UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Culture finder

SPRINT 1: 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 primer sprint

De cara al frontend, ens hem centrat principalment en implementar les vistes més emblemàtiques de la nostra aplicació: el calendari, el mapa i la llista d'esdeveniments.

En un principi volíem implementar el 'walking skeleton' de la nostra aplicació, que inclou, a més a més, les funcionalitats de registrar-se i tancar sessió, i la d'enviar incidències; però degut al temps invertit en aprendre a utilitzar el framework Flutter i el seu llenguatge Dart, no hem pogut assolir totes les tasques que haviem assignat al començament del sprint.

Finalment, ens hem decantat per acabar les vistes que hem designat com a més crítiques: Vista dels detalls (camps) d'un esdeveniment, la vista de la llista de tots els esdeveniments amb paginació per millorar el rendiment, la vista dels esdeveniments en un calendari i finalment, la vista del mapa.
Aquestes vistes ens han servit per plantejar-nos com estructurar el projecte, tant en arquitectura mitjançant patrons com el Repository per accedir a les dades de l'API, com per decidir l'organització dels diferents arxius en carpetes.

Pel que fa al backend, l'equip ha aconseguit implementar la gran majoria de les funcionalitats assignades al sprint. Aquestes funcionalitats inclouen els diferents endpoints rellevants a les històries d'usuari, els quals són accessibles a qualsevol usuari mitjançant una eina que genera automàticament un entorn Swagger localitzada a la ruta: http://nattech.fib.upc.edu:40380/swagger-ui/index.html

Aquest URL mostra tots els endpoints que es troben actualment en producció i s'actualitza automàticament quan s'afegeixen nous endpoints o mètodes.

L'altra gran tasca en la qual han treballat el backend consisteix en formalitzar les dades que retorna la API de la generalitat de la qual partim. Gràcies a com funciona aquella base de dades (la nostra intuïció de com funciona), no hi ha cap conveni/protocol que especifiqui o formalitzi els valors de cada columna de la seva base de dades ni quines columnes poden contenir valors nuls, strings buits o un sol string separat per comes per tal d'indicar una llista de possibles valors.

Per posar un exemple, la primera columna de la seva base de dades s'anomena codi i conté un nombre que en un principi creiem que havia de ser únic i que per tant seria la seva clau primària. Més endavant vam trobar un conjunt de tuples que tenien el mateix valor al camp del codi. Un altre cas seria el tema dels events online, els quals no tots fan servir els mateixos indicadors per denominar-los com a tal; la majoria fan servir dates d'inici i de fi impossibles (9-9-9999), altres entrades feien servir exemples semblants pero no consistens (22-2-2222) o potser mencionant al final de la descripció que l'event en qüestió no era presencial.

Tot això per dir que tot i que la "historia d'usuari" de tenir dades consistents esta com a resolta, som conscients de que haurem de fer-li una ullada més endavant quan ens topem amb un nou cas el qual no haviem previst.

1.2. Sprint master report

Per aquest primer sprint la visió principal es centrava en implementar totes les funcionalitats crítiques de les quals depèn la nostra aplicació. En altres paraules, la idea era tenir un cop finalitzat el Sprint, el 'walking-skeleton' de la aplicación.

En el cas de backend, aquestes funcionalitats consisteixen en recollir les dades de la API que ens proporciona la generalitat, mantenir-les persistens en una base de dades particular i exposar els diferents endpoints per tal de que el front end pogués accedir-hi a aquesta informació.

A més a més, també vam incloure el que anomenem la "neteja" d'aquestes dades, que consisteix principalment en formalitzar i concretar l'estat i els possibles valors que ha de tenir o pot tenir cada camp, com per exemple, definir quins camps poden o no ser nulls i quin hauria de ser el seu valor per defecte en cas de que aquests camps siguin nulls.

