image Picker

És una llibreria que permet agafar imatges guardades al teu dispositiu per utilitzar a l'aplicació, i també fer fotografies utilitzant la càmera. Dona suport a Android des de la versió SDK 21.

Aquesta llibreria la utilitzem per seleccionar imatges per posar com imatge de perfil d'un usuari.

### 2.7. IDEs

Per facilitar el desenvolupament, hem escollit tres 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 inconsistentes 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, el framework de Flutter requereix Android Studio per descarregar i instal·lar el SDK de Android, per això 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.

Per a la web admin, hem utilitzat WebStorm, que és l'equivalent de IntelliJ però per a desenvolupament web fent servir JavaScript.

Com ja hem esmentat, aquests IDEs ja compten amb plugins de formateig de codi per a una major legibilitat, així com l'intellisense, el qual suggereix al programador canvis en el codi per a millorar la sintaxis, el rendiment, o estalviar l'ús de funcions *deprecated*, entre altres. A més, tots ells venen amb Git inclòs, cosa que ens facilita la creació i actualització de branques, i a l'hora de fer *pull*, *merge*, *fetch*, *commit* o *push*.

## 2.8. 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 permet comprovar la interfície en diferents dispositius amb diferents dimensions 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. També hem trobat l'eina [\*'scrcp'\*](#) 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.

En quant al *testing*, en un principi vam fer-ne alguns tests unitaris, però per falta de temps hem decidit que abans de tancar qualsevol tasca, provarem les funcionalitats afegides amb aquesta tasca de manera aïllada i a continuació provarem la resta de funcionalitats que s'integren amb aquesta nova. Per últim realitzem proves a nivell de sistema per comprovar que no han sorgit errors on no toca.

Tots els membres hem utilitzat GitHub Copilot, que ens ha ajudat a generar funcions i línies de codi, tot i que no ha resultat una eina sempre eficaç ja que a vegades genera codi que s'ha de rectificar manualment.

## 2.9. 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.

## 2.10. Convencions de codi

Hem decidit que no tindrem convencions concretes a la hora d'escriure el codi ja que els llenguatges que utilitzem tenen les seves pròpies convencions i bones pràctiques a l'hora d'escriure codi. En canvi, aquestes convencions son més una sugerencia i al final els estils dels dos codebases seran el més consistens possibles de forma independent. És a dir, la consistència ens mantindrà entre el codi del front i per altra banda, el codi del back

A continuació deixem explicades aquestes pràctiques.

### 2.10.1. 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/';

### 2.10.2. 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>** : Si no fos prou evident, 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.

### 2.10.3. 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

### 2.10.4. Fitxers i directoris

Sobre els fitxers (que no són classes) i els packages farem servir camelCase

**Ex:** unDirectorio/unAltreDirectorio/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.

## 2.11. Gestió de bases de dades

Per la part de la base de dades farem servir PostgreSQL. La desplegarem a **virtech** de la mateixa manera que el backend.

Al principi del nostre projecte teníem pensat utilitzar Amazon Web Services (AWS) per fer el deploy tant de la base de dades com de la nostra aplicació. Tanmateix, quan al primer sprint ens vam posar a configurar la base de dades i creant el compte a AWS, ens vam adonar que havia un límit d'hores d'execució si volíem utilitzar la versió "gratuïta". Per aquest mateix motiu, i gràcies a les importants facilitats que ens proporciona el servei de **virtech**, és un entorn de virtualització de màquines virtuals sota demanda. Aquest servei permet crear, configurar, modificar i esborrar servidors virtuals Linux a un grup d'estudiants. Cada grup pot configurar aquests servidors a mesura de les seves necessitats i adaptar-lo per a desenvolupar les seves pràctiques. D'aquesta manera els alumnes de la UPC disposem d'un (o varis) servidors propis en el "núvol" on es pot programar qualsevol servei, amb l'avantatge que aquest servei està disponible des de tota internet.

Hem escollit PostgreSQL perquè és un servei amb el qual tots estem familiaritzats. A l'hora de configurar la base de dades a **virtech** tindrem en compte els permissos d'usuari i d'accés. També seleccionarem l'opció d'accés públic per tal de poder accedir des dels nostres ordinadors i no utilitzar cap mena de VPN.

No ha sigut fins al començament de l'Sprint 3 que hem tingut un problema amb la base de dades: cada cop que alteràvem una taula existent, afegint una columna o eliminant una altra, hibernate ens havia d'esborrar el contingut de totes les taules. Això comportava un problema de pèrdua de dades que, si més no, no és un gran problema en etapes de desenvolupament. Malgrat això, per a evitar pèrdua de dades en un futur i donat que és una bona pràctica, hem afegit liquibase al projecte(veure apartat 2.6.3). Aquesta decisió permet a un equip de desenvolupament fer canvis en l'esquema de la base de dades un cop l'aplicació està en funcionament, sense perdre cap dada de valor.

## 2.12. Gestió de bugs

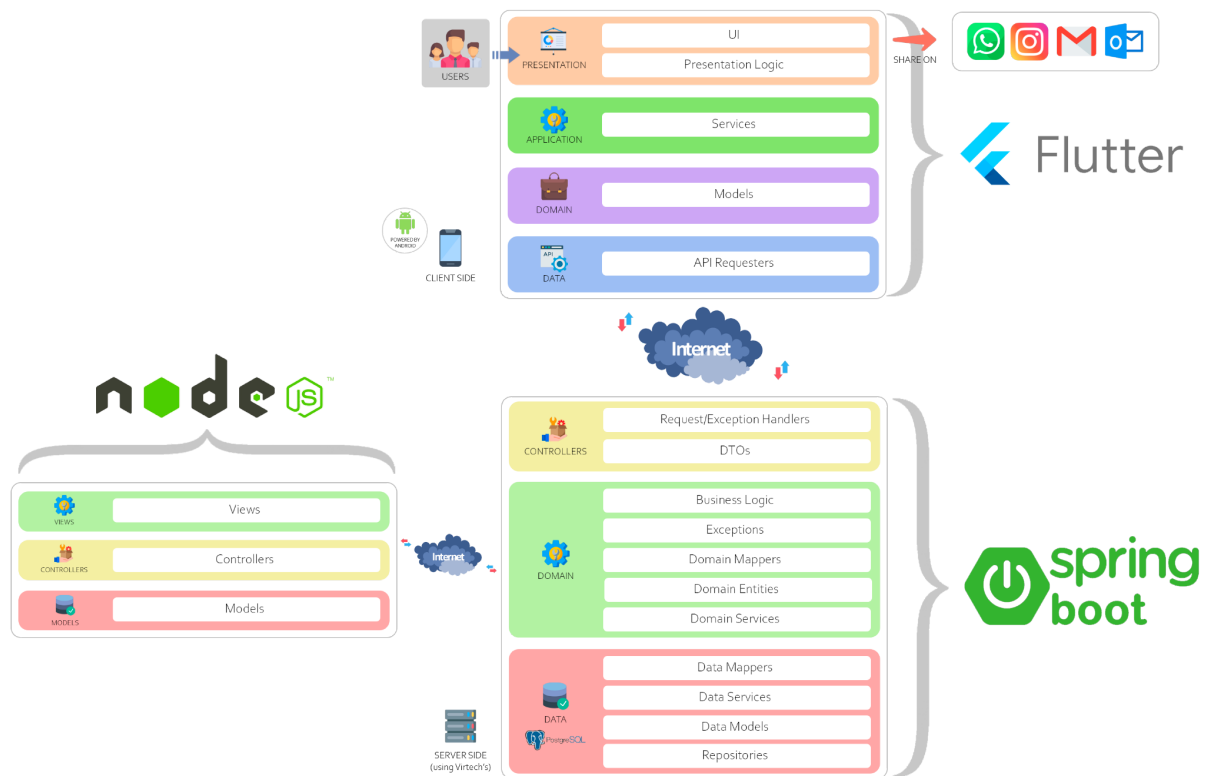
No tenim un sistema organitzat per detectar i solucionar bugs, però sí prioritzàvem solucionar-los.

