UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Culture finder

SPRINT 2: DOCUMENTACIÓ

Github - Taiga - Drive - Project Record Track

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

1. Introducció

1.1. Resum executiu del segon sprint

L'objectiu d'aquest sprint ha estat finalitzar les tasques restants de l'últim sprint i desenvolupar el següent set de tasques que hem considerat importants per la nostra aplicació.

Per la part de Backend, una de les tasques més importants era oferir el servei a l'altre equip. Aquest servei consisteix en retornar tots els esdeveniments dins d'un radi i un rang de dates. Més enllà del servei, es va acabar una part important que havia quedat pendent del primer sprint que es l'actualització periòdica de dades. També s'ha treballat a les crides d'API per les noves funcionalitats i així possibilitar el treball al frontend. Ja que algunes tasques son dependents d'aquestes crides. També es van fer múltiples refactors per millorar l'estructura i l'arquitectura de l'aplicació. Aquests refactors van abarcar la majoria de temps del sprint, ja que tracten conceptes amb els que no érem familiars. També van anar sorgint diferents millores a mesura que es feien els refactors que van anar retardant la finalització. Algunes millores afegides han estat els DTOs i els mappers que donen més independència a les entitats a cadascuna de les tres capes. A més s'ha implementat un sistema de json web tokens per securitzar la majoria de crides d'API.

Per la part de Frontend, s'han arreglat alguns dels problemes que quedaven per resoldre respecte al mapa. Una de les parts més importants d'aquest sprint era la de registre i iniciar sessió. Aquestes funcionalitats s'han completant amb firebase, el qual gestiona els usuaris i retorna un token que utilitzem per fer les crides d'API. Respecte les vistes, s'ha creat una versió inicial de la homepage, s'ha millorat la versió detallada dels esdeveniments, s'ha afegit la funcionalitat de les incidències i també s'han afegit diferentes opcions de filtratge. Les 3 funcionalitats més prioritàries, registre/iniciar sessió, filtratge i afegir incidències han estat completades i han ocupat la major part del temps d'aquest sprint. Més enllà d'aquestes funcionalitats, també s'ha millorat el codi respecte l'anterior sprint. Canviant codi de lloc i eliminant algunes funcions que no eren necessàries per tal de millorar l'arquitectura.

També s'ha treballat a la part de la pàgina web d'incidències per administradors. Aquesta part es nova a aquest sprint ja que no era prioritaria a la nostra aplicació. S'ha desenvolupat mitjançant l'arquitectura model-view-controller i el llenguatge Javascript. Aquesta web consta d'una pàgina principal on l'usuari que

ja ha estat registrat a la part de backend com a administrador pot iniciar sessió. Una vegada loguejat, l'usuari pot veure les incidències dels usuaris i les pot filtrar per email, id, resoltes i pendents. L'administrador pot seleccionar una incidència i canviar el seu estat, a més d'afegir un comentari per avisar a l'usuari que ha posat la incidència. Durant aquest sprint, la web no ha quedat finalitzada. La no familiaritat amb les diferents tecnologies (Javascript, model-view-controller) han fet que el desenvolupament fos més lent del que s'esperava.

1.2. Sprint master report

Com a sprint master d'aquest sprint la meva feina ha estat de facilitar el treball als meus companys. M'he encarregat d'organitzar les reunions setmanals dels diumenges per saber l'estat de les tasques de cada membre de l'equip i assegurar la comunicació entre el backend i el frontend. Si algun membre de l'equip no podia assistir a la reunió, s'ha buscat un espai de temps disponible per posar-nos al dia i comunicar el progrés a la resta de l'equip mitjançant els canals de text de Discord o Whatsapp.

També ha estat important mantenir a tots els membres de l'equip motivats perquè ningú perdés el fil de l'assignatura. Intentant fer-los veure que cada vegada queda menys del projecte i que s'ho prenguessin una tasca a la vegada per no sentir-se aclaparats.

També he estat atent a la gestió del projecte al Taiga. He anat recordant a la resta de companys del grup de mantenir les tasques actualitzades, depenent si estaven en procés, no documentades o acabades. També assignant tasques als diferents membres de l'equip.

En general, he intentat fer que l'equip treballés de manera comode i eficient durant aquest sprint. Implicant-me sobretot en la comunicació i l'organització del projecte.

