

**SPRINT 2: DOCUMENTATION**

[GitHub](https://github.com/Green-Wheel) - [Taiga](https://tree.taiga.io/project/arnau147-pes-green-whee/timeline) - [Drive](https://drive.google.com/drive/folders/18mG6F0EBEJ3B5F38gJwob7Y3os8cv7j3?usp=sharing) - [Project record track](https://docs.google.com/spreadsheets/d/1AAiR52U9Tz-K_jhisDPsraG94YYvX-46vuqM6bt2QEA/edit?usp=sharing)

| **Cognom** | **Nom** | **Responsable** | **UPC e-mail** | **Taiga** | **GDrive** | **GitHub** |
| --- | --- | --- | --- | --- | --- | --- |
| Benali | Isslam | Service | isslam.benali@estudiantat.upc.edu | Isslam1 | isslam.benali@estudiantat.upc.edu | Isslam1 |
| Giménez | Arnau | Dev-1 | arnau.gimenez.bolta@estudiantat.upc.edu | arnau147 | arnau.gimenez.bolta@estudiantat.upc.edu | arnau147 |
| Gutiérrez | Miguel | Demo | miguel.gutierrez.jariod@estudiantat.upc.edu | mikierxxv | miguel.gutierrez.jariod@estudiantat.upc.edu | MikierXXV |
| Migó | Cristina | Dev-2 | cristina.migo@estudiantat.upc.edu | crismigo | cristinamigo01@gmail.com | crismigo |
| Oliveras | Daniel | Inception | daniel.oliveras@estudiantat.upc.edu | daniou | daniel.oliveras@estudiantat.upc.edu | daniou |
| Ollé | Àlex | Dev-3 | alex.olle@estudiantat.upc.edu | aolle99 | aolle99@gmail.com | aolle99 |
| Orensanz | Andreu | Inception | andreu.orensanz@estudiantat.upc.edu | andreuorensanz | andreu.orensanz@estudiantat.upc.edu | andyfratello |

**ÍNDEX**

[**1. Introduction**](#_c90dn47mqa0l) **4**

[1.1. Resum per sprint](#_tpsqqt5wzxbn) 4

[1.2. Sprint master report](#_rh36ad36w7pd) 4

[1.3. Què ha fet cadascú](#_lz540fqujrl7) 7

[1.4. Avaluació de companys](#_ml0o1p9m8f55) 18

[**2. Requisits**](#_70rwu5sjobpj) **20**

[2.1. Not List](#_4orhgki8j36o) 20

[2.2. Backlog](#_92cvirpr68d3) 22

[2.3. Requeriments no funcionals](#_kritwj1w3384) 24

[2.4. Aspectes transversals](#_8te3grf70uxe) 26

[2.5. Serveis](#_yoyozg65c9ij) 30

[2.6. Stakeholders](#_3mio54ig9bkq) 32

[**3. Agile ceremony**](#_hrooxjpoz3n5) **35**

[3.1. Reunions de Sprint Planning, review i retrospectiva](#_rntlxx8uv2gj) 35

[3.1.1. Sprint Planning](#_gscvr1wd32fs) 35

[3.1.2. Retrospectiva](#_qkfsm5fmdyie) 46

[3.1.3. Review](#_wc0ouamkhy8s) 50

[3.2. Estadístiques de l'avanç del projecte](#_lhhufi25mkua) 50

[3.2.1. Sprint Burndown](#_c0j0seq9r1e5) 50

[3.2.2. Release Burndown](#_9wr43svlqzbw) 51

[3.2.3. Velocity chart](#_tdmjma2urelv) 52

[3.2.4. Effort chart](#_u0n75oxaaew) 53

[**4. Metodologia**](#_7xs8m5g4iwq) **54**

[4.1. Visió general](#_helrsp46qtqd) 54

[4.2. Gestió del projecte](#_gemsxi2jg73q) 55

[4.3. Gestió de versions](#_v826808c0dbc) 56

[4.4. Comunicació equip](#_wmna8oxsbbhw) 58

[4.5. Gestió qualitat](#_cz4kiacszyln) 60

[4.6. Gestió de proves](#_gju8rbl5nr8g) 60

[4.7. Gestió configuracions](#_uqzvqqmrfdo4) 61

[4.8. Integració amb altres grups](#_d00yh67p8xss) 62

[4.9. Gestió de bugs](#_h5xnena1qtw2) 63

[4.10. IDEs](#_8b8ucmtqslf6) 64

[4.11. Llenguatges](#_w2q204gcyufl) 65

[4.12. Gestió base de dades](#_mymzkp6mq33h) 67

[**5. Descripció tècnica**](#_cdmxgeylv1j5) **68**

[5.1. Arquitectura](#_5yljauks3cjs) 68

[5.1.1 Diagrama arquitectura física](#_qnmpicmwpse9) 68

[5.1.2 Diagrama Domini Frontend](#_sdw36ho6dy49) 75

[5.1.3 Diagrama Domini Backend](#_annqglh0ew5u) 76

[5.2. Diagrama de components](#_zax39x2ueik0) 76

[5.3. Patrons utilitzats](#_a1mpk4krfcwu) 77

[5.4. Data Models](#_vq3w4dr9o3u9) 82

[5.4.1 Diagrama de classes UML del problema](#_yri6pf2n01bs) 82

[5.4.1 Diagrama de Base de dades](#_niyz472b5hqb) 86

[5.5. APIs](#_tno00ul7au5n) 87

[5.5.1. API pròpia](#_u81axi862f1j) 87

[5.5.2. Nombre d’APIs externes](#_pzme2zsxjl7w) 96

[**6. Referències**](#_45q8ojmxy92g) **98**

# 

# 

# Introduction

## Resum per sprint

#### Sprint 1

L'objectiu principal d'aquest primer sprint és crear totes les eines per començar a programar i desenvolupar les funcionalitats principals relacionades amb el mapa per tal localitzar, afegir i editar els carregadors elèctrics, tant privats com públics. També dedicarem part del sprint al desenvolupament de gestió de reserves i el traçat de rutes.

Amb aquest primer sprint, esperem aprendre i millorar en les tecnologies amb les quals desenvoluparem, agafant agilitat de cara als pròxims sprints.

Per tal d'anotar totes les funcionalitats hem creat èpiques, històries d'usuari i per cada una d'aquestes en surten múltiples tasques. Cada història d'usuari té una puntuació que representa la càrrega de treball que suposa. Les tasques s'aniran assignant dinàmicament al llarg de l'sprint, perquè cadascú pugui adaptar-se als seus temps.

#### Sprint 2

L'objectiu principal d'aquest segon sprint és primer de tot fer un refactor de la feina feta durant el primer sprint, és a dir, unificar el codi fet utilitzant estàndards i patrons comuns. També tenim pendent alguna tasca que no ens ha donat temps a fer-la al primer sprint.

Els primers dies ens centrarem en la configuració i realització del servidor. Començarem per crear el CI/CD que gestionarem amb el Github Actions i definirem el Cloudformation de AWS per poder tenir un servidor on poder penjar el projecte, una base de dades remota (RDS) per a cada entorn (desenvolupament i producció), i un lloc d'emmagatzematge (S3) per guardar les imatges que es pengen a l'aplicació.