Quan algú detectava un bug que no sabia resoldre, informàvem a la resta de l'equip a través del Discord que compartim, i també en WhatsApp per comunicar-ho el més ràpidament possible. Si algú altre de l'equip sabia com solucionar el problema, els membres necessaris de l'equip es connecten en un canal de veu de Discord, o en presencial si era possible, i es feia pair programming per solucionar l'error. El pair programming no era estricte, i, tot i que sovint era el membre que ha tingut l'error amb un altre del seu end (frontend o backend), a vegades ajudava un tercer membre o el que ajudava era d'un altre end perquè era un error intermig.

Si era un error important o no es solucionava ràpidament, es feia un commit indicant a la descripció el problema que s'ha solucionat.

### 3. Descripció tècnica

#### 3.1. Concepte de l'arquitectura



##### 3.1.1. Arquitectura física

Hem decidit fer l'aplicació dividint-la en tres: *frontend*, *backend* i *admin web*.

Pel que fa al *frontend*, fem servir una adaptació de la arquitectura en tres capes fent servir el patró de MVVM:

- **Presentació:** Aquesta capa conté principalment les classes pertanyents al elements de la UI (Widgets) i el seu estat i controladors. Aquesta capa conté les Views i alguns Notifiers (controladors).
- **Aplicació:** En aquesta capa es troben els diferents serveis encarregats de treballar amb els nostres propis model de les dades que agafem del backend. Aquesta capa i la implementació del que serien els *viewmodels* del patró no la tenim molt clara a la hora d'implementar.
- **Domini:** Aquesta capa representa la nostra versió de la estructura de dades que tenim guardades al backend. Aquí és on trobem els diferents Models que utilitzem. Hem decidit que la conversió dels DTOs (principalment en JSON) a

instàncies es troba directament al Model, per tal de concentrar les funcionalitats en un punt.

- **Dades:** Aquesta capa encapsula la d'adquisició de dades per construir els components de la UI o la lògica de la capa d'aplicació.

Pel que fa al *backend*, programat utilitzat el framework Spring Boot amb Java, l'hem dividit en tres capes: Controladors, Domini i Dades. La capa dels controladors s'encarrega de proporcionar les dades al *frontend* mitjançant la nostra pròpia API, així com obtenir les dades de la API de l'Agenda Cultural mitjançant uns objectes que anomenem DTOs(*Data Transfer Object*).

La capa de domini conté les classes de lògica per a cada tipus de model de dades, les quals s'encarreguen de processar les dades que representen aquests models. D'altra banda, conté també les entitats, que són les classes que representen els models de dades d'informació amb les que treballem. A més, també conté els serveis de domini, els quals s'encarreguen d'interactuar amb els de dades. Per últim, conté les excepcions que hem definit nosaltres.

La capa de dades conté les classes de models, on definim les taules que contenen els models de dades amb els que treballem. També les classes de servei, que són aquelles que proveeixen al backend d'interacció senzilla amb la base de dades. Aquestes classes de servei utilitzen els repositoris, interfícies que s'encarreguen de mapejar mètodes directament a sentències SQL sobre les taules generades a partir dels models.

Cal remarcar que les capes de domini i dades tenen un *package* de *mappers*. Dins d'aquests *packages*, guardem les classes que s'encarreguen de fer conversions **DTO <-> Entity** (*Mappers* de domini), i **Entity <-> Model** (*Mappers* de dades). Tota la lògica d'aquesta implementació ha sigut facilitada pel framework de **Mapstruct**, amb el qual només introduint les capçaleres de cada mètode i introduint les dependències de Gradle amb les anotacions corresponents vam poder aconseguir fer el mapeig sense cap problema.

La base de dades, implementada mitjançant un sistema de gestió de models relacionals(PostgreSQL), romandrà dins d'una instància d'una màquina virtual allotjada a Virtech, un servidor de la UPC, així com la *web admin* i l'aplicació *backend*, que estaran en constant execució per tal de poder proveir al *frontend* de la informació i la interacció amb la BD necessàries i per a poder gestionar una part de l'aplicació.

### 3.1.2. Patrons d'arquitectura aplicats

Hem aplicat una arquitectura en tres capes per cadascuna de les dues aplicacions, *frontend* i *backend*, mentre que per a l'aplicació web hem decidit fer-la utilitzant MVC. En quant a l'arquitectura del *frontend* i del *backend*, hem decidit dissenyar-la d'aquesta manera donat que:

- La interacció entre diferents components només ocorre *intra-capa* o entre capes veïnes.

- Aquest disseny permet no fer assumpcions sobre com són els tipus de dades que s'utilitzen.
- Cada capa té un grup de funcionalitats perfectament definides. Això fa que sigui senzill determinar on anirà un mètode, classe o atribut.
- La capa de dades és perfectament reutilitzable: conté funcions i mètodes senzills.
- En el nostre cas, la capa de dades també compta amb completa independència respecte a la tecnologia utilitzada(PostgreSQL).

## 3.2. Capa de domini

### 3.2.1. Patrons de dissenys aplicats

#### 3.2.1.1. Observer

Al front end, un dels patrons que més hem utilitzat és el patró observador. Aquest consisteix en un objecte que manté una llista dels seus “observadors”, quan l'objecte és modificat, aquest s'encarrega d'avisar els seus observadors.

En particular, aquest patró és fundamental al funcionament dels *notifiers* de Flutter. La vista de la llista d'events és un bon exemple on s'aplica el patró mitjançant la llibreria de Riverpod.

#### 3.2.1.2. Factory

Un altre patró que hem aplicat principalment al frontend és el patró factoria. Aquest patró consisteix en delegar la creació d'objectes a mètodes d'una classe factoria. En el cas de *Dart*, aquest accepta la creació de funcions de tipus factoria, els quals vam utilitzar principalment en la creació dels nostres models a partir del diccionari JSON que ens retorna la API.

Un altre lloc on l'hem fet servir al front és a la hora de crear les “instàncies” del nostre model per a les etiquetes. Aquesta factoria (ahora un singleton), s'encarrega de fer una crida al back la primera vegada, i guardar totes les etiquetes per les quals es poden filtrar els events.

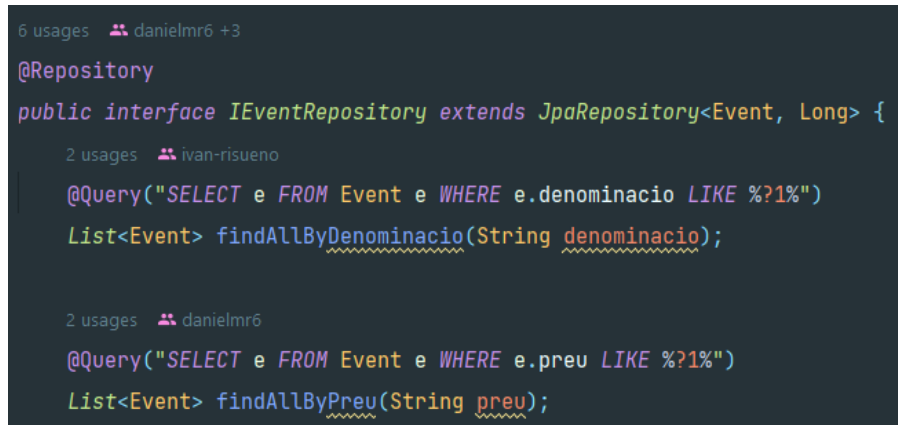
#### 3.2.1.3. Repository

De manera general, aquest patró permet que dos components amb interfícies diferents puguin col·laborar. En el nostre cas hem afegit interfícies a la capa de dades per a que les classes de serveis puguin utilitzar mètodes que interactuen amb la BD, fent ús d'aquestes interfícies (classes de tipus *Repository*) operacions de les quals s'implementen de manera automàtica.