1.3. Que hem fet cada membre de l'equip

Equip del backend

- **Daniel**: Respecte al meu treball pel *backend* d'aquest segon sprint, s'ha basat principalment en la implementació del servei que oferim durant aquesta fase, juntament amb el refactor corresponent relacionat amb la crida relacionada amb els filtres de cerca. Tot i això, al principi del sprint em vaig centrar en acabar les tasques prèvies que s'havien quedat del primer sprint, la qual era la sincronització periòdica de les dades. Cal destacar que a mesura que anàvem implementant nou codi, anaven sorgint noves millores, per aquest mateix motiu ens va retardar una mica el refactor inicial que vam comentar.

  Una vegada vaig acabar això amb l'ajuda d'Ivan fent diferents dies *Pair Programming*, em vaig focalitzar més, en el refactor que vam comentar Ivan i jo amb el Jordi sobre la separació entre capes i les millores possibles que podíem afegir a l'arquitectura del nostre projecte. Al principi vam tenir diferents problemes de configuració i dubtes sobre alguns conceptes, que vam anar esmentant i solucionant conjuntament a mesura que avançàvem en les activitats del sprint. A la part final, em vaig centrar fer diferents modificacions i millores pel que fa a Esdeveniments i Incidències per augmentar i precisar de millor manera el funcionament del filtratge, i a partir d'aquí, vaig implementar el servei que oferim, el qual consisteix en donat un radi, una posició i un rang de data inicial i final, retorna tots els esdeveniments que estan localitzats dins d'aquell radi.

- **Iván(*backend leader*):** Per al *backend* aquest sprint ha sigut sobretot *refactor*. En general, tant en Dani com jo, ens hem dedicat a afegir noves funcionalitats al codi que complementen les que constitueixen els requisits funcionals. Fent *pair programming* hem estat treballant bastant en la crida periòdica a la API de l'Agenda Cultural per a actualitzar la nostra base de dades afegint esdeveniments nous.

  Pel que fa a tasques individuals, m'he encarregat d'afegir l'esquelet per a les excepcions, les excepcions en si mateixes i les comprovacions que les

llencen. A més, m'he informat sobre com i en què consisteix securitzar una API i he dissenyat un sistema utilitzant *json web tokens* per a individualitzar i securitzar la major part de les crides a la API que tenen a veure amb un usuari en concret. També, a mitjes amb en Dani, hem afegit DTOs i mappers a l'aplicació per a donar més independència a les entitats de cadascuna de les tres capes.

En quant a les funcionalitats principals, he afegit les crides i la lògica neccessària per a canviar informació d'usuaris(he afegit a més les categories d'interés i els administradors), les llistes de favorits i les assistències a esdeveniments.

Tot el codi està documentat pertinentment (mètodes, crides a la API i codis d'error).

- **Marc Rodriguez:** Durant aquest sprint la meva feina principalment ha estat enfocada al desenvolupament de la pàgina web per administrador per gestionar incidències. He estat desenvolupant una web amb llenguatge javascript amb arquitectura model-view-controller. Aquesta web conté una pagina inicial on pot iniciar sessió d'un usuari ja registrat a l'aplicació i a més, amb permisos d'administrador. Una vegada loguejat, l'usuari pot veure les incidències filtrant per diferents filtres i obrir un formulari on resoldre la incidència. Apart d'aquesta feina, també he desenvolupat la feina de sprint master. Intentant facilitar la feina i la comunicació de l'equip per treballar de manera eficient.

Equip del frontend

- **Oscar:** He començat aquest sprint modificant els credencials de l'API de Google Maps per tal de que els meus companys poguessin visualitzar el mapa correctament. També vaig modificar el mapa perquè comencès amb un zoom centrat a la ubicació actual del dispositiu, utilitzant l'API de Geolocator.

  Vaig crear les diferents vistes de crear un compte i login de l'aplicació. Per facilitar l'autentificació d'usuaris tant amb correu i contrasenya com amb Google, vaig utilitzar Firebase. També vaig implementar la verificació de correu electrònic i restabliment de contrassenya amb l'API de Firebase. Per realitzar el registre, autentificació i logout d'usuaris vaig implementar

les crides al back a través de la nostra API. Per mantenir la persistencia dels usuaris loguejats i facilitar als meus companys l'accés a l'API token per poder fer crides a la nostra API, he utilitzat Shared Preferences, implementant tots els metodes per gestionar l'API token i la informació de l'usuari loguejat.