1.3. Que hem fet cada membre de l'equip

Equip del backend

- **Daniel**: Respecte al meu treball pel backend d'aquest sprint, s'ha basat principalment en la implementació de les crides API que utilitza el model *Event* i el model *Incident*. Al principi em vaig focalitzar en la configuració de la base de dades, la qual ens va retardar bastant tot el tema d'aprenentatge i formació. Després a l'hora de programar, les seccions en les quals em vaig centrar van ser: processar correctament el preu i les dates finals i inicials proporcionades per l'API de l'Agenda Cultural, permetent tots els tipus de filtratges que tindrà el nostre producte pels esdeveniments amb una mateixa crida a l'API, a més de les crides individuals per ambit, categoria i d'altres categories i per rang de dates. Addicionalment, he estat fent una primera versió del model *Incident* esmentat anteriorment, el qual ens facilita la creació d'incidències per reportar qualsevol tipus d'error.

  Per acabar, vaig estar implementant la tasca de crida periòdica a l'API de l'Agenda Cultural, ja que ens interessa fer la sincronització una vegada al dia per tenir les nostres dades actualitzades. Vaig estar fent *pair programming* amb Ivan per acabar de treure l'actualització periòdica de les dades, ja que ens hem trobat el problema de que si esborrem la BD i tornem a carregar-la, a més de que els esdeveniments antics es perden, els identificadors canvien i això implica que la informació personalitzada de cada usuari romandria inconsistent.

- **Iván(*backend leader*):** La meva aportació pel que fa al *backend* han sigut, sobretot, el disseny de la infraestructura inicial del codi dividida en *packages*, *subpackages* i classes. A més, vaig establir la estructura de la base de dades que utilitzaríem i vaig fer la connexió per tal de poder fer-la servir.

  Havent requerit de bastanta formació prèvia, vaig especificar quines classes necessitariem per a cada model de dades i quines funcionalitats implementarien aquestes classes. Tot seguit, vaig desenvolupar la primera versió del model d'usuari(classe *User*), així com les funcions i mètodes

necessaris per a poguer guardar a la base de dades, editar un usuari en concret, eliminar-lo, i obtenir els que teniem guardats mitjançant la seva clau primària o obtenir-los tots amb o sense paginació. Després, vaig fer exactament el mateix amb el model dels esdeveniments(classe *Event*). Aquest últim, però, em va portar molta més feina ja que requerim de mètodes per fer *parsing* de les dades de l'API de l'Agenda Cultural(vaig dissenyar les operacions que la criden i processen la seva informació) a instàncies d'*Event*, així com haver-ne d'agafar correctament els atributs de cadascun i els valors que prenen. També vaig fer les funcions de processament d'àmbits, categories, i altres categories, i el *deploy* de l'aplicació a la màquina virtual de Virtech.

Per acabar, ja al final de l'sprint, he estat fent *pair programming* amb en Dani per acabar de treure l'actualització periòdica de les dades, ja que ens hem trobat el problema de que si esborrem la BD i tornem a carregar-la, a més de que els esdeveniments antics es perden, els identificadors canvien i això implica que la informació personalitzada de cada usuari romandria inconsistent.

- **Marc Rodriguez:** La meva aportació al Backend aquest esprint ha estat principalment la formació i la inicialització del projecte mitjançant el framework Spring Boot. A més vaig estar desenvolupant maneres de com realitzar la crida de dades periòdica de manera eficient. També vaig col·laborar en algunes tasques com el processament de dades. A més de col·laborar a les reunions setmanals de Backend sobre com implementar la base de dades i quin servidor utilitzar per pujar l'aplicació.

Equip del frontend

- **Oscar:** Durant aquest sprint he focalitzat el meu treball al frontend, concretament a les vistes i funcionalitats relacionades amb el mapa. També m'he encarregat de les vistes de crear usuari i el model d'usuari.
  Per tal crear un mapa funcional he hagut d'utilitzar els plugins de Google Maps i permission handler. El primer m'ha servit per mostrar un mapa interactiu a la view, el qual es pot centrar a l'ubicació actual del dispositiu i mostra els diferents esdeveniments al mapa, permetent fer una preview d'aquests si es seleccionen. El segon plugin m'ha servit per tot el que està relacionat amb els permissos d'android d'ubicació.