Trobem aquest patró útil i necessari ja que ens permet implementar consultes d'SQL utilitzant la JPA(Java Persistent API) sense cap mena d'esforç. Aquesta API funciona de la manera següent: donada una classe Servei que vol fer ús de la BD, només cal crear una interfície de tipus Repositori, que hereti de *JpaRepository*, i definir mètodes sobre els quals afegirem l'anotació *@Query* amb la consulta SQL en concret.

Això es fa per a obtenir una capa extra d'abstracció en els components corresponents, evitant la repetició del codi (principi DRY), facilitant el testatge i desacoblant-lo de la lògica de negoci.

Gràcies a l'ús d'aquest repositori, concretament a la nostra interfície anomenada *IEventRepository.java*, existeix una major independència entre la capa de dades i la resta de l'arquitectura, ja que si es produeix alguna modificació en concret es podrà rebre a partir de les seves crides utilitzant les consultes SQL i pel cas de voler fer un refactor de la crida a la capa de dades només caldria modificar aquest fitxer en concret. A continuació mostrem una part de la interfície que hem explicat:

A screenshot of a code editor showing the *IEventRepository* interface. The code is in Java and uses Spring Data JPA annotations. It shows the interface extending *JpaRepository<Event, Long>* and defining two methods: *findAllByDenominacio* and *findAllByPreu*, each with a corresponding *@Query* annotation. The code is color-coded and includes usage statistics for each line.

```
6 usages  danielmr6 +3
@Repository
public interface IEventRepository extends JpaRepository<Event, Long> {

    2 usages  ivan-risueno
    @Query("SELECT e FROM Event e WHERE e.denominacio LIKE %?1%")
    List<Event> findAllByDenominacio(String denominacio);

    2 usages  danielmr6
    @Query("SELECT e FROM Event e WHERE e.preu LIKE %?1%")
    List<Event> findAllByPreu(String preu);
```

#### 3.2.1.4. Singleton

La finalitat d'aquest patró és que una classe pugui arribar a tenir com a màxim una instància, i que aquesta instància sigui accessible de manera global. A més, també evita que el valor d'aquest objecte es sobreescrigui ja que l'accés és només de lectura.

Aquest patró ens és útil ja que a la nostra aplicació utilitzem constantment variables de classes que s'encarreguen de desenvolupar una activitat com pot ser crear una instància d'un objecte, modificar els atributs d'un objecte, guardar a la base de dades un objecte, etc. Aquests tipus de classes ens és convenient utilitzar-les sense crear diferents instàncies, ja que l'estat de les mateixes no es modifica.

Gràcies a Spring Boot i la seva característica més important, la *Injecció de dependències*, podem indicar que un atribut d'una classe ha de ser una dependència injectada mitjançant la notació *@Autowired*. Aquesta notació indica a un contenidor que té Spring per a instanciar automàticament una dependència (una classe normal i corrent) en temps de creació de la classe que fa servir aquesta dependència, a més de fer-la servir allà on calgui sense generar instàncies noves.

Sense ser una aplicació directa del patró *Singleton*, trobem que el fem servir indirectament quan anotem atributs de tipus *IDomainXService*, *IXService*, o *IXRepository*. Aquestes classes, com ja hem esmentat, són classes merament funcionals, que desenvolupen accions sobre el sistema i sobre les dades que formen el sistema i, per tant, no tenen atributs com a tal més enllà d'altres dependències injectades amb *@Autowired*.

```
/*
 * Descripció: Instància del servei que s'encarrega d'interactuar amb el repositori d'incidents.
 */
@Autowired
private IIncidentService iIncidentService;

/*
 * Descripció: Instància del servei que s'encarrega d'interactuar amb el repositori d'users.
 */
@Autowired
private IUserService iUserService;
```

### 3.2.1.5. DTO

El patró *Data Transfer Object* té sentit quan, a una aplicació, s'intercanvien dades entre un client i un servidor. Aquestes dades acostumen a ser dades derivades, calculades o generades d'altra manera a partir d'unes dades existents. En definitiva, la informació que s'envia comunment no representa a la perfecció cap model de dades que utilitza el sistema.

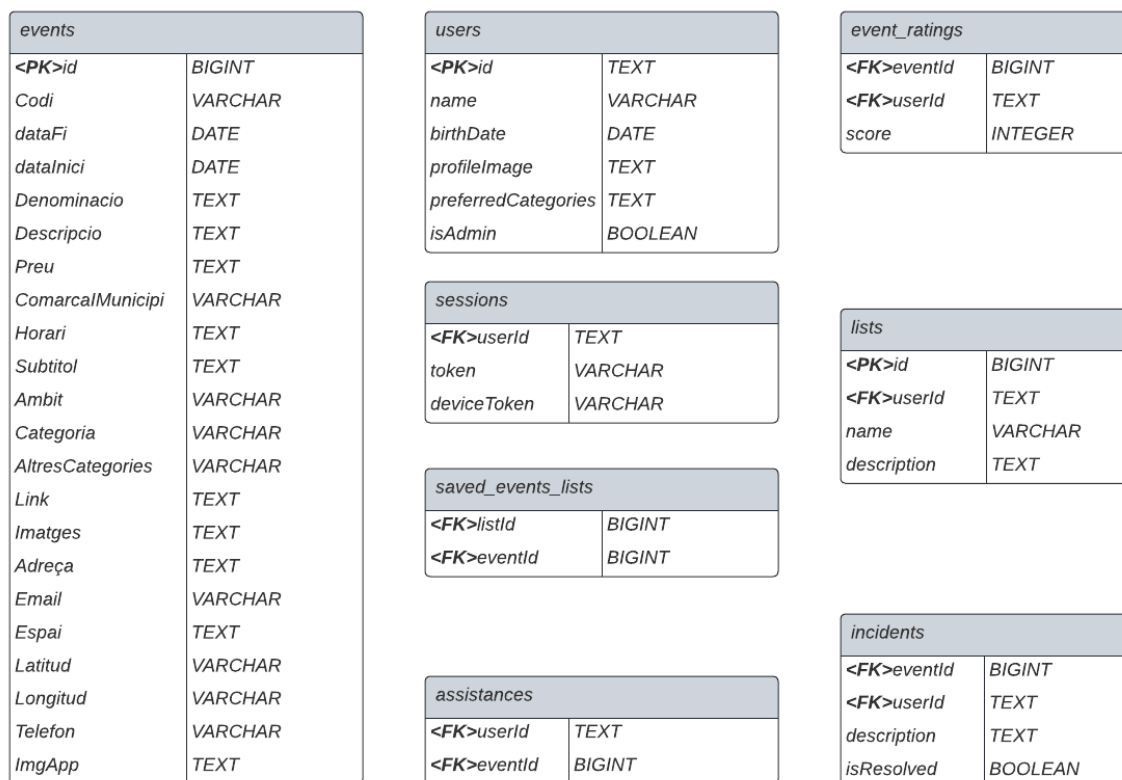
És per això que aquest patró consisteix en la creació d'unes classes auxiliars, que s'utilitzen per a intercanviar informació client-servidor, i que contenen atributs dels models de dades, i altres atributs d'altres models o derivats.

En concret, el nostre *backend* utilitza aquest patró per a enviar informació de més d'una taula en concret de la base de dades, com pot ser el cas de les llistes de favorits (no consten en la mateixa taula els esdeveniments que les formen), o informació calculada dels esdeveniments com per exemple el número d'assistents (taula d'assistències) o la valoració mitjana (taula valoracions).

```
public class EventDTO {
    private Long id;
    private LocalDate dataFi;
    private LocalDate dataInici;
    @NotBlank
    private String denominacio;
    @NotBlank
    private String descripcio;
    private String preu;
    private String horari;
    private String subtitol;
    @NotBlank
    private String ambit;
    private String categoria;
    private String altresCategories;
    private String link;
    private String imatges;
    private String adreca;
    private String comarcaIMunicipi;
    private String email;
    private String espai;
    private Float latitud;
    private Float longitud;
    private String telefon;
    private String imgApp;
    private Float score;
    private Integer numberOfAssistants;
}
```

Com ja hem dit, aquest patró ens és útil ja que l'aplicació mòbil *frontend* necessita mostrar més d'un tipus de dades (entenent tipus com a model o concepte) en un context determinat, així com enviar al *backend* informació que cal guardar però que no representa un model de dades al 100%.