- **Marc Duch(*frontend leader*):** Aquest segon sprint el vaig començar implementant una primera versió de la "homepage", una vista que permetés accedir a les pàgines principals que vam implementar al primer sprint. Vaig aprofitar la setmana santa per reescriure la vista en detall d'un esdeveniment per que s'apropés més al disseny mostrat al mockup.
  Amb això acabat, vaig començar amb el tema dels filtres de events, per els quals he creat una base que hauria de poder ser reutilitzada més endavant si volem fer servir aquests filtres en altres vistes o situacions.
  Els filtres que he implementat son, filtrar per nom, categories, distància i per dates d'inici i de fi.
  També he fet petits ajustos a parts del codi que vam fer al primer sprint, com treure mètodes del EventRepositori que no feien falta o separar la lògica de la vista del Calendari en dos classes View i Notifier (controlador).

- **Miguel:** En aquest segon sprint, he començat fent la part de front de la user story de postejar una incidència. Per fer aquesta tasca, vaig afegir un botó a la vista detallada d'event, vermell i amb un triangle d'advertència; quan es prem aquest botó, s'obre un pop-up on indica que s'introdueixi la incidència en un camp de text. Quan es prem acceptar, tanca el pop-up i s'agafa el contingut escrit al camp. Fa una crida al repositori d'incidències, que he fet nou només per aquesta tasca, i que s'utilitzarà per tasques al proper sprint. De moment, però, el repositori d'incidències té una única crida, la de postejar una incidència. Es passa a aquesta funció l'id de l'usuari que fa la incidència i de l'event, la descripció que és el que s'escriu al camp de text, i una string buida i un bool a false ja que aquestes variables són pels administradors. La funció crida a la API del backend, i posteja la incidència amb les variables indicats.
  També he implementat la part de front de les llistes de favorits. Primer vaig fer la vista on surten les diferents llistes de cada usuari. En aquesta vista, es crida al repositori de llista, que vaig crear nou amb les funcions necessàries, i es demana a la API totes les llistes de l'usuari que estigui registrat en aquell moment, enviant el seu API-Token. Des d'aquesta vista també es poden afegir noves llistes a l'usuari; vaig fer un botó que obre un

pop-up amb un camp de text on s'introdueix el nom de la nova llista. S'envia el nom de la llista al repositori, que crida a la API per fer un post d'una nova llista buida. En la vista de llistes, permet clicar a cadascuna de les llistes, i redirigeix a la vista de llista. Vaig crear també aquesta vista, on, al entrar-hi, demana al repositori (que demana a la API) els events assignats a aquesta llista.

1.4. Avaluació dels companys

Avaluació de l'equip. Cada membre de l'equip avalua als seus companys en cada sprint segons 3 valors possibles:

- El membre de l'equip no va fer el que s'esperava (-1).
- El membre de l'equip va fer el que s'esperava (+1).
- El membre de l'equip va fer més de l'esperat i va fer un treball excepcional (+2). Cada membre de l'equip només pot donar aquest valor a un altre membre de l'equip com a màxim.

Aquesta és l'avaluació de l'equip CultureFinder del segon sprint:

	Daniel	Iván	Marc R.	Marc D.	Miguel	Òscar
Daniel	–	+2	+1	+1	+1	+1
Iván	+2	–	+1	+1	+1	+1
Marc R.	+2	+1	–	+1	+1	+1
Marc D.	+2	+1	+1	–	+1	+2
Miguel	+1	+1	+1	+2	–	+1
Òscar	+1	+1	+1	+2	+1	–

2. Cerimònia agile

2.1. Report del sprint planning, revisió & meetings de retrospectiva

2.1.1. Sprint Planning

L'objectiu principal d'aquest segon sprint ha sigut acabar d'implementar el "walking skeleton" que inclou les funcionalitats crítiques de la nostra aplicació, i implementar les funcionalitats més importants. Aquestes funcionalitats són:

- Visualitzar un esdeveniment de forma detallada
- Visualitzar tots els esdeveniments en format llista, mapa i calendari
- Les funcionalitats bàsiques de crear usuari i iniciar/tancar sessió
- Crear incidències (sense resoldre-les)

A partir de la feina feta al primer sprint, ens hem centrat en acabar aquestes funcionalitats i a afegir aspectes importants com filtratge d'esdeveniments, modificació de perfil, llistes i resolució d'incidències.