A més a més, implementarem totes les tasques relacionades amb la gestió de bicicletes, la qual no hauria de portar molta feina, ja que s'ha de fer una feina molt semblant a la ja feta amb els carregadors. També dedicarem temps a fer la gestió d'usuaris al sistema, és a dir, el sistema d'inici de sessió, de registre i la part del perfil. Finalment, també durem a terme les tasques relacionades amb la gestió de reserves.

## Sprint master report

### Sprint master report Inception - Dani i Andreu

Per a deixar com a finalitzada la fase d'incepció i per a iniciar ja la fase de desenvolupament amb el primer sprint, els Scrum masters fem aquest report per a fer constar la feina feta des de l'últim deliverable.

Com a responsables d'aquesta fase, la nostra finalitat ha sigut organitzar l'equip, crear i distribuir les tasques a cadascun dels membres de l'equip de forma equitativa, organitzar reunions a través de Discord per a tal d'organitzar-nos i fer tasques conjuntament, entre d'altres.

Vam començar a través de l'eina Taiga a crear totes les tasques que calia fer per a tancar la fase d'incepció, les quals es troben totes documentades a continuació, i tenir-ho tot clar per començar a programar. Seguidament, les vam assignar als membres del grup igualant més o menys la càrrega de treball per a cadascuna de les tasques. Vam decidir, però, què algunes de les tasques, tot i que estan assignades a un parell o tres de persones, que caldria revisar-les tots 7 a la vegada a la reunió weekly que fem cada dissabte (com és el cas del diagrama de classes i les èpiques i històries d'usuari), ja que vam decidir que era crucial que tots els membres del grup les tinguessin present.

Una vegada aclarit qui estava al càrrec de totes les tasques a realitzar, tots hem anat fent la nostra feina demanant ajuda als companys si és que calia. Un cop el membre o membres del grup consideren que la tasca està acabada i la qualifiquen de Ready for test, encoratgem a tot el grup a revisar i donar feedback a la feina que han fet els altres.

Les tasques clau per a tenir clar el funcionament del nostre sistema ha sigut portar a cap l'UML i escriure les històries d'usuari classificades en èpiques. En aquestes, els Scrum masters hem organitzat reunions amb tot el grup per tal de repassar a poc a poc tots els aspectes per tal d'entendre bé l'arquitectura de l'App i per tenir-ho ben present quan comencem a programar.

El punt final d'aquesta fase d'incepció ha sigut la presentació del producte al client, el qual els responsables d'incepció hem organitzat una altra reunió amb tots els membres per tal d'aclarir com transmetre la idea de la manera més entenedora possible, deixant anar també una mica alguns dels aspectes tècnics. A partir d'això hem fet entre tots el Power Point també.

Com a conclusió, cal comentar que, pel que fa als contratemps, no hem tingut massa problemes en aquesta fase, ja que tothom ha complert la seva part correctament i a temps. Prediem com a grup que segurament en tindrem bastants ara que comencem els sprints.

### Sprint master report Sprint 1 - Arnau

Per a deixar com a finalitzat el primer sprint i per a iniciar ja el següent sprint, l'Scrum master fa aquest report per a fer constar la feina feta des de l'últim deliverable.

Com a responsable d'aquesta fase, la meva feina ha sigut, organitzar la selecció les històries d'usuari i la divisió en tasques d'usuari, que s'hauran de fer durant l'sprint. També com a sprint master tinc la responsabilitat d'organitzar reunions que s'han anat realitzant periòdicament durant l'sprint mitjançant el Discord. Aquestes reunions començaran sempre amb cada un dels membres de l'equip contestant breument a tres preguntes: què has fet des de l'última reunió? Què tens planejat fer a continuació? Tens algun conflicte o dificultat? També hem utilitzat el Discord com a principal eina de comunicació.

Primer de tot abans de començar l'sprint vam celebrar la reunió del sprint planning, una reunió on es defineix els objectius de l'sprint amb tot l'equip, per a planificar l'sprint vam usar l'eina del Taiga. Per planificar-lo, primer es van decidir les històries d'usuari que es durien a terme durant l'esprint i es van passar del backlog al tauler del primer sprint. Seguidament, entre tots vam definir les diferents tasques per a cada història d'usuari que creiem adients per a poder repartir la feina amb més facilitat. Un cop definides les tasques vam anar assignant-les als diferents membres de l'equip tenint en compte la càrrega de treball i les preferències de cada un dels membres per a poder obtenir d'aquesta manera el millor resultat possible.

En finalitzar la reunió, vaig consultar tota la documentació que s'havia d'entregar al final de l'sprint, per a poder preparar una plantilla del document a entregar, d'aquesta manera aconseguim tenir un esquema de document en el qual tots podem anar afegint i modificant la documentació de forma ordenada. Per mantenir un major control sobre els apartats que encara no s'havien redactat durant la fase d'incepcio vaig afegir-los al Taiga com a storyless tasks.

Com a grup vam prendre la decisió que la següent setmana es dedicaria realitza formació en els frameworks de Flutter i Django, ja que la gran majoria del nostre equip no havia treballat mai amb ells i no coneixia la seva forma de treballar.

Un cop acabada la setmana de formació vam dedicar la weekly, reunions periòdiques que duem a terme els dissabtes al matí, a fer un set-up de l'enviroment per tal d'assegurar-nos de què tots tinguéssim un entorn preparat per a treballar de forma correcta.

A partir d'aquí ens vam posar a treballar seguint la metodologia especificada en aquest document més endavant.

Finalment, es va prendre la decisió que dos dies abans de l'entrega, es deixarien de desenvolupar noves funcionalitats per tal de poder dur a terme tests manuals d'integració deixant una versió funcional de l'aplicació. Durant aquests dos dies també es revisarà tota la documentació per deixar-la preparada per a l'entrega.

### Sprint master report Sprint 2 - Cristina

Per a deixar com a finalitzat el segon sprint i per a iniciar ja el tercer i últim sprint, l'Scrum master fa aquest report per a fer constar la feina feta des de l'últim *deliverable*.

La meva principal feina ha estat organitzar a l'equip, m'he encarregat de cada setmana fer una o dues reunions tots junts per tal de veure per on anàvem i què faltava, d'aquesta manera tots dedicàvem uns minuts a poder expressar si estàvem encallats en alguna cosa o a ajudar a solucionar algun error d'algun company.

També he intentat d'anar recordant sovint als companys que imputin les hores que van treballant perquè no s'imputin 50 hores de cop cada una setmana, ja que sinó després no ens recordem d'en què hem estat treballant i les hores imputades per tasques no són realistes. Vaig proposar que no triguéssim més de dos dies en imputar les hores treballades i més o menys tots ho hem complert.

Abans de començar a posar-nos en matèria vam fer el sprint planning. Vaig deixar preparat el Taiga amb les tasques que havien quedat pendents del primer sprint i les històries d'usuari que corresponien al nou sprint. Entre tots vam definir les tasques que conformarien cada història d'usuari, a continuació ens vam repartir entre backend i frontend i vist que havia més tasques de front que de back vaig suggerir que a frontend hagués mes gent. Finalment, vam acabar dues persones a backend i cinc a frontend. Vam parlar, i entre tots vam decidir que el millor que podíem fer abans de posar-nos amb les noves tasques, era que féssim un refactor del codi i de l'estructura del projecte, ja que al primer érem la majoria molt inexperts i hi havia coses que amb els coneixements que teníem llavors es podien fer millor i ajudarien a fer més fàcil la feina nova a fer.