### 3.3. Diagrama de la base de dades (UML)



**Nota (I):** No hem afegit les línies que representen claus foranes per simplicitat.

**Nota (II):** Les claus primàries de les taules que referencien a altres taules estan compostes per totes les seves claus foranes.

### 3.4. Llistat de les tecnologies

#### 3.4.1. Visió general dels servidors

Els professors ens han subministrat un usuari i una contrasenya per a accedir a Virtech, una aplicació web de la UPC que permet allotjar servidors com a màquines virtuals que corren instàncies de Ubuntu (Linux).

Nosaltres, per a la nostra aplicació, utilitzem dos servidors:

- Un servidor que executa l'aplicació Backend.
- Un servidor que executa l'aplicació web de la pàgina d'administradors.

Per altra banda també utilitzem Firebase com a servidor cloud per guardar les imatges dels usuaris de la nostra aplicació. Aquests servidors estan dissenyats per a garantir la



disponibilitat i durabilitat dels arxius, cosa que significa que els arxius estaran disponibles i segurs fins i tot en cas de fallades en un servidor o una ubicació específica.

Firebase Storage utilitza un sistema basat en buckets per a organitzar i estructurar els arxius. A més de l'emmagatzematge en si, Firebase Storage també proporciona característiques addicionals, com a regles de seguretat que permeten controlar qui té accés als arxius i en quines condicions, i funcions de processament d'imatges, que et permeten fer tasques com redimensionar, comprimir o generar miniatures de les imatges emmagatzemades.

Concretament el nostre servidor és el següent:

<https://console.firebase.google.com/project/culturefinder-3a452/storage/culturefinder-3a452.appspot.com/files>

### 3.4.2. Configuració dels servidors

El servidor de *Virtech*, com ja hem comentat, corre una instància d'Ubuntu 20.04 i té 4GB de memòria RAM, més que suficient per a l'aplicació backend i la base de dades.

### 3.4.3. Bases de dades

#### 3.4.3.1. Backend

La màquina virtual del *backend* té instal·lat *Postgresql 14.7*. Amb aquesta *fresh install*, la única cosa que hem hagut de fer ha sigut definir que la contrassenya per a l'usuari per defecte, el qual és *postgres*, seria 1234. Amb aquest usuari el *backend* es connectarà a la base de dades i realitzarà creacions de taules, alteracions o migracions i qualsevol tipus d'inserció, modificació o esborrat de tuples.

La base de dades, també es diu *postgres*, i allotjem les nostres taules dins del catàleg *public* (és el catàleg per defecte). A aquesta base de dades guardem tota la informació neta de l'Agenda Cultural i la informació que correspon als nostres models, amb totes les dades processades de la millor forma per facilitar la implementació a l'equip del *frontend*.

Pel que fa a l'aplicació, hem hagut d'instalar Gradle i Java 17 per a poder executar-la.

#### 3.4.3.2. Admin web

Al principi vam decidir que la admin web tindria una màquina virtual per a executar-se, però per falta de temps no hem pogut desplegar-la i, per tant, al nostre compte de *Virtech* només hi ha una màquina virtual.

### 3.4.3.3. Frontend

*explicar la BD de firebase*

## 3.4.4. Llenguatges utilitzats

En quant als llenguatges, fem servir SQL per a la base de dades, Dart per al *frontend*, Java per al *backend*, Kotlin per a les dependències especificades als arxius de *Gradle*, i *JavaScript* per a la *web admin*.

### 3.4.4.1. SQL (PostgreSQL)

SQL (Structured Query Language) és un llenguatge de programació dissenyat per a gestionar bases de dades relacionals. És àmpliament utilitzat en la indústria per a interactuar amb sistemes de gestió de bases de dades (SGBD) com ara MySQL, PostgreSQL, Oracle, Microsoft SQL Server, entre d'altres.

SQL permet als desenvolupadors i administradors de bases de dades realitzar diverses operacions essencials. Per exemple, es pot utilitzar SQL per a crear bases de dades i definir l'estructura de les taules que emmagatzemen les dades. També permet inserir dades en les taules existents i consultar dades mitjançant comandes de consulta, com ara SELECT, WHERE, GROUP BY i ORDER BY. També ofereix la capacitat d'actualitzar i eliminar dades existents en una base de dades mitjançant comandes com UPDATE i DELETE. A més, proporciona eines per a gestionar l'accés a les bases de dades, incloent la gestió de permisos i controls d'accés.

Aquest llenguatge ens és de molta utilitat per a interaccionar amb la nostra base de dades, la qual està gestionada per *Postgresql*, i trobem que era la millor opció pel que fa al tractament de la BD.

### 3.4.4.2. Dart (Flutter)

Dart és un llenguatge de programació desenvolupat per *Google*, dissenyat per a la construcció d'aplicacions mòbils i web. És orientat a objectes i estàticament tipat, que ofereix una sintaxi clara i concisa per al desenvolupament d'aplicacions eficients i escalables. Una de les característiques clau de Dart és el suport per a programació asíncrona, que permet gestionar tasques asíncrones com ara crides a API i operacions de xarxa de manera eficient. Això es realitza mitjançant futures i streams.

Una altra característica important de Dart és que també promou la reutilització de codi a través de l'arquitectura basada en components i el suport per a paquets. Això permet

als desenvolupadors escriure codi una sola vegada i utilitzar-lo en múltiples plataformes, com ara Android, iOS i web.

En quant al nostre projecte, la integració de Dart amb Flutter és una de les seves característiques més importants. Amb Flutter, es poden crear interfícies d'usuari atractives mitjançant widgets personalitzables.

#### 3.4.4.3. Java (Spring boot)

Java és un llenguatge de programació orientat a objectes àmpliament utilitzat en el desenvolupament d'aplicacions empresarials. Destaca per la seva portabilitat, ja que les aplicacions Java poden ser executades en diverses plataformes amb la màquina virtual de Java (JVM). Això permet als desenvolupadors escriure una sola vegada i executar en diferents sistemes operatius.

Java i Spring Boot ofereixen una sèrie de beneficis per al desenvolupament d'aplicacions empresarials. Java és àmpliament conegut i té una gran comunitat de desenvolupadors que proporciona suport i recursos. A més, gràcies a la seva orientació a objectes, Java permet una organització clara i estructurada del codi.

Spring Boot, d'altra banda, ofereix funcionalitats avançades per al desenvolupament d'aplicacions empresarials, com ara injecció de dependències, gestió de transaccions i seguretat. A més, simplifica la configuració tècnica amb les seves convencions predeterminades, permetent als desenvolupadors centrar-se en el desenvolupament de l'aplicació.

#### 3.4.4.4. Kotlin(Gradle)

Kotlin és un llenguatge de programació modern i concís que s'utilitza cada vegada més en el desenvolupament d'aplicacions empresarials. Una de les seves característiques destacades és la seva interoperabilitat amb Java, el que permet als desenvolupadors utilitzar Kotlin als arxius de configuració de Gradle i a altres parts del projecte mentre segueixen utilitzant Java en altres aspectes. És per això que utilitzem Kotlin, per als arxius de *Gradle*. En un principi vam voler fer servir Kotlin per a aprendre més d'aquest llenguatge, però al final hem vist que el paper que juga als arxius de *Gradle* és molt bàsic.