Per a aquest segon sprint vam estimar 254,5 punts d'història, superant els 190 punts que hauríem assignat si haguéssim distribuït la feina de manera equitativa entre els 3 sprints. Al sprint anterior, vam aconseguir 136,5 punts i, tot i que no els vam poder completar tots, vam decidir afegir més punts a aquest sprint perquè el primer sprint va implicar molta feina que no estava inclosa en les històries d'usuari, com ara la formació, la inicialització del projecte, les bases de dades, el servidor i la familiarització amb els llenguatges. Amb aquesta idea i aquesta xifra, tot i que és optimista, ens ha permès acabar moltes de les funcionalitats plantejades.

Hem cumplit amb quasi tot el planning inicial fet per a l'sprint 2, tot i que pel camí ens han sortit noves tasques que no contemplàvem, a més d'un munt de refactor per al *backend*. Amb tot, les tasques que no hem pogut tancar són de petita granularitat, concretes, i es poden acabar ràpidament en el tercer sprint.

2.1.2. Sprint Retrospective

Per tal de revisar de manera precisa i eficaç com ha anat el procés, l'equip es va reunir el 12 de Maig per fer una reunió retrospectiva sobre com havia anat aquest segon sprint.

Per una banda, respecte al primer sprint, hem millorat considerablement la **comunicació *frontend* <-> *backend***, la qual cosa ha tingut un impacte molt positiu en el rendiment dels dos equips i hem tingut un **desenvolupament més àgil**, ja que hem parlat de manera dinàmica sobre els problemes i possibles solucions que anaven sorgint. A més a més, per aquest mateix motiu cada equip del projecte ha tingut un millor coneixement de l'estat de la feina de cada membre de l'equip.

Per altra banda, si poguessim tornar a començar en aquest segon sprint, hem detectat el **problema de priorització** que hem tingut respecte a la relació de les tasques amb el *backend* i *frontend*, ja que en algun moment del sprint l'equip del frontend no podia avançar massa per dependències amb l'equip del backend. Realment les tasques sí que estaven prioritzades, però a l'hora de començar a desenvolupar el codi en alguns casos no vam tenir en compte la prioritat corresponent. Respecte a aquest punt, concretament parlant del backend, hem parlat per sobre sobre el refactor que hem tingut, el qual possiblement ha retardat un conjunt de tasques del frontend que es podrien haver tancat una mica abans i potser dinamitzar de millor manera el procés de treball.

Un altre punt que vam comentar en la retrospectiva va ser el tema de que vam fer **merge a *dev* des de les branques creades** en alguns punts en els quals no estava la historia d'usuari tancada, però com sempre eren casos puntuals en els quals la raó principal de fer això era dependències o *hotfix* de conflictes que sorgien, hem acordat que en aquest context no hi ha cap problema.

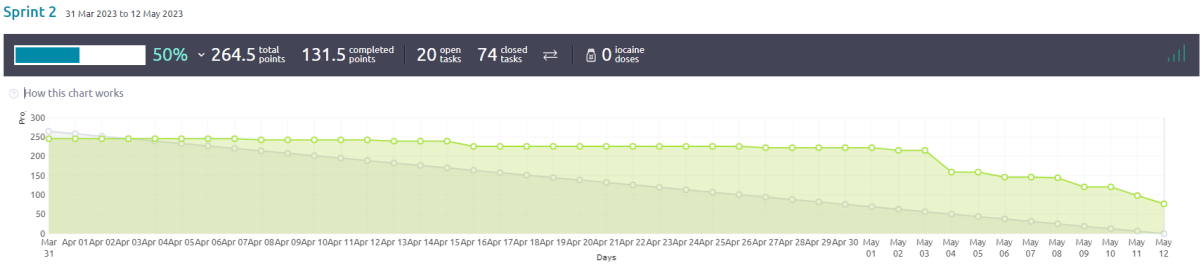
Finalment, com a grup estem molt contents amb la feina feta tot aquest sprint, ja que des de bon començament vam introduir una gran quantitat de punts per ficar-nos una pressió extra afegida, i com a conclusió podem afirmar que ens ha anat bé, tot i que no haguem aconseguit acabar totes les tasques amb les quals havíem començat. Hem comentat que de cara al següent sprint intentarem solucionar tots els problemes esmentats anteriorment per tal d'assolir amb èxit els nostres objectius del projecte.