Per a la creació d'usuari he construit una vista amb un formulari i una altra per seleccionar els interessos d'entre les categories de tots els esdeveniments del sistema. També he creat el model d'usuari i navegabilitat entre algunes vistes.

- **Marc Duch(*frontend leader*):** En aquest Sprint he fet de *Scrum master* i com a tal m'he encarregat de dirigir les primeres reunions per tal de dividir-nos la feina, decidir quines històries incloure al sprint amb l'equip i d'escriure el Sprint planning report.
  Com ha desenvolupador del front end, en aquest sprint m'he encarregat de generar el codi inicial que es basava en el *EventModel* i una implementació mock de la futura classe *EventRepository*, que com el nom indica, és una classe que aplica el patró repository per tal de crear una abstracció a la hora de aconseguir els EventModels. Tot i que aquest codi inicial el vaig escriure amb la intenció de que fos temporal, degut a la meva falta de coneixement, el vaig haver de rescriure varies vegades al llarg d'aquestes setmanes, fins que finalment vaig implementar la primera crida de la API un cop el back va pujar a producció el seu codi.
  En quant a vistes, m'he centrat principalment el la vista de la llista d'esdeveniments a la qual vaig afegir-hi més tasques després de aprendre que era la paginació i més importat, perque l'haviem d'implementar.
  Apart del desenvolupament, també he dedicat molt de temps a aprendre com funciona el framework i el llenguatge intentant donar suport als altres membres del front.

- **Miguel:** En aquest sprint, la meva feina ha estat enfocada a implementar vistes per la part de frontend. La primera setmana la vaig dedicar sobre tot a formació, per familiaritzar-me amb l'entorn de flutter i el llenguatge dart. A continuació vaig fer la vista del calendari, que tot i que era una tasca la vaig implementar en el que podrien ser dues parts. Primer, implementar el calendari, que mostra un mes en forma de quadrícula, i permet seleccionar un dia. Segon, mostrar events al calendari, de manera que quan es selecciona un dia que té events, es mostra a sota del calendari el nom dels events del dia en un quadre de text. Una vegada implementada la vista del calendari, vaig passar a la de mostrar informació detallada d'un event. Després de provar a implementar que es cridi al repositori per obtenir les dades d'un event donada la seva id, vaig descobrir que era més fàcil i no donava errors si es passava un event directament. Amb l'event passat a la vista, es van mostrant els camps en un cert ordre, sempre que

hi existeixin (alguns són obligatoris i d'altres no).

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 primer sprint:

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

El nostre pla inicial per planejar la feina que volíem fer durant l'sprint consistia en implementar el que vam considerar com les funcionalitats bàsiques de la nostra aplicació, o el que s'anomena, el 'walkin-skeleton'. Aquest walking-skeleton estava compost de les següents funcionalitats:

- 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 (la resolució d'aquestes la deixem per més endavant).

Per tal de facilitar l'inici del projecte vam assignar dos nous rols, els Lead Backend i Frontend (Ivan Risueño i Marc Duch respectivament). Aquests dos rols serveixen com a punts de referència per a cada un dels codebases a la hora de tancar els pull requests, inicialitzar el projecte, i en el cas del backend, pujar al servidor de producció el codi.

Acabat l'Sprint, tot i que no tenim del tot aquest walking skeleton que volíem tenir en la seva totalitat, hem aconseguit avançar molta feina. I més important encara, hem après com funcionen els frameworks/llenguatges que farem servir de cara als següents sprints.

2.1.2. Sprint Retrospective

Per tal d'aprendre del procés, l'equip es va reunir el 30 de Març per fer una reunió retrospectiva sobre com havia anat aquest primer sprint. Per una banda, sabíem que anàvem a tenir problemes per acabar totes les tasques que vam assignar a l'sprint, pero a la vegada ens hem sorprès a la quantitat de feina que hem aconseguit fer de cara al tancament del sprint.