#### 3.4.4.5. JavaScript (NodeJS)

JavaScript és un llenguatge de programació de scripts d'alt nivell, orientat a objectes i interpretat. Va ser creat originalment per a ser utilitzat en el context del navegador web,

però avui en dia s'ha expandit i es fa servir en diversos entorns, incloent-hi el desenvolupament web i d'aplicacions mòbils.

Una de les àrees on JavaScript té més presència és en el desenvolupament web. Fa ús de scripts emmagatzemats directament a l'HTML o en fitxers separats. Amb l'arribada de tecnologies com HTML5 i CSS3, JavaScript s'ha convertit en una eina imprescindible per a crear aplicacions web complexes.

El llenguatge ha evolucionat amb el temps i ha guanyat molta popularitat, principalment perquè és un estàndard obert i compatible amb la majoria de navegadors moderns. Hi ha una gran quantitat de biblioteques i frameworks disponibles per a facilitar el desenvolupament en JavaScript.

En resum, JavaScript és un llenguatge de programació ampliament utilitzat i versàtil que ofereix funcionalitats per a desenvolupar aplicacions web interactives i altres aplicacions en diferents entorns. Amb l'expansió de les tecnologies web, JavaScript continua sent un llenguatge essencial per als desenvolupadors.

## 3.5. APIs

### 3.5.1. API del nostre producte

La nostra API prové al *frontend* de la informació necessària per a mostrar a l'aplicació nativa, així com la possibilitat d'interacció amb la base de dades. Dividim la nostra API en tres grans grups, un per a cada model de dades.

#### 3.5.1.1. /users

| user-controller |                      |  | ^ |
|-----------------|----------------------|--|---|
| GET             | /users               | Gets the profile of all the users registered in the system | ▼ |
| PUT             | /users               | Edits the profile of a logged user                         | ▼ |
| POST            | /users               | Registers an user into the system                          | ▼ |
| DELETE          | /users               | Deletes a logged user                                      | ▼ |
| POST            | /users/logout        | Logs out an user   | ▼ |
| POST            | /users/authenticate  | Logs an user in  | ▼ |
| GET             | /users/profile       | Gets the profile info of a logged user                     | ▼ |
| GET             | /users/notifications | Sends all notifications to a user                          | ▼ |

Els endpoints que componen aquest grup, com el seu nom indica, tenen a veure amb els usuaris. En ordre, proveïm una breu descripció de les crides:

- GET /users: Demana el perfil de tots els usuaris guardats a la base de dades i els retorna amb paginació o sense, segons s'hagi especificat (el treurem per a la entrega, només és per a facilitar-nos la vida a l'hora de desenvolupar).
- PUT /users: Donada una API token que representa la sessió d'un usuari loguejat i un perfil d'usuari en format JSON, modifica l'usuari a qui pertany aquesta API token i li assigna la informació especificada al JSON.
- POST /users: Registra un usuari a la base de dades.
- DELETE /users: Donada una API token que representa la sessió d'un usuari loguejat, esborra la sessió de l'usuari, les seves assistències, incidències, llistes i finalment el propi usuari.
- POST /users/logout: Donada una API token que representa la sessió d'un usuari loguejat, esborra aquesta sessió de la BD.
- POST /users/authenticate: Donat l'identificador d'un usuari i un device token (obtinguts a partir d'una crida mitjançant *Firebase*), crea una sessió per a aquest usuari i retorna una API token.
- GET /users/profile: Donada una API token que representa la sessió d'un usuari loguejat, obté les dades que conformen el perfil d'aquest usuari.
- GET /users/notifications: Donada una API token que representa la sessió d'un usuari loguejat que a més és administrador, aquesta crida envia una notificació d'esdeveniment proper a l'usuari administrador.

### 3.5.1.2. /lists

| list-controller |               |  | ^ |
|-----------------|---------------|--|---|
| GET             | /lists        | Gets all the lists of a logged user            | ▼ |
| PUT             | /lists        | Edits the name and description of a given list | ▼ |
| POST            | /lists        | Creates an empty list for a logged user        | ▼ |
| DELETE          | /lists        | Deletes a list of a logged user                | ▼ |
| GET             | /lists/events | Gets all the events of a specified list        | ▼ |
| POST            | /lists/events | Adds an event to a list                        | ▼ |
| DELETE          | /lists/events | Removes an event from a list                   | ▼ |

Els endpoints que componen aquest grup tenen a veure amb les llistes d'esdeveniments que els usuaris es fan de manera personalitzada. En ordre, proveïm una breu descripció de les crides:

- GET /lists: Donada una API token que representa la sessió d'un usuari loguejat, obté totes les llistes de l'usuari en qüestió.
- PUT /lists: Donada una API token que representa la sessió d'un usuari loguejat, l'identificador d'una llista d'aquest usuari, un nom i una descripció, assigna el nom i descripció a la llista identificada per l'identificador si aquesta pertany a l'usuari de l'API token.
- POST /lists: Donada una API token que representa la sessió d'un usuari loguejat i un string, crea una llista buida per a aquell usuari amb el nom indicat.
- DELETE /lists: Donada una API token que representa la sessió d'un usuari loguejat i un identificador de llista, esborra la llista si aquesta pertany a l'usuari en qüestió.
- GET /lists/events: Donada una API token que representa la sessió d'un usuari loguejat i un identificador de llista, obté tota la informació dels esdeveniments que formen aquesta llista.
- POST /lists/events: Donada una API token que representa la sessió d'un usuari loguejat, un identificador de llista i un identificador d'esdeveniment, afegeix l'esdeveniment a la llista si aquesta pertany a l'usuari en qüestió.
- PUT /lists/events: Donada una API token que representa la sessió d'un usuari loguejat, un identificador de llista i un identificador d'esdeveniment, esborra l'esdeveniment de la llista si aquesta pertany a l'usuari en qüestió.

### 3.5.1.3. /incidents

| incident-controller |                           |  | ^ |
|---------------------|---------------------------|--|---|
| GET                 | /incidents                | Gets all incidents   | ▼ |
| PUT                 | /incidents                | Edits a concrete incident                                    | ▼ |
| POST                | /incidents                | Saves a concrete incident in the database                    | ▼ |
| DELETE              | /incidents                | Deletes a concrete incident specified by its id              | ▼ |
| GET                 | /incidents/filters/users  | Gets a concrete incident by an specific user mail and status | ▼ |
| GET                 | /incidents/filters/events | Gets a concrete incident by an specific event and status     | ▼ |
| DELETE              | /incidents/event          | Deletes all the incidents with its event id                  | ▼ |

Els endpoints que componen aquest grup, com el seu nom indica, tenen a veure amb les incidències sobre els esdeveniments de la nostra aplicació. En ordre, proveïm una breu descripció de les crides:

- GET /incidents: Demana totes les incidències guardades a la base de dades i les retorna amb paginació o sense, segons s'hagi especificat.
- PUT /incidents: Donat una incidència en format JSON, modifica els atributs especificats de la incidència corresponent identificada per l'id, el qual es troba com a primer paràmetre del JSON.
- POST /incidents: Crea una incidència i la guarda a la base de dades.
- DELETE /incidents: Elimina una incidència guardada a la base de dades a partir de l'identificador d'aquesta.
- GET /incidents/filters/users: Obté el conjunt d'incidències reportades per un usuari concret donat el seu email i l'estat d'aquesta, tenint en compte si ha estat resolta o encara no.
- GET /incidents/filters/events: Obté el conjunt d'incidències d'un esdeveniment en concret donat el seu identificador i l'estat d'aquesta incidència, tenint en compte si ha estat resolta o encara no.
- DELETE /incidents/event: Elimina el conjunt d'incidències guardades a la base de dades relacionades amb l'esdeveniment identificat pel seu id corresponent.