2.2. Estadístiques del projecte

Els diferents gràfics *burndown* que mostrem a continuació informen de l'avanç del segon sprint del nostre projecte, tots han estat extrets del Taiga. Principalment ve determinat per l'evolució de les tasques al llarg de tot el sprint.

2.2.1. Sprint Burndown

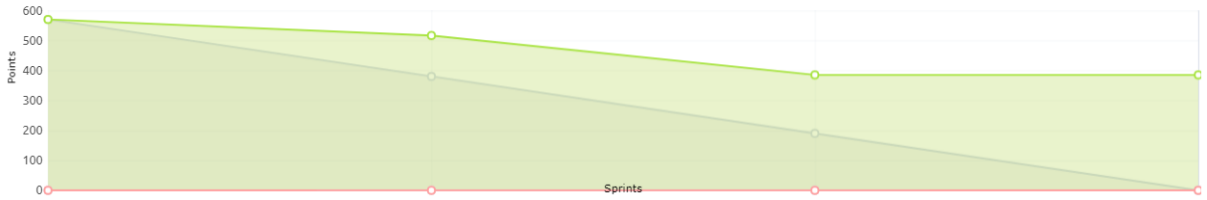
Aquest gràfic mostra el progrés aproximat o projectat del sprint basat en les tasques tancades estimades de les històries d'usuari. Les històries d'usuari estimades acabades reals se segueixen més amunt. Degut al problema que vam tenir per definir quan es tancava una tasca, es pot veure que a la part final del sprint es quan es comencen a tancar les tasques previstes. Ja èrem conscients des d'un primer moment que havíem afegit una gran quantitat de feina, però tot i així crèiem que ha anat millor del que pensàvem.



Com podem veure, el resultat no és ideal. Sabíem que haviem estat ambiciosos a la hora d'escollir les histories a incloure al sprint, pero amb l'assignació de story points a la que havíem arribat, era necessari fer-ne tantes, de fet més. També és cert que una gran part del temps l'hem dedicat a refactor per part del backend i implementació i correcció d'errors en vistes al frontend, i això ens ha suposat un temps extra a l'hora de desenvolupar el programari.

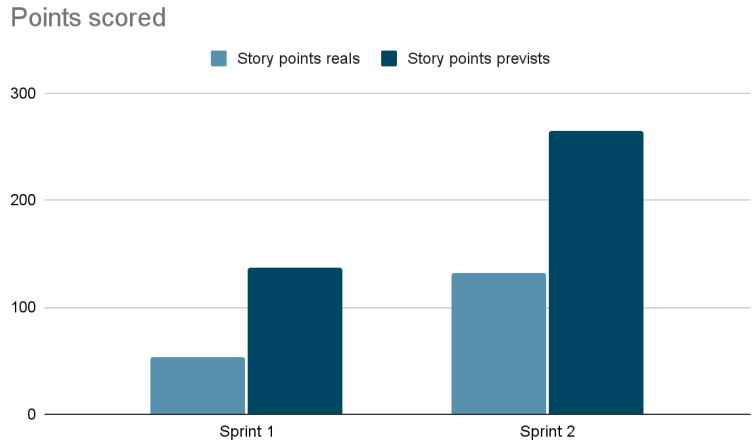
L'altre detall aparent es la baixada dramàtica dels últims dies. En parlem més en detall a la retrospectiva, pero la principal raó es deu a que el refactor del backend va suposar una aturada en quant a la implementació tenint en compte algunes dependències.

2.2.2. Release Burndown



En relació al darrer sprint que ens queda caldria destacar que hi ha moltes històries d'usuaris a punt de tancar-se per detalls, la qual cosa ens ajudarà per fer l'últim sprint de manera més eficient.