Un cop cadascú ha anat acabant el refactoring ha anat agafant tasques que no estiguessin assignades a ningú al Taiga.

En general tot l'equip ha funcionat molt bé, ens hem sabut coordinar i comentar les coses que no funcionaven correctament. He intentat de promoure que tothom se sentís lliure de preguntar qualsevol cosa en qualsevol moment i que si algú feia més de dues hores lluitant la mateixa cosa, aixequés la mà.

Sí que és veritat que hem tingut un contratemps i és que un company que no ha pogut acabat les seves tasques i això ha comportat que només s’hagi complert un 57% del que teníem programat al Taiga. Això ha estat degut a que tenia molt punts assignats i ningú ens havíem adonat fins al final de l’sprint. D’aquest problema podem extreure que pel pròxim sprint tots ens hem d’assignar poques tasques i a mesura que anem acabant-les ens anem agafant més.

## Què ha fet cadascú

### Isslam Benali

#### Sprint 1

En aquest primer sprint m’he centrat sobretot en el *Frontend*, concretament en el tractament del perfil de l’usuari. Primer de tot m’he enfocat en implementar la barra de cerca de l’aplicació situada a l’*AppBar* de la *home screen*. Sobre aquesta, li he implementat un *drawer* que conté el conjunt de funcionalitats que un usuari en el seu perfil pot veure i on els meus companys podran navegar a les seves *screens* a través d’aquesta. Finalment, he aprofundit més sobre la *screen* del perfil de l’usuari. Sobre *Backend* m’hauria agradat realitzar funcionalitats, no obstant em tocarà fer-ho en el següent sprint.

#### Sprint 2

Quan va començar el sprint, vam decidir fer la divisió entre programadors de frontend i backend i em va tocar ser dels de frontend.

Durant aquest sprint, al principi ens vam basar a fer un refactoring de les nostres classes tant de backend com de frontend que havíem programat al primer sprint. I tot després ens vam assignar les històries corresponents que ens toquen en aquest sprint.

Primer de tot, vaig fer un refactor en les meves pròpies branques. Llavors vaig anar creant més entrades del Drawer de l'aplicació. A mig sprint a cadascú del frontend ens vam assignar uns widgets que fer i elaborar, per tal de no repetir codi i em vaig ficar a realitzar-los. Vaig dur a terme el widget de la icona d'usuari, el botó verd de "reservar" o "sign up" de l'aplicació, el widget d'usuari + valoració entre altres.

Finalment, em vaig ficar mà a l'obra a realitzar primer les vistes de cada screen:

* Registre Perfil : Sobre aquesta screen és un formulari els camps típics d'email, nom. Cognom, contrasenya etc.
* Veure Perfil : Aquesta screen mostra l'usuari i l'about, però falten les publicacions de l'usuari.
* Editar Perfil : Aquesta screen els camps modificables per l'usuari i el widget de canvi d'Image que accedeix els recursos del mòbil.
* LogOut : És un botó en el menú del drawer que s'encarrega d'eliminar l'api key de l'usuari i desloguear-lo.
* Recuperar contrasenya i pantalla d'enviar email: Són dues pantalles que la primera, has de posar el teu username, i aquest username i a l'email d'aquest usuari se li un codi que ha de posar per poder anar a la pantalla de canviar contrasenya. Una vegada l'ha introduït correctament aquest usuari podra cambiar la contrasenya escrivint-la.
* Serializer usuari : És el model del frontend d'usuari, que tradueix el JSON de les crides de get d'usuari del backend.
* Service d'usuari: Són les crides al backend de cada funcionalitat i vista que he mencionat anteriorment.
* Service de contraseña: Són les crides al backend d'enviar l'email, i canviar la contrasenya.

Personalment, m'hauria agradat acabar totes les tasques que tenia adjudicades, però no he pogut acabar la de veure perfil amb les publicacions de l'usuari ni de poder mostrar la searchBar en la pantalla principal, després de molts intents intentar crear una que quedes bé en l'aplicació i no tingués problemes.

### Arnau Giménez

#### Sprint 1

Durant aquest primer sprint la meva focalització ha sigut sobre la creació del formulari per afegir punts de càrrega privats. A part, també m’he encarregat de tot el *backend* relacionat amb les publicacions de carregadors privats.

Per a la creació del formulari he hagut de gestionar la necessitat d'afegir imatges als punts de càrrega, per gestionar aquest requisit vaig crear un component reutilitzable que obté imatges de la memòria del telèfon o de la càmera i les retorna.

Aprofundint en l'apartat de *backend* m’he encarregat de crear els models inicials de la base de dades per a gestionar el xat, que s’implementarà més endavant. També he implementat les funcionalitats *POST* per a crear un nou punt de càrrega privat, les funcions *GET* per a obtenir els carregadors privats, per obtenir un carregador privat en concret i per obtenir els *type\_speeds*, *charger\_types* i *current\_types* necessaris per al formulari de creació, les funcions *delete* per a eliminar un carregador en concret i la funció *PUT* per a editar la informació d'un carregador en concret. Finalment, en el *backend* vaig implementar tots els serializers referents a les publicacions, carregadors, localitzacions..., excepte el de carregadors públics

#### Sprint 2

Durant aquest sprint m'he centrat en la creació de formularis per a editar carregadors privats i bicis i el formulari de creació de bicis. Durant aquest sprint també he sigut responsable d'aprovar les pull request que es feien de frontend juntament amb l'Andreu

Per a poder crear tots els formularis de forma eficient he fet un refactor del formulari

punt de càrrega privat. En aquest refactor he separat les diferents pàgines del formulari en components per a poder-los reutilitzar i fer el codi més llegible, també he canviat la pàgina de selecció d'ubicació per a poder seleccionar la localització de la publicació des del mapa proporcionat per l'api de Google o des d'una barra de cerca que agafa les direccions també de l'api de Google. Finalment, he centralitzat les crides al backend en una classe per a utilitzar-les de forma genèrica en altres funcionalitats.

Un cop fet el refactor he implementat la pàgina d'un punt de càrrega, i les de creació i modificació de bicicleta usant els components creats durant l'etapa de refactor.

Un cop el bakend tenía un primer sistema de pujada d'imatges vaig adaptar el frontend amb noves multiform request per tal de poder enviar les imatges, i juntament amb la Cristina vam debugar el codi per assegurar que tot funciones bé.

Finalment, també vaig crear un widget per a mostrar múltiples imatges genèriques i vaig crear una plantilla per a un nou sistema de rutes per poder anar afegint les redireccions al document mentre anàvem avançant.