Amb aquest sprint hem après que la *formació*, i potser més important, el temps que es triga en realment en aprendre a utilitzar una nova tecnologia, és un aspecte que haurem de valorar més a la hora de planificar els següents sprints i tenir el coneixement necessari per tal d'estimar de manera precisa.

Una de les coses sobre les que hem parlat, ha estat la ambigua *Definition of Done* que havíem definit anteriorment tant per les històries d'usuari com les tasques que les formen. Per aquest motiu, hem tornat 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 pasa a estar acabada.
- **In Progress** - Història d'usuari ha estat assignada a un sprint
- **Ready for Test** - Algunes històries d'usuari, les que creiem més crítiques, estaran marcades mitjançant *tags* al Taiga, indicant que requereixen ser testejades. Hem decidit que farem 2 tipus de test (a nivell d'història):
    - Test funcional -> Test que ve descrit pels criteris d'acceptació
    - Smoke Test -> Test no exhaustiu de punts critics de l'aplicació, la idea d'aquest test és que salti si hi ha un problema que afecta a una part crítica del nostre codi.

  Aquesta fase de "*Ready for Test*" només afectarà a aquelles històries que estiguin marcades com a crítiques o que requereixin algun d'aquests dos tests esmentats. Només podrà passar a la següent fase, si i només si, passen els tests que s'han creat per aquella història.
- **Closed** - La historia d'usuari (i per tant les seves tasques) està implementada i funciona correctament. En el cas de ser una història critica/que requereixi testeig, haurà passat prèviament els tests sense cap problema.

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*.

Com hem esmentat anteriorment, la nostra definició de quines històries requereixen tests i quins test haurien de ser / com d'exhaustius haurien de ser aquests tests no era gaire concisa. A més a més, aquesta fase donava a entendre que cada tasca havia de estar testejada i això ens ha mantingut estancats durant aquesta última setmana del sprint, tenint la gran majoria de tasques ja implementades a la columna *Ready for Test*.

Per tant, amb el canvi aquest, les fases d'una tasca passen a ser:
- **New** - Nova tasca creada al sprint
- **In Progress** - La tasca ha estat assignada a un membre de l'equip
- **Un-documented** - 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.

Un altre procés que ha sorgit de forma espontània, però que volem incorporar a les nostres convencións de manera continuada, és l'ús dels *pull requests* a l'hora d'ajuntar les diferents branques/features amb la branca de producció (**dev**).

Amb aquest procés podem assignar un reviewer (o més) per a comprovar quins són els canvis que hi entren i resoldre els conflictes del merge.

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

El nostre pla de cara als següents sprints és integrar aquestes accions i proves de testatge automàtic en el nostre procés de finalitzar tasques i històries d'usuari.

Finalment, l'últim punt sobre el que hem parlat durant la retrospectiva han estat les *convencións de codi* que es van pactar al Inception 2. La idea d'aquesta convenció era fer que el codi fos consistent a nivell estètic i més agradable a la hora de llegir, però a la pràctica hem trobat que el haver d'estar pendents de com escrivíem el codi o el haver de tornar enrere per re-escriure els noms de variables ens estava traient temps de desenvolupament, per exemple en aspectes com afegir un caràcter '_' per variables privades, quan en Java s'utilitza el "private" de manera explícita. El temps emprat no era una barbaritat, pero el benefici de tenir el codi consistent a nivell estètic no ens sortia rentable, i per tant, hem decidit que ja no seguirem la guia escrita que vam definir al inception 2.

Un altre motiu per aquesta decisió és per les convencions o "bones pràctiques" que estableixen els dos frameworks (el del front i el del back), les quals no son sempre compatibles.

Respecte a la comunicació, estem molt contents de l'horari i les reunions que hem efectuat al llarg del sprint. Les quatre hores a classe ens han ajudat molt a

solucionar problemes que afectaven tant el Frontend com el Backend amb l'ajuda del professor i les reunions setmanals dels diumenges ens permeten com a grup mantenir-nos al dia de què s'estava fent i com a equips especialitzats, ajudar-nos mútuament a solucionar problemes que ens havíem trobat al llarg de la setmana amb els coneixements adquirits de manera pràctica o mitjançant formacions.