2.2.3. Velocity chart



Finalment tenim el velocity chart, que com podem veure, seguim amb la tendència de sobre estimar les nostre capacitats. Però, s'ha de dir que en aquest sprint hem aconseguit fer gairebé els mateixos punts que els que ens vam proposar al primer Sprint.

3. Changelog

3.1. Canvis principals a la metodologia amb justificació

3.1.1. General

3.1.1.1. Persistencia i autentificació del usuaris

Hem decidit unificar la informació rellevant dels usuaris al nostre backend i l'autentificació dels usuaris en el backend i frontend.
Per autentificar els usuaris d'una forma segura, simple i robusta hem decidit utilitzar l'API de Firebase Auth al frontend. Per autentificar els usuaris al backend utilitzem API tokens que genera el backend donat el user ID de l'usuari registrat a Firebase.

El procés de registre d'un usuari és el següent:
Primer es registra un usuari al frontend, s'autentifica a Firebase, es fa una crida al backend per crear l'usuari al servidor, i per últim s'autentifica l'usuari al backend fent una crida a l'API passant el ID de l'usuari i rebent un API token únic.

3.1.2. Frontend

3.1.2.1. Control de versions

Hem decidit desviar-nos una mica de la metodologia del *Git flow* que vam decidir originalment i fer servir una barreja entre el *Trunk based development* i el *Git flow*. Aquesta mescla manté les dues branques principals del git flow (*dev* i *main*) pero ens ofereix la flexibilitat de portar commits intermitjos a la branca de desenvolupament com descriu el *Trunk based*.

La idea es dividir cada historia d'usuari o funcionalitat en la seva propia branca i fer *merge* amb la branca de *dev* quan s'acaba la història o una funcionalitat es requereix en una altre història, com podria ser el model de l'usuari loguejat.

Tot i que vam decidir aquest canvi al principi, aquest merges a *dev* no han estat tan freqüents com creiem, en part perquè voliem mantenir la branca *dev*

funcionant sense haver de dedicar-hi més temps solucionant problemes de codi inacabat. En part també perquè ja ens havíem acostumat al procés de *Git flow* del primer sprint.

Finalment, l'últim canvi que hem fet ha estat procurar fer els "merges a *dev*" mitjançant els pull requests per tal "d'avisar" a la resta del equip sobre un nou canvi a dev i per poder visualitzar més directament els canvis introduits.

3.1.3. Backend

3.1.3.1. Canvi de llenguatge

Hem decidit canviar el llenguatge que utilitzarem per al backend sencer de Kotlin a Java. Donat que el backend serà el més independent possible respecte al *frontend*, que fem servir dependències amb llibreries que permeten estalviar la redacció de codi i que tota la documentació i exemples que podem trobar està majoritàriament en Java, no té cap sentit fer servir Kotlin (més enllà de voler aprendre un llenguatge nou).

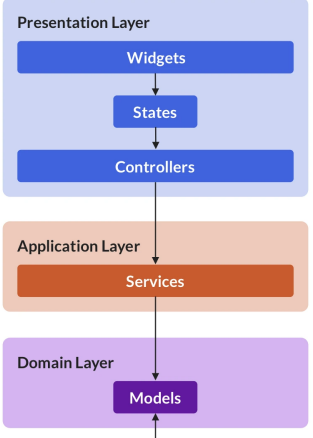
3.1.3.2. Nom de les variables i atributs

En un principi vam decidir que els atributs de classe seguirien l'estil _attr, però hem decidit no fer ús d'aquesta sintaxi, ja que els atributs es mapejen directament a la base de dades i a les funcions setters i getters (es generen de manera automàtica), i volem que tot aquest codi sigui més fàcil de llegir.

3.2 Canvis principals a l'arquitectura amb justificació

3.2.1. Frontend

En quant al sprint anterior no hi han hagut canvis drastics, seguim fent servir el patró repositori per a accedir als elements del backend.



La estructura que seguim segueix la del diagrama que es mostra a continuació. La capa de presentació engloba gran part del codi

què escrivim al front. En aquest sprint ens hem centrat en separar una mica les responsabilitats amb widgets o components intermitjos i amb controladors (*notifiers*) per tal de separar la lógica.