### Miguel Gutiérrez

#### Sprint 1

La meva tasca principal ha estat desenvolupar la visualització de la informació dels diferents carregadors, fins ara els públics, de la nostra aplicació. Dintre d'aquesta trobem dues pantalles i l'obtenció de les dades dels marcadors. A continuació detallo cada punt:

La funcionalitat principal de la pantalla 1/3 és mostrar la informació del carregador públic clicat per l'usuari. Per dur a terme la tasca, inicialment va ser necessària la cerca d'informació tant del llenguatge de *Flutter*, com la funcionalitat en qüestió. Un cop obtinguda la corresponent informació vaig realitzar una pantalla de proves on es visualitzava el mapa de Google Maps i es pintaven dos carregadors.

Seguidament, vaig desenvolupar el corresponent *widget* per poder mostrar la informació d'un punt. Per poder continuar va ser necessària la integració del *widget* a la pantalla del mapa. A continuació vaig aprofitar la implementació del *backend* dels carregadors per obtenir les dades. Un cop obtingut tot el material necessari faltava ajuntar-lo. Aquí és on es va implementar la funcionalitat del clic per mostrar el *widget* i l'ús de les dades obtingudes del *Back* per mostrar la informació corresponent al carregador premut.

La principal funcionalitat de la pantalla *listview* és mostrar la informació dels carregadors públics en format llista per tal de tenir una perspectiva alternativa al mapa. Per desenvolupar la tasca, de nou es va requerir cercar informació per tal de familiaritzar-se amb el llenguatge i tractar la funcionalitat en qüestió.

Un cop obtingut els coneixements vaig procedir a reutilitzar el *widget* de la pantalla anterior i crear la disposició d'aquest en format llista per aquesta pantalla. Es va utilitzar un joc de proves inicial per a poder visualitzar-ho. Ja funcionant el joc de proves, vaig procedir a fer la crida al *backend* per obtenir les dades dels carregadors i vaig integrar-les als *widgets* informatius. Finalment, es va fer un formateig de la pantalla per seguir l'estàndard proposat pel grup.

També vaig realitzar la *migration* de la classe *Report* i els seus corresponents un cop fet l'anàlisi del diagrama desenvolupat a l'*inception*.

Com a resum del sprint puc dir que ha estat dedicat, majoritàriament, a la recerca d'informació i funcionalitats de *Flutter*. En moltes ocasions havia desenvolupat unes funcionalitats d'una forma i s'han hagut de fer *refactorings* per al reaprofitament i millores en l'aparença. En el meu cas tenia assignada una altra història que per aquest sprint no he arribat a realitzat. He preferit tenir funcionat la primera assignada en comptes de tenir una barreja entre les històries i no tancar cap.

#### Sprint 2

En aquest sprint m'he dedicat a fer la refactorització d'una part dels *widgets*, ja que hi havia temes funcionals que en el primer no estaven tractats correctament.

He tingut dificultats en mantenir-me al dia a dia amb el projecte perquè la primera setmana, després del primer sprint, tenia altres matèries pendents i entregues a lliurar. Això va donar peu a atorgar una prioritat menys altra la resta, provocant una dedicació i inferior respecte el primer sprint.

Un cop en marxa, vam distribuir-nos diferents widgets per tal d'evitar solapaments i repetició de codi, el qual ha funcionat adientment. Respecte a les meves tasques he seguit treballant en les pantalles dels carregadors, proporcionant una funcionalitat adequada, tant pel format llista com card. En canviar del format *json* als serializers hem hagut de fer focus a què estiguessin preparats per tal de poder obtenir la infomació del backend i mostrar-la al frontend. Això també ha suposat haver de fer un refactor més profund per tal d'assolir la funcionalitat desitjada.

Dintre d’aquest refactor he realitzat l’actualització de les *cards* de la pantalla principal per tal de tenir un slider que mostri la informació corresponent, tant pels carregadors com per les bicicletes. També he incorporat el tema de la pantalla infinita en la llista dels carregadors i bicicletes per tal d’anar mostrant carregadors/bicicletes i no carregar-los tots alhora. Entre d’altres s’han creat diversos widgets necessaris per a les pantalles i en la integració dels elements per tal de poder reutilitzar-los en la mesura del possible.

Finalment, he anat just en temps per a l'entrega. Espero per la pròxima poder rectificar i intentar-ho portar tot més al dia per tal d’evitar colls d’ampolla.

### Cristina Migó

#### Sprint 1

En el primer sprint m’he focalitzat principalment amb la part de *Backend*, concretament amb els carregadors i amb la classe carregadors públics, que hereta de carregadors.

He treballat amb l’*API* que la Generalitat ofereix respecte als punts de càrrega elèctrics i he fet una crida per tal de guardar tot el que retorna al nostre model de *chargers*, tenint així una còpia en local dels punts. Per tal de fer això he hagut de modificar i reestructurar els models que es necessiten per crear una instància de carregador que teníem inicialment definits. He fet un sistema de sincronització entre l’*API* i la còpia local, per tal de fer crides a l*’API* només cada un cert temps i no per a cada petició, ja que fer la crida a l’API i guardar totes les dades als nostres models triga uns 10 segons, i no ens podem permetre que trigui aquest temps cada cop que un usuari selecciona algun botó que comporti mostrar punts de càrrega.

A part m’he encarregat d’habilitar la funcionalitat de filtrar els carregadors, ja siguin públics o privats. D’aquesta manera depenent dels filtres que se seleccionin a la pàgina principal el nostre sistema retornarà aquells punts que coincideixen amb els filtres.

#### Sprint 2

En aquest segon sprint he dedicat la major part de les hores a la configuració del servidor i d’aquells serveis d’AWS que necessitem per pujar l’applicació i la definició del workflow de Github Actions.

Vaig començar per definir el workflow que es pot generar amb Github Actions, allà vaig definir tres *jobs* diferents, un que executa tests, l’altre que buildeja la imatge de docker definida al Dockerfile i la penja a una conta de DockerHub que vaig haver de fer-me i un últim que configura les credencials de la conta de Amazon que també em vaig crear prèviament i fa deploy d’una stack a AWS CloudFormation. Amb això funcionant vaig poder posar-me a definir la plantilla.

Vaig començar pel Bucket S3 i la seva política de contenidor, seguit de la EC2 (Elastic Cloud Compute) i de la RDS (Relational Database Server). Vaig trigar bastant de temps sobretot per poder crear la RDS i la EC2 ja que necessiten de una VPC on crear-se i pertànyer a un grup de seguretat, el qual té definit unes regles d’entrada i sortida. Finalment vaig acabar per utilitzar tant la VPC i el security group que venen per defecte i va funcionar.

També m’he encarregat de fer les funcions del backend que penjen les imatges que es passen des del *frontend* al Bucket de S3 i de realitzar la creació d’una publicació de bicicleta.

L’última tasca que també he realitzat i m’ha portat bastant de temps ha estat configurar el fitxer de docker-compose que se li passa a la EC2 i que s’executa en quan ens connectem a la instància. El més complicat ha estat que connecti correctament amb la instància i un cop connectada es pugui accedir a la IP on està penjada l’aplicació.

Bàsicament durant el segon sprint m’he dedicat a fer possible que l’aplicació es pugui penjar a un servidor i que ara al tercer sprint el grup que agafarà dades nostres pugui fer-ho.