#### 3.5.1.4. /events

Els endpoints que componen aquest grup, com el seu nom indica, tenen a veure amb els esdeveniments. En ordre, proveïm una breu descripció de les crides:

| event-controller |                          | ^   |
|------------------|--------------------------|---|
| GET              | /events                  | Gets all events from the database   |
| PUT              | /events                  | Edits a concrete event  |
| POST             | /events                  | Saves a concrete event to the database  |
| POST             | /events/rate             | Rates an event  |
| DELETE           | /events/rate             | Removes the user rating of an event   |
| GET              | /events/{id}/allInfo     | Gets an event with the specified id from the database                                   |
| GET              | /events/tags             | Gets the possible filter tags from the database   |
| GET              | /events/suggestions      | Gets the recommended events for a specific user   |
| GET              | /events/preu             | Gets events with the specified price from the database                                  |
| GET              | /events/distance         | Gets events with the specified distance, user position and date range from the database |
| GET              | /events/descripcio       | Gets events with the specified description from the database                            |
| GET              | /events/denominacio      | Gets events with the specified denomination from the database                           |
| GET              | /events/date             | Gets events with the specified range of dates from the database                         |
| GET              | /events/comarcaIMunicipi | Gets events with the specified region from the database                                 |
| GET              | /events/categoria        | Gets events with the specified category from the database                               |
| GET              | /events/ambit            | Gets events with the specified ambit from the database                                  |
| GET              | /events/altres           | Gets events with the specified id from the database                                     |
| GET              | /events/allFilters       | Gets events with the specified filters from the database                                |
| DELETE           | /events/{id}             | Deletes a concrete event from the database  |

- GET /events: Demana tots els esdeveniments guardats a la base de dades i els retorna amb paginació o sense, segons s'hagi especificat.
- PUT /events: Donat un esdeveniment en format JSON, modifica els atributs especificats de l'esdeveniment identificador del qual es troba com a primer paràmetre del JSON.
- POST /events: Crea un esdeveniment i el guarda a la base de dades.
- POST /events/rate: Donada una API token que representa la sessió d'un usuari loguejat, crea una valoració per un esdeveniment concret i la guarda a la base de dades.
- DELETE /events/rate: Donada una API token que representa la sessió d'un usuari loguejat, elimina una valoració d'un esdeveniment guardat a la base de dades donat el seu identificador.
- GET /events/{id}/allInfo: Obté tots els camps d'un esdeveniment guardat a la base de dades identificador del qual s'especifica mitjançant l'URL de la crida.



- GET /events/date: Obté tots els camps dels esdeveniments guardats a la base de dades que es portin a terme entre el rang de dates especificat.
- GET /events/tags: Obté tots els àmbits, categories i altres categories que es troben presents a la nostra base de dades.
- GET /events/suggestions: Donada una API token que representa la sessió d'un usuari loguejat, obté tots es esdeveniments recomanats per l'usuari que està loguejat a l'aplicació.
- GET /events/preu: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent al preu proporcionat mitjançant l'URL de la crida.
- GET /events/distance: Obté tots els esdeveniments que es portin a terme durant un rang de dates determinat i estiguin a una distància menor o igual del radi especificat donada una posició en concret, la qual consta d'una latitud i una longitud.
- GET /events/descripcio: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a la descripció proporcionada mitjançant l'URL de la crida.
- GET /events/denominacio: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a la denominació(nom) proporcionada mitjançant l'URL de la crida.
- GET /events/comarcaIMunicipi: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a la comarca i municipi(en format *comarca,municipi*) on es porten a terme dits esdeveniments proporcionada mitjançant l'URL de la crida.
- GET /events/categoria: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a la categoria proporcionada mitjançant l'URL de la crida.
- GET /events/ambit: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a l'àmbit proporcionat mitjançant l'URL de la crida.
- GET /events/altres: Obté tots els camps dels esdeveniments guardats a la base de dades que continguin un string referent a les altres categories proporcionades mitjançant l'URL de la crida.

- GET /events/allFilters: Obté tots els esdeveniments que concorden amb els filtres especificats com a JSON que es troben a la nostra base de dades.
- DELETE /events/{id}: Elimina un esdeveniment guardat a la base de dades identificador del qual s'especifica mitjançant l'URL de la crida.

### 3.5.1.5. /assistances

| assistance-controller |  |
|-----------------------|--|
| GET                   | /assistances Checks if an user will assist to a specific event |
| POST                  | /assistances Saves a concrete assistance                       |
| DELETE                | /assistances Deletes a concrete assistance                     |
| GET                   | /assistances/events Gets all user-specific assistances         |

- GET /assistances: Donada una API token que representa la sessió d'un usuari loguejat i un identificador d'esdeveniment, retorna *true* si l'usuari de l'API token assistirà a l'esdeveniment especificat, *false* altrament.
- POST /assistances: Donada una API token que representa la sessió d'un usuari loguejat i un identificador d'esdeveniment, afegeix una assistència a la BD d'aquest usuari a aquest esdeveniment.
- DELETE /assistances: Donada una API token que representa la sessió d'un usuari loguejat i un identificador d'esdeveniment, esborra de la BD l'assistència d'aquest usuari a aquest esdeveniment.
- GET /assistances/events: Donada una API token que representa la sessió d'un usuari loguejat obté la informació de tots els esdeveniments als que l'usuari en qüestió assistirà.

**NOTA:** Les crides de tipus GET que retornen més d'un element implementen totes paginació opcional, que s'ha d'especificar mitjançant paràmetres opcionals de les pròpies crides.

### 3.5.1.6. Gestió de la seguretat

La nostra api compta amb securització mitjançant *Json Web Tokens*. Cada token és únic per a cada usuari, i s'obté com a resposta al moment de fer login mitjançant la crida de tipus GET a /users/authenticate.

El token(anomenat en tota la aplicació com a API Token, no confondre amb Device Token), es demana com a header a les crides que són personalitzades per a cadascun dels usuaris, especificades més amunt. Aquest token es decodifica al backend i s'obté el seu *body*, el qual només conté l'identificador de l'usuari al que fa referència. Amb aquest identificador, podem efectuar operacions per a obtenció, actualització, inserció o esborrament de dades de manera individualitzada.

De totes maneres, al codi de l'aplicació *backend*, sempre realitzem comprovacions per a veure que l'usuari que està intentant accedir a unes dades és realment el que toca i, en cas contrari, llancem l'excepció *PermissionDeniedException*, que resulta en la crida a la API retornant el codi 403(*Forbidden*) i el missatge que calgui.

### 3.5.2. APIs externes

Tot i que l'API principal i més utilitzada es la que hem implementat nosaltres, la nostra aplicació utilitza varies APIs que ens faciliten la implementació de funcionalitats com el mapa i el registre d'usuaris. En el cas de Flutter utilitzem plugins, que ens proporcionen els mètodes per comunicarnos amb les següents APIs:

#### 3.5.2.1. Google Maps API

Permet mostrar mapes interactius de Google en pantalla completa, personalitzar els marcadors. També permet realitzar diverses interaccions amb els mapes, com fer zoom, moure's i obtenir informació sobre ubicacions específiques.

Per utilitzar el plugin de Google Maps de Flutter, és necessari configurar una API token de Google Maps perquè l'aplicació pugui accedir als serveis de mapes de Google. Aquesta API token és única per a cada aplicació i s'utilitza per a autenticar les sol·licituds i assegurar que l'ús del servei estigui dins dels límits i restriccions establerts per Google.

La API ofereix una sèrie de característiques addicionals, com la detecció 'touch events' en el mapa, la geocodificació (convertir direccions en coordenades geogràfiques) i la inversió de geocodificació (convertir coordenades geogràfiques en direccions). També és possible superposar capes personalitzades en els mapes, com a polígons, polilíneas i cercles, per a representar àrees o elements específics.

De tots els serveis mencionats, el que més utilitzarem seran la detecció de 'touch events' per gestionar les crides al back per obtenir només els esdeveniments que s'haurien de veure per pantalla, i per interaccionar amb els marcadors i veure la info

d'un esdeveniment. També utilitzarem la personalització de marcadors, del mapa i la funcionalitat de mostrar la ubicació a temps real del dispositiu.