En quant a la *Application Layer*, encara no hem introduït serveis com a tal, pero com a capa si que ens interessa tenir-la present. Durant un temps, el tema dels permisos de la localització i el seu estat estaven presents en un servei apart. En teoria segueix sent un servei, pero en aquest cas és un servei "extern" (un package).

Finalment, a la capa de dades, hem delegat la responsabilitat de instanciar els diferents models (DTO -> Model) l'hem encapsulat en el model ja que Dart compta amb mètodes de tipus "factoria", que permet instanciar els models apartir d'un diccionari JSON.

3.2.2. Backend

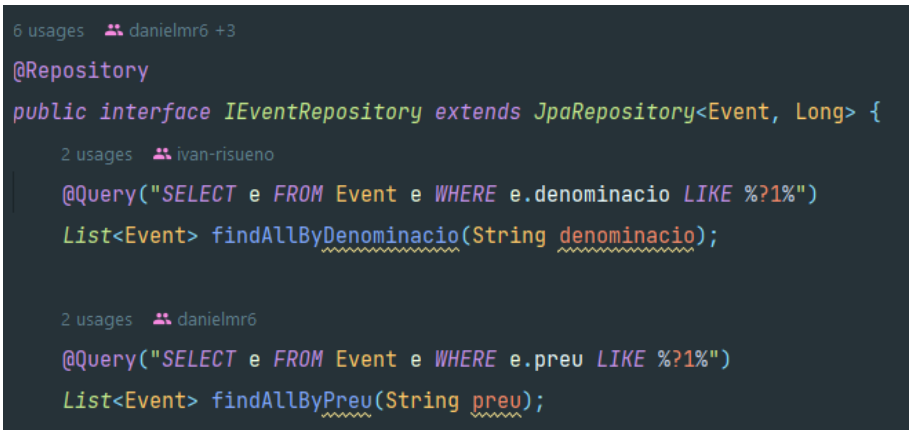
Respecte a l'arquitectura del *backend* hem utilitzat com a patró principal ***Adaptador***, el qual és un patró que permet que dos components amb interfícies diferents puguin col·laborar. En el nostre cas hem afegit interfícies a la capa de dades per a que les classes de serveis puguin utilitzar métodes 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 innecessari. Aquesta API funciona de la manera següent: donada una clase 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:



El Patró **Model-View-Controller (MVC)** és un patró de disseny de programari que s'utilitza de manera comuna en el desenvolupament d'aplicacions web. MVC és una manera de separar la lògica de negoci d'una aplicació de la lògica de presentació. L'objectiu és tenir una aplicació escalable, fàcil de mantenir i fàcil d'entendre.

MVC consta de tres components principals:

Model: Aquest component representa la lògica de negoci de l'aplicació. Aquí és on es defineixen i manegen les dades i les regles de negoci. Al Backend, el model s'encarregaria de la interacció amb la base de dades i de realitzar les operacions necessàries per a processar la informació.

Vista: Aquest component representa la interfície d'usuari (UI) de l'aplicació. La vista s'encarrega de mostrar les dades a l'usuari i de permetre-li interactuar amb l'aplicació. Al Backend, aquesta vista no seria necessària ja que no es mostra informació a l'usuari.

Controlador: Aquest component actua com a intermediari entre el model i la vista. El controlador rep les sol·licituds de l'usuari i les processa en el model. Després, envia la resposta del model a la vista corresponent. En un Backend, el controlador s'encarregaria de rebre les sol·licituds HTTP, processar la informació necessària en el model i retornar la resposta corresponent.

Per aquest mateix motiu el *backend*, programat utilitzant 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 anomenen 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).

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 proveïr al *frontend* de la informació i la interacció amb la BD necessàries i per a poder gestionar una part de l'aplicació.

3.3 Canvis principals a l'estructura de codi amb justificació

3.3.1. Frontend

3.3.1.1. Estructura de *packages*

L'estructura de packages s'ha mantingut respecte a l'sprint 1, però amb alguns canvis. Continuem tenint tota la implementació al package lib, i un altre de test amb els testos crítics.

Dins de lib, hem afegit una carpeta anomenada model on hem inclos els diferents models que s'utilitzen per les vistes, com EventModel, TagModel, UserModel, i més. Els subpackages per èpiques els hem fet servir més bé per a vistes o funcionalitats, amb la vista i elements addicionals com ara controladors.