### Daniel Oliveras

#### Sprint 1

En aquest primer sprint m'he centrat en la secció de perfil. He fet el *frontend* de tota la pantalla. Com que encara falten coses per desenvolupar per a la gestió d'usuaris, les primeres seccions del perfil són estàtiques i no connecten amb la base de dades (actualment estan comentades i no es renderitzen). En canvi, la secció de “Els meus punts”, de la tasca 82, sí que està connectada amb el backend i rep i mostra les dades. En aquesta secció, alguns dels paràmetres, com la distància, encara no es calculen i es passen directament des del *frontend*, és per això que la tasca queda pendent d'acabar per al següent sprint.

Per a les targetes del *frontend*, que representen els meus punts, vaig fer que s'adaptessin afegint, traient o modificant camps segons les dades i el tipus de publicació que rebien des del backend.

Pel que fa al *backend*, vaig començar per fer els models de les bicicletes per poder emmagatzemar aquesta mena de publicacions. Després vaig passar a fer els *serializers* per transformar les dades a *JSON*.

També es creen les diferents vistes que em permetien fer les peticions per obtenir les dades tant de les bicicletes com bicicletes elèctriques. I alguns *útils* per agafar instàncies de la base de dades més fàcilment. Finalment vaig crear les rutes necessàries per poder fer la crida des de *frontend*.

#### Sprint 2

La meva tasca principal, durant aquest esprint, ha estat desenvolupar *widgets* que permetin gestionar les reserves.

En un inici, vam pensar en fer servir el calendari per defecte de flutter i el seu selector d'hores, per seleccionar l'inici i la fi de les reserves. Però ho vaig haver de descartar, ja que necessitàvem impedir a l'usuari, visualment, que reservés hores no disponibles.

Vaig trobar una llibreria que permet diferenciar les hores disponibles de les ocupades al selector d'hores, però no era compatible amb el nostre projecte, així que vaig haver de buscar alternatives.

El següent pas va ser cercar alguna llibreria de flutter per gestionar reserves i així facilitar-nos el desenvolupament. Però després de provar amb diferents, no en vaig trobar cap que ens donés la flexibilitat necessària. Així que vaig acabar fent-ho a mà.

Les hores disponibles, va decidir mostrar-les en una *grid* amb estils diferents per indicar-ne la disponibilitat, de manera similar a com es mostren les reserves de la biblioteca de la FIB.

Per seleccionar la data, vaig descarregar una llibreria amb un calendari que va oferia molta personalització i el vaig adequar perquè encaixés amb l'estètica del projecte.

Finalment, vaig crear un widget que engloba i comunica al calendari amb el llistat d'hores disponibles. D'aquesta manera, l'usuari pot seleccionar una data, i es mostraran les hores disponibles i ocupades d'aquell dia. Per fer la reserva l'usuari només ha de marcar les hores disponibles que voleu reservar i el widget crearà tantes reserves com períodes d'hores contigües hagi seleccionat.

### Àlex Ollé

#### Sprint 1

Al ser el que tenia més experiència, tant en tecnologies *backend* (concretament en *Django*) com en aplicacions mòbils (sobretot *Android*), el meu càrrec principal ha estat la transmissió de coneixements i la generació dels projectes. Concretament, en la primera setmana del sprint, vaig crear tant el projecte de *frontend* com el projecte de *backend*, instal·lant tot allò necessari per tal de poder treballar de manera còmode, vaig crear les connexions entre base de dades i *backend* i vaig assegurar-me de que tot funcionés correctament. A continuació vaig documentar, per a cada repositori, el procés d’instal·lació pas a pas, per tal de que els meus companys ho poguessin instal·lar de manera independent. Tot i així, vam aprofitar dos reunions per tal de fer les instal·lacions i, en cas de dubte, m’anaven preguntant. Finalment, també vaig fer apunts sobre les funcionalitats principals de cada tecnologia i com calia utilitzar-les, fent que cada cop que tinguin algun dubte, puguin mirar els apunts i en la majoria de cops, no els calgui buscar per Internet o preguntar-me.

Un cop el projecte ja estava en marxa, em vaig posar a fer la meva part, la qual va començar amb la realització del sistema de rutes i la internacionalització de l’aplicació mòbil. La internacionalització o localització la vaig haver de fer dos cops, ja que no vaig tenir en compte totes les necessitats que havia de tenir i, un cop implementat, vaig veure que no ens feia servei tal i com estava plantejat, així que es va replantejar i decidir utilitzar una altra llibreria per tal de fer la traducció del lloc, que és la utilitzada actualment. Un cop fet el sistema d’internacionalització, vaig fer el selector d’idiomes per a permetre canviar l’idioma i poder fer proves amb els diferents idiomes.

Seguidament, vaig fer una classe al *frontend* la qual generalitzava i unificava les crides al *backend* en un sol fitxer, fent que fos molt més fàcil fer crides al *backend* i requerís de molt menys esforç. També en relació amb el fet de facilitar la feina als meus companys, vaig fer una sèrie de classes de *útils* les quals permetien obtenir dades de la *API* de Google Maps de manera fàcil i sense haver-se de saber l’*API* de i com fer-li consultes.

Per últim, com a funcionalitats, he fet el sistema de rutes, el qual consisteix en dibuixar rutes sobre un mapa i obtenir totes les dades de la ruta, com el recorregut, temps, distàncies, etc. Al sprint 1 s’ha fet la primera iteració d’aquesta pantalla, i de cara als següents ens agradaria seguir ampliant-la i millorant la seva funcionalitat.

A part de les tasques més relacionades amb coses productives de cara a l’aplicació, durant l’sprint he anat donant suport als meus companys i companyes, resolent dubtes, ajudant a tirar endavant algun fragment de codi, solucionant problemes, entre d’altres.

Com a resum de l’sprint, puc dir que vaig començar dedicant bastant temps a la pràctica, ja que havia de tenir com abans millor tot el projecte inicial per a que els meus companys s’hi poguessin posar. Un cop ja estava tot estable i tots podien treballar de manera individual, em vaig dedicar a altres assignatures que havia deixat una mica enrere per acabar la creació dels projectes. Un cop ja ho tenia tot estable, em vaig posar a implementar la resta de tasques que tenia assignades, podent-les fer totes i millorant alguns fragments dels meus companys.

#### Sprint 2

En aquest segon sprint, m’he centrat en les tasques relacionades amb el backend, ja que tal i com s’ha comentat, en aquest segon sprint ens hem repartit les dues plataformes fent que, uns s’encarreguin del frontend i altres del backend.

Les primeres dues setmanes vaig estar menys actiu degut a que va haver-hi la setmana d'exàmens i vaig estar fora uns dies, i per tant, per a mi l’sprint va començar gairebé a la tercera setmana. Des de llavors, em vaig posar les piles i vaig començar a implementar les tasques que em vaig assignar, que eren la gran majoria de les relacionades amb el backend.

Dit i fet, vaig començar fent el refàctoring de tot el codi de backend i reestructurant les taules de la base de dades. També reorganitzant les diferents funcions i eliminant codis menys reaprofitables o afegint comprovacions. Aprofitant que ja tenia coneixements previs de Django, vaig poder-ne exprimir més les seves capacitats. En aquest pas, també vaig aprofitar per a dividir l’estructura del codi de l’aplicació en 3 capes, on originalment era de 2, afegint així la capa de serveis per a cada app.