### 3.5.2.2. Firebase Auth API

Firebase Auth és un servei de Firebase que ofereix una varietat de mètodes d'autenticació per a usuaris, com correu electrònic i contrasenya, Google Sign-In, Facebook Login, Twitter Login, entre d'altres. Permet agregar fàcilment mètodes d'autenticació segurs i confiables, sense haver d'implementar tot el codi des de zero.

Aquestes són les principals funcionalitats que utilitzem per la nostra aplicació:

**Autenticació d'usuaris:** Proporciona mètodes simples per registrar nous usuaris, iniciar sessió amb credencials existents i tancar sessió. Permetem que els usuaris creïn comptes utilitzant la seva adreça de correu electrònic i contrasenya, o iniciïn sessió amb un proveïdor externs com Google.

**Gestió d'usuaris:** Podem accedir a informació i propietats d'usuari, com el seu ID únic, nom i adreça de correu electrònic. A més, podem actualitzar la informació de l'usuari, com canviar l'adreça de correu electrònic o la contrasenya.

**Verificació de correu electrònic:** Firebase Auth permet enviar correus electrònics de verificació als usuaris per verificar les seves adreces de correu electrònic. Podem verificar si un usuari ha verificat el seu correu electrònic, incrementant la seguretat en el moment de la creació d'un compte.

**Restabliment de contrasenya:** Si un usuari oblida la seva contrasenya, Firebase Auth ens proporciona mètodes per restablir-la enviant un correu electrònic de restabliment de contrasenya. Això ens facilita la recuperació l'accés als comptes de forma segura.







### 3.5.2.3. Firebase Storage API

Firebase Storage és un servei en el núvol proporcionat per Firebase que s'utilitza per a emmagatzemar i administrar arxius multimèdia, com a imatges, vídeos i documents, de manera segura i escalable.

Firebase Storage proporciona un emmagatzematge d'objectes basat en el núvol, cosa que significa que els arxius es desen en contenidors anomenats "buckets" i s'accedeix a ells a través d'una API. Els desenvolupadors poden utilitzar aquesta API per a carregar i descarregar arxius des de les seves aplicacions mòbils o web de manera senzilla.

Una dels avantatges de Firebase Storage és que maneja automàticament l'escalabilitat i la redundància de les dades. Els arxius s'emmagatzemen de manera segura en múltiples ubicacions per a garantir la seva disponibilitat i durabilitat.

Nosaltres utilitzem aquest servei per a emmagatzemar i administrar les imatges de perfil dels usuaris de la nostra aplicació. Ens permet guardar-les de manera segura i relativament senzilla. Generem un directori únic per cada usuari i allà guardem la seva imatge de perfil. Cada vegada que l'usuari l'actualitza esborrem la imatge antiga i l'actualitzem per la nova, de manera que optimitzem l'espai de memòria necessari. En la següent imatge es poden veure diferents directoris on emmagatzemem imatges d'usuari:

| gs://culturefinder-3a452.appspot.com > profile_images |   |        |         |                           | <a href="#">Subir archivo</a> |  |
|---|---|--------|---------|---------------------------|-------------------------------|--|
| <input type="checkbox"/>                              | Nombre  | Tamaño | Tipo    | Modificación más reciente |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJFYk5rMFIESGFxWWVoTGxjTTJOWTRvbEVzdFMyIn0.NrcAr9psIXW4drIvbfhNDsaeWu072BwZgfqX7MATQc/    | —      | Carpeta | —                         |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJHZDBickZPMXRmVuc3ZlUldhdG9mFoZ1VJbEozIn0.WdDNF6XqnbERDoJvBJSyelX7aLz-OVW_13ynS1-zzqE/   | —      | Carpeta | —                         |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJNa0FIVGh5S3NnVXg1S0hsdXJKZS52OWdnYXYyIn0.k5TNJfuzB0B4BqVzR_p-PqNUbH9w0mMkGr5xgz2MJrY/   | —      | Carpeta | —                         |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJlPWHRTR3Nhb0hqU3ZmaWw3QTU4cEdDMDxSWQyIn0.p9a2MaYbQXOC_Zt92zqQoHX4trGqUqx3bLpD21IY1g/   | —      | Carpeta | —                         |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJUNnQzNG02WRRBWXB6eWhXVm9tQWswY1QzQlcyIn0.uGfMkzGYDmFmf9v7ffBXfva77NBcLi6TeO1Y9Uw/     | —      | Carpeta | —                         |                               |  |
| <input type="checkbox"/>                              |  eyJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQIOiJlNIA2UXgwdWZST1BlaWVpVmVwcG1XNXpta3oyIn0.vcZwLMb7adTaux4iXsd7xuL1m9vNXAyWnexWCZV8zLc/ | —      | Carpeta | —                         |                               |  |

#### 3.5.2.4. Firebase Cloud Messaging API

Firebase Cloud Messaging (FCM) és un servei de missatgeria en el núvol desenvolupat per Google. Permet als desenvolupadors enviar missatges i notificacions de manera de fiable a dispositius mòbils.

FCM proporciona una infraestructura escalable i segura per a enviar missatges a través d'una xarxa global de servidors. Utilitza el protocol de notificació push per a lliurar missatges instantàniament als dispositius registrats, la qual cosa permet als usuaris rebre actualitzacions i notificacions en temps real. Els desenvolupadors poden utilitzar la API de FCM per a enviar missatges a individus o grups de dispositius. Aquests missatges poden contenir informació personalitzada, com a text, imatges o enllaços, i es poden dirigir a usuaris específics o segments d'usuaris en funció de diversos criteris.

A més de les notificacions push, FCM també admet altres tipus de missatges, com a missatges de dades, que permeten enviar informació addicional a l'aplicació en segon pla sense mostrar una notificació visible per a l'usuari.

En el nostre cas utilitzem aquest servei per enviar notificacions als dispositius d'usuaris loguejats els quals han indicat que assistiran a algun esdeveniment. Des del backend el dia abans de que comenci un esdeveniment, enviem una notificació a tots els usuaris que han indicat que assistiran a aquest com a recordatori. En el frontend ho hem implementat de tal manera que també arribi la notificació quan l'aplicació s'està executant en segon pla.

#### 3.5.2.5. Geolocator API

Ens permet obtenir la ubicació del dispositiu i fer un seguiment en temps real. Ens facilita la obtenció de les coordenades de longitud i latitud, configurar opcions de precisió i freqüència d'actualització, i verificar i demanar permisos necessaris. L'utilitzem principalment per centrar el mapa en la ubicació a temps real del nostre dispositiu i per obtenir la distància en km entre diferents esdeveniments.

### 3.5.3. Consum del servidor

Hem consensuat amb l'equip que desenvolupa l'aplicació *PlugFinder* per a fer ús d'una de les crides a la seva API. La crida ens proporcionarà una llista d'endolls on carregar un cotxe elèctric, informació que li serà útil al *frontend* per a mostrar marcadors amb aquests endolls als mapes on es mostrin esdeveniments.

Respecte a com ens vam comunicar en aquest tercer sprint per saber en més detalls la informació de la seva crida, vam tenir una petita reunió amb els membres de *PlugFinder* per tal de que ens expliquessin una mica més en profunditat com funcionava tot. A més a més, per facilitar-nos la vida ens van enviar un correu amb el qual teníem la informació disponible per si sorgia qualsevol dubte relacionat.

### 3.5.4. Subministrament del servidor

Hem arribat a un acord amb el grup que desenvolupa l'aplicació *Concessiona't* per a oferir-los una crida a la nostra API. La crida en qüestió és una que s'ha descrit a l'apartat 3.5.1.4, concretament, */events/distance*. Aquesta crida, donats un *radi*, *latitud*, *longitud*, i

dates d'inici i fi(i paràmetres opcionals de paginació per al resultat), obté una llista d'esdeveniments presencials que tenen lloc a un radi de *radi* kilòmetres a la ubicació (*latitud*, *longitud*) entre les dates *dataIni* i *dataFi*.