Amb el discord hem après a crear threads que ens permet crear sub-chats sobre un topic, com poden ser threads per discutir certes funcionalitats, reportar problemes específics al front/back o per recollir links a recursos per a cada equip.

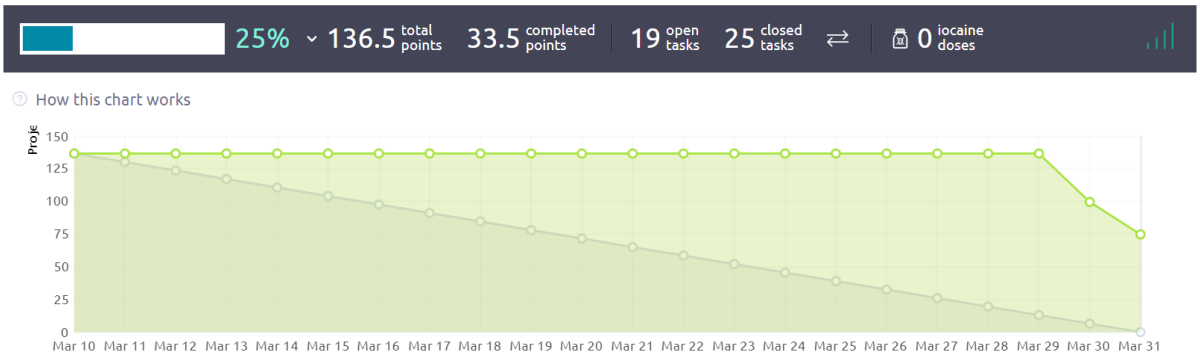
Com a grup estem molt contents amb la feina feta tot i que no haguem aconseguit acabar totes les tasques amb les quals havíem començat. De cara als següents sprints haurem de fer un esforç de més per poder lliurar totes les funcionalitats que tenim incloses a la NOT LIST. Per sort, el nivell de coneixement de les eines ha augmentat i això hauria de facilitar el desenvolupament d'ara endavant.

2.2. Estadístiques del projecte

Els diferents gràfics *burndown* que mostrem a continuació informen de l'avanç del primer 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

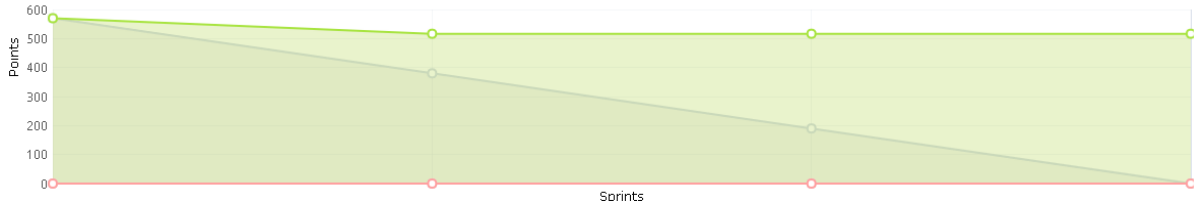
Aquest gràfic mostra el progrés aproximat o projectat del sprint basat en les tasques tancades estimades de les històries d'usuari. Les històries d'usuari estimades acabades reals se segueixen més amunt. Degut al problema que vam tenir per definir quan es tancava una tasca, es pot veure que a la part 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. Sabiem 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 formació, i aquesta formació ens facilitarà més endavant amb les següents històries.