Al veure que estava costant comunicar-se amb el frontend i posar-nos d’acord amb els endpoints i les necessitats (degut als canvis realitzats en el backend, que afectaven l’obtenció de dades de l’aplicació mòbil), vaig decidir redactar un full de càlcul en el qual estarien dibuixats tots els endpoints i especificant quins requisits tenien i quines respostes donaven.

Un cop vaig tenir tot el backend refactoritzat, em vaig posar a implementar les diferents tasques que tenia per aquest sprint, però ja només em quedava poc més d’una setmana, així que la meva dedicació va haver d’augmentar per tal de poder arribar a complir el meu objectiu.

Durant aquesta última setmana, he fet tots els endpoints de bicicletes, així com els relacionats amb l’usuari i l’accés a l’aplicació, els relacionats amb bookings i amb les ocupacions.

També he realitzat la pantalla de logueig de l’aplicació mòbil, ja que sabia com podia implementar l’accés en el frontend i per això ja vam decidir que ho implementés.

Els últims dies m’he dedicat a resoldre les incidències que han anat sortint al intentar utilitzar els recursos de l’api des de l’aplicació i on podien sorgir alguns conflictes o dificultats, que s’han anat resolent.

Per últim, he estat donant suport als meus companys, tant de frontend com de backend ajudant a resoldre els diferents problemes que els anaven sorgint, ja fos relacionat amb el backend, amb la part visual o amb el servidor.

### Andreu Orensanz

#### Sprint 1

En aquest primer sprint he realitzat, en primer lloc, totes les tasques relacionades en col·locar els *markers* al mapa que localitzen la posició dels carregadors elèctrics públics i privats. En aquestes m’he dedicat a fer principalment totes les tasques relacionades amb el *frontend*: col·locar el mapa, posar un botó per a centrar la vista a la posició actual de l’usuari i agafar els punts de càrrega del *backend* i localitzar-los al mapa.

Les altres tasques que m’han tocat fer són les relacionades amb mostrar, editar i cancel·lar reserves. Primer vaig començar pel *backend* per tal d’obtenir les reserves que havia fet l’usuari (en aquest sprint han sigut *hardcodejades* per un usuari predeterminat) amb tots els atributs corresponents. Com que aquestes reserves pertanyen a carregadors que tenen una valoració, he decidit muntar la classe *Rating* per tal que es poguéssin crear valoracions de reserves determinades. Aquestes de moment mostren una valoració mitjana d’un carregador per així poder mostrar des del *frontend* el que es voldrà posar per pantalla finalment (en futurs sprints *Rating* es reformatejarà). Seguidament, vaig dissenyar la pàgina on es mostren totes les reserves que té l’usuari. Aquestes es recullen des del *backend* mostrant determinats atributs i he implementat un botó per tal de cancel·lar la reserva.

Com a conclusió, puc dir que aquest primer sprint ha sigut intens perquè hi ha hagut molt temps dedicat a l’aprenentatge de llenguatges de programació i *frameworks* que no havia tocat mai encara. Espero que el pròxim sprintem pugui focalitzar principalment en fer les tasques sense tenir tants problemes. Gràcies a l’ajuda dels meus companys he après molt i he pogut realitzar les tasques tal com es demanaven a les històries.

**Sprint 2**

En aquest segon sprint, he dedicat bona part del temps a crear *widgets* i a crear i configurar els *serializers* al *frontend*. A diferència del sprint anterior, on s’agafen les dades a partir de crides al *backend,* que retornaven un codi *JSON*, i aquest és parsejava manualment tal i com ens convenía, ara es passava el *JSON* a l’objecte serialitzat corresponent. D’aquesta manera ens seria molt més fàcil accedir a les dades adients i ens seria més còmode per programar en futurs sprints.

A mida que anava arreglant els *serializers* anava fent les tasques puntuables representades al Taiga, que en aquest sprint estaven relacionades en mostrar i reservar bicicletes. En primer lloc, vaig dedicar-me a mostrar les bicicletes al mapa i quan aquestes eren seleccionades per l’usuari es mostrava una targeta lliscant amb la informació corresponent. A partir de la barra inferior, vaig fer que si es seleccionava la icona del cotxe, es mostressin només carregadors, i si es feia clic a la icona de la bicicleta, només publicacions de bicicletes. A diferència del sprint anterior, en aquest he anat realitzant tot el que s’ha programat al *frontend* a partir de les dades del *backend*, per tal de provar que la connexió era correcta en tot moment i no haver de *hardcodejar* tantes dades com abans, fet que ens va portar problemes més endavant. He treballat juntament amb l’equip de *backend* per a demanar els *endpoints* que necessitava per tal de tenir les dades que tocaven en el format corresponent.

Un altre apartat que vaig haver de corregir per complet, per tal d’adaptar-ho amb les dades dels objectes serialitzats, és l’opció de veure les reserves que ha fet l’usuari, ja siguin de bicicletes o de carregadors, i aquestes poder-les cancel·lar i filtrar per data i tipus.

El que m’ha anat molt bé ha sigut programar juntament amb altres companys, fent una mica de *pair programming* en alguns casos, ja que alguns dels *widgets* que havia desenvolupat un membre havien d’encaixar al d’un altre.

Com que he tingut feina de diverses assignatures a la vegada, al principi del sprint no vaig avançar gaire, fet que aquestes dues últimes setmanes han sigut molt més intenses. Però en aquest sprint ja sabia programar molt més amb el *framework* de *Flutter* i tenia molt més coneixement del funcionament de tot el sistema, de manera que el desenvolupament de nous *widgets* ha sigut molt més àgil.

## Avaluació de companys

### Sprint 1

En general, tot ha anat molt bé i no hem tingut problemes entre nosaltres ni a l’hora de treballar, així com tampoc hi ha hagut cap discussió o conflicte durant tot l’sprint.

En quant a càrrega de feina, sí que hi ha hagut membres de l’equip que han tingut més càrrega que altres, sobretot degut a falta de temps, noves etapes, han començat a treballar, càrrega de feina elevada d’altres assignatures, etc. Això ha fet que en el *project record* hi hagi un desequilibri d’hores treballades elevat. No ens preocupa, ja que sabem que als següents sprints, la dedicació s’anirà igualant i les hores dedicades s’haurien d’anar ajustant una mica més, encara que sempre existirà una diferència deguda a la motivació per l’assignatura dels membres, així com del seu temps lliure.

Si parlem de distribució de feina, si que hi ha hagut una gran desviació, sobretot deguda als coneixements previs i la velocitat d’aprenentatge dels membres, fent que alguns de nosaltres hagin necessitat més temps de formació i proves, i altres s’hagin posat més directe. Tot i així, com ara ja tenim tots un nivell acceptable, de cara als següents sprints, aquesta càrrega també s’hauria d’anar ajustant.

Després, en quant al compromís, creiem que tothom s’ha esforçat en estar en totes les reunions que hem anat fent, les quals eren molt importants a l’hora de veure com anavem, resoldre dubtes i planificar les pròximes accions. Era molt important estar per tal de que tothom fos conscient de les coses i no es perjudiqués a ningú en les accions realitzades.