Aquesta informació els hi serà útil a l'equip de desenvolupament de *Concessiona't* per a poder mostrar al mapa de la seva aplicació marcadors amb els esdeveniments que retorna la crida.

Per a facilitar a l'altre grup la documentació necessària perquè no tinguessin cap problema a l'hora d'integrar el nostre servei a la seva aplicació, vam decidir oferir-los una part de la documentació que tenim del nostre **Swagger**, però concretament, de la crida de l'API (GET /events/distance) que els hi oferim.

D'aquesta manera, el responsable de l'equip, en aquest cas, *Daniel Morón*, els hi va informar via email de com poden accedir-hi a aquesta informació adjuntant el fitxer *openapi* corresponent per poder visualitzar-lo a l'editor de Swagger online.



**Daniel Moron Rocés** <daniel.moron.roces@estudiantat.upc.edu>

dt., 16 de maig 18:26 (fa 12 dies)

per a joel.rivera, oscar.delgado.gomez, ivan.risueno, miguel.moreno.alcaraz, marc.rodriguez.martin, marc.duch ▾

Buenas Joel,

La llamada sería:

GET

<http://nattech.fib.upc.edu:40380/events/distance?dataIni=X&dataFi=X&radi=X&latitud=X&longitud=X&page=X&size=X&enablePagination=X>

: Aquí podéis configurar si activáis paginación o no, y de qué página queréis recoger la información.

Un ejemplo real:

```
Curl
curl -X GET \
  'http://nattech.fib.upc.edu:40380/events/distance?dataIni=2023-07-20&dataFi=2023-10-20&radi=10&latitud=41&longitud=1&page=0&size=10&enablePagination=true' \
  -H 'accept: application/json'

Request URL:
http://nattech.fib.upc.edu:40380/events/distance?dataIni=2023-07-20&dataFi=2023-10-20&radi=10&latitud=41&longitud=1&page=0&size=10&enablePagination=true
```

URL: <http://nattech.fib.upc.edu:40380/events/distance?dataIni=2023-07-20&dataFi=2023-10-20&radi=10&latitud=41&longitud=1&page=0&size=10&enablePagination=true>

En caso de no querer paginación, poniendo enablePagination = false serviría.

Cualquier duda o problema nos comentáis!

Saludos,



**Daniel Moron Rocés** <daniel.moron.roces@estudiantat.upc.edu>

dt., 23 de maig 9:46 (fa 5 dies)



per a joel.rivera ▾

Buenas Joel,

Os paso un txt para que podáis ver la llamada de GET /events/distance en el Swagger. Así podéis ver de manera más gráfica como funciona y los datos que necesitaréis pasarle a la llamada, pero para utilizarla usad el enlace del otro correo.

Tendréis que añadir la información del txt en: <https://editor.swagger.io/>

Saludos,

Servers

<http://nattech.fib.upc.edu:40380> - Generated server url

event-controller

GET

</events/distance> Gets events with the specified distance, user position and date range from the database

Com a conclusió, podem comentar que al principi vam tenir certs problemes de comunicació amb l'altre equip, però a mesura que van avançar els sprints va millorar considerablement, vam aclarir tots els dubtes al voltant de la funcionalitat i nosaltres, com a membres de *CultureFinder*, vam intentar ajudar el màxim possible aportant tota la documentació que teníem, sense perjudicar la seguretat i privacitat del nostre producte.

### 3.5.5. Consum de Dades Obertes

Per a proveir al *frontend* d'informació sobre els esdeveniments que tenen lloc a Catalunya, necessitem consumir la API de l'Agenda Cultural. Concretament, per a obtenir els esdeveniments que formen el dataset (en podem obtenir 1000 com a màxim per crida) hem de fer una petició de tipus GET a l'endpoint "<https://analisi.transparenciacatalunya.cat/resource/rhpy-yr4f>".

Aquesta crida, si la fem sense paràmetres, ens retornarà com a màxim els 1000 primers esdeveniments que es troben a la base de dades de l'aplicació de l'Agenda Cultural. Nosaltres, per conveniència i per fer ús de dades representatives, hem decidit afegir a la crida el paràmetre de tipus Query `<data_inici>`. En concret, la crida resultant retorna els esdeveniments pertinents amb una data d'inici major a la del dia quan es fa la crida, així ens estalviem obtenir esdeveniments que són passats.

Malgrat això, "no tot són flors i violes", i el dataset que ofereix la API de l'Agenda Cultural està ple d'informació inconsistent que, nosaltres, des de l'aplicació *backend*, hem de tractar per netejar o directament eliminar:

- Codi: el codi dels esdeveniments a priori pot semblar que funciona com a clau primària, però no és el cas: és una combinació de l'any quan es fa l'esdeveniment i altres dígits arbitraris, i no és únic per a cada esdeveniment.



- Dates: les dates d'inici i de fi dels esdeveniments venen en format "dd/mm/AAAA" + "T" + "hh:mm:ss", i n'hi han algunes que valen 99/99/9999.
- Descripció: malgrat haver un camp de descripció en format HTML, el camp de descripció que hauria de ser text pla no ho és en alguns casos.
- Subtítol: mateix cas que amb les descripcions.

Aquests camps que hem esmentat són inconsistents, però també hi han d'altres, com per exemple *àmbits*, *categories*, *altres categories* o *comarca i municipi* que només ens hem d'encarregar de donar-li un format més llegible.

Per últim, remarcar que a banda del sodi, denominació i descripció dels esdeveniments, tota la resta de camps poden ser buits.

Davant d'aquestes adversitats ens veiem obligats al *backend* a netejar tota la informació que rebem a través de la API de l'Agenda Cultural per a guardar a la nostra BD dades llegibles i útils per al *frontend* o per a l'aplicació *Concessiona't*.

## 3.6. Eines de desenvolupament i entorn de treball

### 3.6.1. Deployment

#### 3.6.1.1. Backend

Donat que l'aplicació *backend* ha d'estar en constant execució per a poguer subministrar informació al *frontend*, hem hagut de fer un deploy a un servidor. Com ja hem esmentat prèviament, en un principi vam decidir fer servir Amazon Web Services, però donada la seva complexitat vam decidir canviar-nos al servei que ofereix Virtech.

En una primera instància, el deploy el fèiem desde la màquina virtual clonant el repositori del *backend* i executant el codi de la branca *main*. Aquesta metodologia, però, és molt poc sofisticada i sobre tot poc segura, ja que qualsevol que pugui entrar a la màquina virtual podria veure el codi font de l'aplicació.

És per això que vam decidir canviar el mètode i començar a desplegar l'aplicació només amb un arxiu. Gràcies al IDE escollit, IntelliJ, podem crear una configuració nova que s'encarregui de compilar tot el codi i generar un arxiu jar. Aquesta compilació, donat que

fem servir Gradle, resulta molt simple ja que es pot fer executant la comanda “./bootJar” al directori arrel de l'aplicació (el que conté els arxius de configuració de Gradle). Aquesta comanda juntament amb la configuració al nostre IDE ens permet generar un arxiu executable de la nostra aplicació que rep el nom “CultureFinderBackend-*nVersion.jar*”, on *nVersion* correspon a un paràmetre de tipus string especificat a l'arxiu *build.gradle.kts*. Aquest arxiu es genera dins del directori “/build/libs/” de manera predeterminada.

Un cop obtingut l'executable, només cal connectar-se a la VPN de la UPC per tal de poguer efectuar la comanda “*scp -P 22038 /path/to/executable/jar alumne@nattech.fib.upc.edu:/destination/path*” cap al servidor de Virtech. Et demanarà la contrassenya, i un cop executat l'*scp*, a la consola de la màquina virtual executar l'arxiu jar fent només “*java -jar /path/to/executable/jar*”.