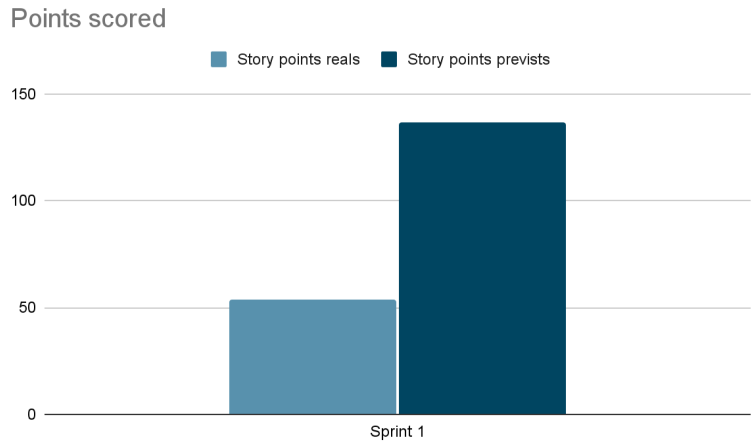
L'altre detall aparent es la baixada dramàtica dels últims dos dies. En parlem més en detall a la retrospectiva, pero la principal raó es deu a que en el moment de definir el DoD i el *lifecycle* de les tasques, no vam acabar de definir com *acabar* una tasca i nos ens vam adonar fins a la penultima reunió.

2.2.2. Release Burndown



En relació a la resta de sprints és encara més evident la quantitat de feina que arrossegarem als següents sprints.

2.2.3. Velocity chart



Finalment tenim el velocity chart que actualment només té la "velocitat" del primer sprint en relació a la estimada..

3. Changelog

Summary of main changes with respect to previous sprint and implementations of features

3.1 Canvis principals a la metodologia amb justificació

3.1.1. General

3.1.1.1. Canvi de DoD

Com s'explica amb més detall al Sprint Retrospective (apartat 2.1.2.), hem canviat la Definition of Done que havíem proposat a la segona fase d'incepció i amb la que vam començar a treballar en aquest primer Sprint. Les fases de les tasques passen de ser {New, Ready, In Progress, Ready for Test, Closed}, a ser {New, In Progress, Un-documented, Closed}. El canvi principal té a veure amb els tests, els quals hem decidit no fer per a totes les tasques, sino només per aquelles que són funcionalitats bàsiques de la nostra aplicació. D'aquesta manera, evitem invertir massa temps en tasques "*nice to have*" com poden ser afegir un esdeveniment a favorit, la qual cosa és important, però a nivell global del producte no és la part més important.

3.1.2. Frontend

3.1.2.1. Convencions de codi

Vam acordar unes convencions de codi a la fase d'incepció 2, però no les hem seguides estrictament. Cada membre del front ha programat com ha considerat, tot i que degut a la bona comunicació entre membres la manera d'escriure codi ha resultat similar.

La majoria de variables les hem escrit en camelCase i les classes en UpperCamelCase, però sense revisions per assegurar que tot sigui així, ja que això ens consumiria temps que no ens ha sobrat en aquest primer Sprint. On hem sigut menys estrictes ha sigut a l'hora d'escriure if i else, i l'ordre d'elements a les Classes.

Per les variables privades hem utilitzat un guió "_" al començament, ja que així és com es declaren en Dart, per tant aquí sí hem seguit la convenció. Però gairebé no hem utilitzat variables privades, ja que la majoria d'atributs, especialment de classes de model i controladors, els criden altres classes, i per tant han de ser pùbliques.

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

En relació als canvis principals a l'arquitectura, des del començament hem aplicat una arquitectura en tres capes per cadascuna de les dues aplicacions, frontend i backend. Hem decidit dissenyar l'arquitectura d'aquesta manera donat que:

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

- Aquest disseny permet no fer assumcions sobre com són els tipus de dades que s'utilitzen.

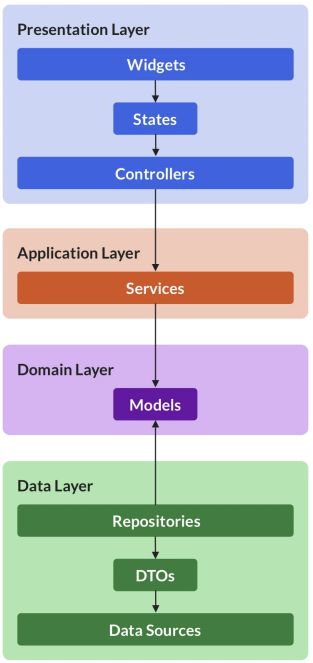
- Cada capa té un grup de funcionalitats perfectament definides. Això fa que sigui senzill determinar on anirà un mètode, classe o atribut.