A l’hora de treballar, hem anat realitzant les tasques de manera més individual i rarament hem treballat en parelles o en grup, ja que els horaris de treball de cadascun dels membres és divers i no coincidim uns amb els altres.

En conclusió, estem contents de com ha anat aquest primer sprint, tot i que sempre hi ha coses a millorar i que esperem que es facin millor de cara al segon sprint.

### Sprint 2

En aquest segon sprint, hem continuat amb tot molt semblant al sprint anterior, on no hi ha hagut cap conflicte entre nosaltres ni a l’hora de treballar. Tothom ha assistit a les reunions quan s’han convocat i ha respectat la manera de treballar que vam decidir els primers dies de classe.

En quant a dedicació, tot i que tots i totes hi hem dedicat moltes hores (com també es pot veure al Project record track), sí que hi ha hagut un desbalanceig clar entre els membres, però aquí també hi recau la motivació i disponibilitat dels diferents membres cap al projecte, i per tant, no és algo que haguem de destacar o que ens preocupi.

Si parlem de la distribució d’hores al llarg de l’sprint, sí que hi ha hagut molta irregularitat, ja que durant les primeres dues setmanes no es va acabar res en el projecte, la qual cosa ha fet que hi hagi hagut més pressió a l’última setmana i por a no arribar als objectius marcats al inici de l’sprint. Això ha fet que en alguns moment estiguéssim una mica més tensos entre nosaltres, no culpant o en conflicte, simplement perquè estàvem lluitant per a que sortissin les coses i poder oferir aquelles funcionalitats que requerien els nostres companys.

A l’hora de treballar, en aquest sprint hem treballat molt més en equip, i això ho hem aconseguit gràcies a l’acte involuntari que vam fer de connectar-nos als canals de veu de discord quan ens posàvem al projecte, la qual cosa motivava als companys a unir-se. També hi ha hagut molts moments de pair programming, on ens posàvem per parelles per a treballar, el que ha fet que el grup millorés molt. Tot i això, al treballar en hores molt diferents (alguns pel matí, altres per la tarda i altres per la nit), ha fet que en aquestes sessions no tothom hi pogués ser.

En conclusió, estem contents de com ha funcionat el treball en equip, encara que la pressió ens ha fet males passades, i sabem que hi ha coses a millorar, tal i com es comenta posteriorment a la retrospectiva.

# 

# Requisits

## Not List

* Les funcionalitats afegides a la Not List estan marcades amb el color verd.
* Les funcionalitats que passen d’estar Unresolved a Out of Scope están marcades de color taronja.
* Les funcionalitats que passen d’estar Unresolved a In Scope estan marcades de color lila.

| **IN SCOPE** | **OUT OF SCOPE** |
| --- | --- |
| Geolocalització | Accidents o assegurances de cotxe. |
| Localitzar punts de càrrega públics i privats. | Sistema de compartir cotxe per a fer una mateixa ruta, compartint els costos. |
| Poder filtrar punts per preu i rellevància (geogràficament, preu, tipus d’endoll, velocitat de càrrega). | Estadístiques de conducció (ej: sobrepassa el límit de velocitat) |
| Guardar dades del cotxe (model, tipus de carregador). | Assegurances de bicicletes i fiances per a robatori. |
| Valoració de punts de càrrega i bicicletes i sistema de trofeus. | Lloguer de patinets, skates, motos, camions, cotxes… |
| Poder reportar problemes en una terminal de càrrega o en una bicicleta. | Sincronització amb calendaris externs (Google) |
| Traçar el camí fins a la ubicació de la bicicleta o el punt de càrrega. | Poder compartir rutes o altres a través de xarxes socials. |
| Sistema de reports i queixes. | Connexió en temps real amb el cotxe, per tal d’obtenir la informació del vehicle |
| Web d’administració i moderació. | Control de velocitat o de males pràctiques |
| Sistema de xat de contacte entre llogater i client. | Traçat de rutes que tinguin més d’un punt de destí (com fa Google Maps) |
| Sistema de publicació d’un punt de càrrega privat | Càlcul del trànsit en temps real a l’hora de fer la ruta |
| Sistema de publicació d’una bicicleta privada | Control de l’estat meteorològic |
| Sistema per sol·licitar un carregador o una bicicleta que està a l’aplicació. | Comprovació de la veracitat i confiança del perfil |
| Sistema de gestió d’usuari (registre, accés, modificar perfil). | IA que aproximi el consum del cotxe a partir de les conduccions de l'usuari |
| Sistema multi idioma (català, castellà, anglès) | Sistema d’indicacions detallades i/o assistent de veu indicant les direccions. |
| Estat de la contaminació de l’aire en una estació de càrrega o en una ruta de bicicleta | Disponible per a iOS |
| Botó que et porti a Google Maps amb la ruta seleccionada (de A a B). | Funcionament fora de Catalunya |
| Guarda l'inici de sessió (quan es tanqui l'explicació no s’ha de tornar a iniciar sessió) | Fer visibles els tallers de cotxe depenent de la marca i model del vehicle |
|  | Mostrar la contaminació ambiental que evites conduint un cotxe elèctric o anant en bicicleta |
|  | Monitoritzar el nivell de càrrega que té el cotxe després de ser carregat, a partir de la velocitat de càrrega i el tipus de cotxe. |
|  | Poder filtrar per carregadors amb punts d’interès a prop (restaurants, lavabos, centre comercial, etc) |
|  | Integració d’un sistema de pagaments dins de l’aplicació. |
|  | A mesura que la valoració d’un usuari millora, la comissió que es cobra baixa. |
|  | Implementació de punts públics de bicicletes a Barcelona (Bicing) |
|  | Dibuixar rutes tenint en compte la càrrega del vehicle, tant inicial com final. |
|  | Poder indicar preferències de conducció (interval entre parades). |
|  | Traçar rutes passant pel lloc de recollida o retorn d’una bicicleta, és a dir, que la ruta pugui continuar després de recollir o deixar la bicicleta. |
| **UNRESOLVED** | |
| Poder seleccionar el tipus de bicicleta de forma detallada (mida, pes, nivell, etc) | |
| Consells de conducció eficient, per tal de no gastar tant, durant les pantalles de càrrega. | |
| Poder guardar com a preferits punts de càrrega o bicicletes. | |
| Sistema de notificacions (quan reps un missatge, quan reps una sol·licitud de lloguer, etc) | |
| Pàgina de FAQs | |

## Backlog

Per definir les diferents èpiques ens hem basat en els conceptes generals del nostre projecte, cada un d’aquests conceptes corresponen a una èpica.