Hem mantingut el subpackage de respository i route com s'havien descrit a l'sprint anterior, incloent els nous repositoris i configuracions de rutes al subpackage corresponent.

3.3.2. Backend

3.3.2.1. Estructura de *packages*

Pel que fa als *packages*, a dins de la capa de dades, tenim dos *subpackages*: un per a les classes de tipus repositori, encarregades de fer consultes SQL a la BD, i un altre per a les classes de tipus servei, encarregades de fer ús de les classes repositori. D'altra banda, a la capa de domini, la hem dividit també en dos *subpackages*: un per la lògica de negoci de cada model de dades, i un altre per als models que utilitzarem al sistema.

Un petit esquema de la estructura actual del projecte dins de *src/main/java/*:

```
CultureFinderBackend/
├─ controllers/
│  ├─ dtos/
│  │  ├─ AssistanceDTO.java
│  │  ├─ EventDTO.java
│  │  ├─ ...
│  ├─ APIExceptionHandler.java
│  ├─ EventController.java
│  ├─ IncidentController.java
│  ├─ UserController.java
│  ├─ ...
├─ data/
│  ├─ mappers/
│  │  ├─ DataAssistanceMapper.java
│  │  ├─ DataEventMapper.java
│  │  ├─ ...
│  ├─ models/
│  │  ├─ Assistance.java
│  │  ├─ AssistanceId.java
```

```
│   │   ├── ...
│   ├── repositories/
│   │   ├── IEventRepository.java
│   │   ├── IIncidentRepository.java
│   │   ├── IUserRepository.java
│   ├── services/
│   │   ├── EventService.java
│   │   ├── IEventService.java
│   │   ├── ...
├── domain/
│   ├── businesslogic/
│   │   ├── EventLogic.java
│   │   ├── JWTLogic.java
│   ├── exceptions/
│   │   ├── Error.java
│   │   ├── ObjectAlreadyExistsException.java
│   │   ├── ...
│   ├── mappers/
│   │   ├── DomainAssistanceMapper.java
│   │   ├── DomainEventMapper.java
│   │   ├── ...
│   ├── models/
│   │   ├── AssistanceEntity.java
│   │   ├── EventEntity.java
│   │   ├── ...
│   ├── services/
│   │   ├── DomainAssistanceService.java
│   │   ├── IDomainAssistanceService.java
│   │   ├── ...
├── BackendApplication.java
├── other resources...
```

Cal dir també pel que fa a les classes *mappers*, dins de *packages* “mappers” a la capa de dades i domini, són interfícies o classes abstractes que s’implementen de manera automatica en temps de compilació al un directori de codi generat fora de */main/*.

3.4 Comparació amb funcionalitats implementades a la 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
Sistema multi idioma (català, castellà, anglès) [Només menú]	Funcionament fora de Catalunya
Recomanació d'activitats segons els interessos dels usuaris	Possibilitat de promocionar activitat cultural
Compartir esdeveniments a través de xarxes socials o URL	
Valoració d'esdeveniments entre 1 i 5 punts	Xat amb els responsables dels esdeveniments

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

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 del primer sprint, 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.

També tenim uns altres tipus de tasques implementades com podrien ser llistes de favorits i personalitzades, recomanacions d'esdeveniments i valoració de les activitats que dins de la nostra aplicació són funcionalitats complementàries que li donen un valor extra al projecte.

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 segon sprint per problemes de configuració amb la base de dades de l'Agenda Cultural i el procés corresponent per parsejar la informació.

Un altre conjunt de tasques que s'han allargat amb el pas del temps duran aquest segon sprint ha sigut l'impacte important que ha tingut el fet d'afegir DTOs, ja que eren modificacions importants a la arquitectura al mateix temps que vam haver de començar a securitzar l'API. 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 la pàgina d'administració encara queda terminar-la amb les crides a la nostra API, ja que en aquest sprint s'ha començat a implementar la base pero amb proves *Dummy*. 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 del Frontend. Cal destacar que tampoc s'ha implementat la modificació de perfil en aquest sprint però es realitzarà de cara al tercer.

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.