- La capa de dades és perfectament reutilitzable: conté funcions i mètodes senzills.

- En el cas del backend, la capa de dades també compta amb completa independència respecte a la tecnologia utilitzada (PostgreSQL).

3.2.1. Frontend

A partir del patró de disseny MVVM, hem utilitzat una arquitectura adaptada que es basa en el package Riverpod, el qual utilitzem per la gestió d'estats. Aquesta es compon de 4 capes: presentació, aplicació, domini i persistència. En la següent figura es mostra un esquema:



Hem escollit una arquitectura similar a MVVM perquè separa la lògica de negoci i de presentació d'una aplicació de la interfície d'usuari, fent que el codi sigui més testejable i fàcil d'entendre. Quan s'utilitza MVVM, la view no pot accedir directament al model. Més aviat, el view model gestiona els inputs dels usuaris i transforma les dades del model perquè es pugui presentar fàcilment. El model de vista està connectat a la view mitjançant data binding que utilitza el patró de disseny observador. Tenir aquest view model intermedi es especialment útil quan les dades retornades del backend no s'ajusten bé a la vista que volem mostrar. A diferència de MVVM hi ha services i repositories per tal d'encapsular l'accés a dades i fer caching.

En la nostra capa de presentació els controllers contenen la lògica de negoci, gestionen l'estat dels widgets i interactuen amb els repositoris de la capa de persistència. Actualment tenim la lògica d'aplicació als models i controllers com en l'arquitectura MVVM, però tenim com a objectiu crear la capa d'aplicació perquè accedeixi als repositoris necessaris i així a mantenir el principi de responsabilitat única.

En la capa de domini, model és un model conceptual del domini que conté dades i el seu comportament. Aquestes dades es poden representar per un conjunt d'entitats, que en el nostre cas són esdeveniments i usuaris, juntament amb les seves relacions, mentre que el comportament està codificat per alguna lògica de negoci per manipular aquestes entitats.

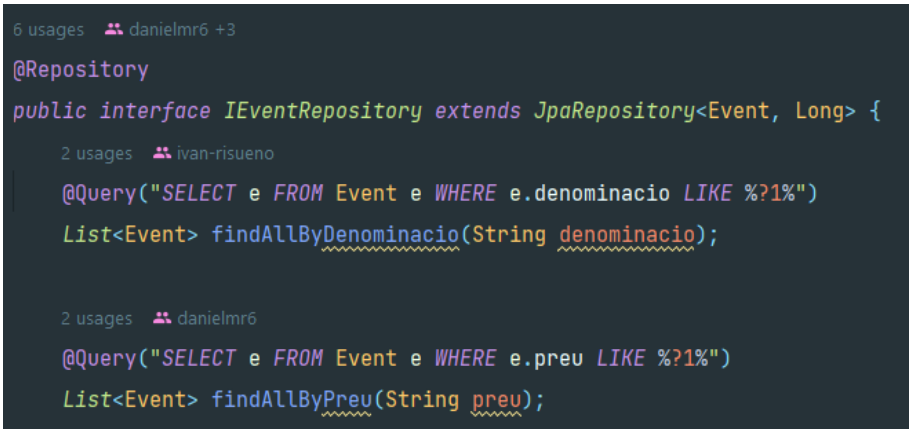
Un dels patrons estructurals més importants que apliquem és el repository, que s'encarrega d'aïllar els models de domini de la implementació detallada de les fonts de dades a la capa de dades. També ens permet realitzar operacions com el caching de dades i ens transforma els objectes data transfer a models validats que la capa de domini pot interpretar.

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:



Per acabar de l'arquitectura del *backend*, tenim pensat de cara a propers sprints implementar algun patró més com per exemple ***Factoria***, el qual és un patró de disseny creacional que proporciona una interfície per a crear objectes en una superclasse, mentre permet a les subclasses alterar el tipus d'objectes que es crearan.

3.3 Canvis principals a l'estructura de codi amb justificació

3.3.1. Frontend

3.3.1.1. Estructura de *packages*

La nostra aplicació de Flutter conté un package *lib*, on es troba tota la implementació del frontend. També conté un package *test* que de moment agrupa tots els testos de les funcionalitats crítiques, és a dir, els smoke tests.

Dins de *lib* es troben els subpackages de cada funcionalitat o èpica, els quals dins contenen un subpackages per les views i els view models d'aquestes viste. *lib* també té un package amb els models de domini, que en el nostre cas només són els esdeveniments i usuaris. A demès *lib* conté un subpackage anomenat *route* amb els fitxers que configuren les rutes de les views i el subpackage *repository* que conté els fitxers que apliquen el patró.

Actualment també tenim un subpackage dins de *lib* que conté els widgets usats a les views. Tenim els widgets en aquest package enlloc de dividir-los per view per facilitar la reutilització de widgets.

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/
│  ├─ EventController.java
│  ├─ IncidentController.java
│  ├─ UserController.java
├─ data/
│  ├─ repositories/
│  │  ├─ IEventRepository.java
│  │  ├─ IIncidentRepository.java
│  │  ├─ IUserRepository.java
│  ├─ services/
│  │  ├─ EventService.java
│  │  ├─ IncidentService.java
│  │  ├─ UserService.java
├─ domain/
│  ├─ businesslogic/
│  │  ├─ EventLogic.java
│  ├─ models/
│  │  ├─ Event.java
│  │  ├─ Incident.java
│  │  ├─ User.java
├─ BackendApplication.java
├─ other resources...
```

Cal dir també que la estructura dins de *src/test/java/* és la mateixa, on cada classe tindrà al final del seu nom el sufix *Tests*. D'aquesta manera, assegurem que s'entengui perfectament que comportament s'està testejant a cada escenari.

3.4 Comparació amb funcionalitats implementades a la NOT-LIST

Aquesta es la Not list definitiva amb la qual vam acabar a Inception 2, tot i que hem destacat aquelles que s'han implementat en aquest primer sprint.

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

Compartir esdeveniments a través de xarxes socials o URL	
Valoració d'esdeveniments entre 1 i 5 punts	Xat amb els responsables dels esdeveniments
Creació, modificació i alta de perfils	Traçat del camí fins l'esdeveniment / direccions
Notificar assistència als esdeveniments	Sistema multiplataforma
Llista d'esdeveniments favorits	Fòrum de comentaris sobre els esdeveniments
Creació de llistes d'esdeveniments personalitzades	
Sistema per reportar errors d'esdeveniments	
Pàgina per administrador per consultar i respondre als reports dels usuaris	
Sincronització amb Google Calendar	
Apartat de dubtes freqüents i ajuda	
Sistema de notificacions per als usuaris sobre els esdeveniments als que han afegit a alguna llista, assistiran o els poden interessar	
Estadístiques de l'esdeveniment (número assistents, valoració mitjana...)	
Mostrar els esdeveniments més populars a través d'un mapa de calor	

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 altres, estan marcades de color groc. Aquestes funcionalitats inclouen parts crítiques de l'aplicació que han portat una complexitat alta i altres que encara no ser massa complexes, no se'ls ha donat prioritat alta. Per exemple, la cerca d'esdeveniments amb filtres està implementada al Backend, però no s'han fet les vistes al Frontend. I la sincronització periòdica de dades ha tingut una complexitat més alta de l'esperat. Aquestes dues funcionalitats no s'han implementat a temps a causa de la seva complexitat. Les altres dues, sistema per reportar errors d'esdeveniments i creació, modificació i alta de perfil, no s'han implementat a causa que no eren tan prioritàries com les altres. En el cas dels perfils, s'ha implementat la lògica de creació d'usuari i l'inici de sessió a la part del Backend, però no s'han fet les vistes al Backend. Tampoc s'ha implementat la modificació de perfil. I sobre el sistema per reportar errors només s'ha creat la taula d'incidències al Backend.

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 a l'esprint 1.