A continuació hi ha una taula dividida entre les diferents èpiques definides i per cada èpica mostrem les històries d’usuari que engloba. També hem afegit una columna amb els punts que li donem a cada història i una altra amb la seva prioritat. Els punts surten de jugar al *planning poker* amb tot l’equip.

| **Èpiques** | **Històries d'usuari** | **Punts** | **Prioritat** | **Estat** |
| --- | --- | --- | --- | --- |
| **Gestió usuaris** | Login natiu | 8 | Mitjana | Feta |
| Registre d’usuari | 13 | Mitjana | Feta |
| Tancar sessió | 2 | Mitjana | Feta |
| Edició perfil | 8 | Mitjana | Feta |
| Veure perfil | 5 | Mitjana | Mitges |
| Login Google | 3 | Baixa | - |
| Login Racó | 2 | Baixa | - |
| Canviar a perfil de llogater | 8 | Alta | Feta |
| Veure trofeus | 13 | Baixa | - |
| Seleccionar idioma | 8 | Alta | Feta |
| **Gestió Carregadors** | Veure carregador concret | 13 | Alta | Feta |
| Afegir carregador | 21 | Alta | Feta |
| Eliminar carregador | 3 | Alta | - |
| Mostrar contaminació en punt | 13 | Mitjana | - |
| Mostrar punts de càrrega públics | 34 | Alta | Feta |
| Edició carregador | 8 | Alta | Feta |
| Obtenir punts de càrrega privats | 21 | Alta | Feta |
| Mostrar punts de càrrega privats | 13 | Alta | Feta |
| **Gestió Bicicletes** | Veure bicicleta concreta | 13 | Mitjana | Feta |
| Afegir bicicleta | 21 | Mitjana | Feta |
| Mostrar contaminació en punt | 8 | Mitjana | - |
| Eliminar bicicleta | 5 | Mitjana | Feta |
| Mostrar punts de bicicleta | 13 | Mitjana | Feta |
| Editar bicicletes | 8 | Mitjana | Feta |
| **Gestió rutes** | Iniciar ruta | 13 | Alta | - |
| Mostrar indicador de bateria per la ruta | 21 | Baixa | - |
| Cancel·lar ruta | 5 | Mitjana | - |
| Mostrar ubicació actual | 8 | Alta | - |
| **Gestió vehicle** | Afegir vehicle | 34 | Mitjana | - |
| Seleccionar vehicle | 5 | Mitjana | - |
| Modificar vehicle | 8 | Mitjana | - |
| Eliminar vehicle | 5 | Mitjana | - |
| **Gestió reports** | Enviar queixa | 8 | Baixa | - |
| Processar queixa | 34 | Baixa | - |
| Eliminar punt de càrrega | 3 | Baixa | - |
| Eliminar usuari | 3 | Baixa | - |
| Eliminar lloguer de bici | 3 | Baixa | - |
| Enviar advertencia | 8 | Baixa | - |
| **Gestió xat** | Enviar missatge | 8 | Baixa | - |
| Rebre missatge | 8 | Baixa | - |
| Consultar xat | 21 | Baixa | - |
| Eliminar xat | 3 | Baixa | - |
| **Gestió reserves** | Reservar un carregador | 21 | Alta | - |
| Reservar una bicicleta | 13 | Mitjana | - |
| Eliminar reserva bicicleta | 5 | Mitjana | - |
| Eliminar reserva carregador | 5 | Alta | - |
| Administració reserves (propietari) | 21 | Mitjana | - |
| Veure les meves reserves | 8 | Alta | Feta |
| Bloquejar franja | 13 | Mitjana | Mitges |
| Calendari reserves (propietari) | 13 | Alta | Feta |
| **Gestió valoracions** | Valorar carregador | 13 | Baixa | - |
| Valorar bicicleta | 8 | Baixa | - |
| Valorar propietari | 8 | Baixa | - |
| Valorar usuari | 5 | Baixa | - |
| **Gestió cerca** | Filtrar Carregadors | 21 | Alta | - |
| Ordenar punts de càrrega | 13 | Alta | - |
| Veure els meus punts (propietari) | 13 | Alta | - |
| Ordenar bicicletes | 8 | Mitjana | - |
| Filtrar bicicletes | 13 | Mitjana | - |

## Requeriments no funcionals

En aquest apartat es descriuen els requisits no funcionals que té el nostre sistema, és a dir, tots aquells que descriuen les característiques de funcionament de la nostra aplicació i, a diferència dels funcionals, no descriuen les funcions a realitzar. Aquests requisits van numerats per un identificador i segueixen la classificació de *Volere*:

### Requisits de capacitat d’ús i humanitat

| **Número** | **1** |
| --- | --- |
| **Tipus de requisit** | 11a. Requisits de facilitat d’ús |
| **Descripció** | Es descriuen les característiques que ha de tenir el sistema per tal que un determinat usuari li sigui fàcil utilitzar. Entre aquestes propietats hi trobem l’eficiència d’ús, la facilitat de recordar i la satisfacció en general a l’usar el producte, entre d’altres. |
| **Justificació del requisit** | Els usuaris els hi ha de ser fàcil d’interactuar amb el nostre sistema. D’aquesta manera la utilitzaran més que els sistemes dels nostres competidors i tindran una experiència agradable fent ús del nostre sistema. |
| **Condició de satisfacció** | * Els usuaris han de poder utilitzar el nostre sistema sense tenir cap tipus de coneixement previ de com es fa servir. És a dir, el sistema ha de ser extremadament intuïtiu. |

| **Número** | **2** |
| --- | --- |
| **Tipus de requisit** | 11b. Requisits de personalització i internacionalització |
| **Descripció** | Especifica la manera com el producte pot ser modificat o configurat per prendre a compte les preferències personals de l'usuari o elecció del llenguatge. Ha de tenir en compte aspectes com el llenguatge, i opcions personals de configuració. |
| **Justificació del requisit** | Fer que tot usuari del nostre sistema pugui entendre-ho tot mentre fa ús de l’aplicació. |
| **Condició de satisfacció** | * L’aplicació contindrà al perfil l’opció de canviar l’idioma de tot el sistema (anglès, català i castellà). |

| **Número** | **3** |
| --- | --- |
| **Tipus de requisit** | 11c. Requisits d’aprenentatge |
| **Descripció** | Aquest requisit especifica com de fàcil ha de ser aprendre a utilitzar un producte |
| **Justificació del requisit** | És important que als usuaris de la nostra aplicació no els hi costi gaire aprendre com funciona aquesta, ha de ser força intuïtiva. Si no és així podrien frustrar-se i anar a buscar una altra plataforma. |
| **Condició de satisfacció** | * Els usuaris han de ser capaços d’utilitzar la nostra aplicació sense la necessitat d’haver fet un curs previ que expliqui com funciona. |

### Requisits de preservació i suport

| **Número** | **4** |
| --- | --- |
| **Tipus de requisit** | 14b. Requisits de suport |
| **Descripció** | Aquest requisit especifica el nivell de suport que el sistema requereix. |
| **Justificació del requisit** | Cal assegurar que el nostre sistema proporciona suport als nostres usuaris quan tenen dubtes o problemes quan fan ús de GreenWheel. |
| **Condició de satisfacció** | * S’oferirà un apartat de preguntes freqüents sobre l’ús de l’aplicació. Es proporcionarà la informació de contacte |

### Requisits de seguretat

| **Número** | **5** |
| --- | --- |
| **Tipus de requisit** | 15a. Requisits d’accés |
| **Descripció** | Especifica les persones autoritzades a accedir al producte, tant a les diverses funcionalitats que ofereix com a les dades que conté, en quines circumstàncies es concedeix l’accés i a quines parts del producte es permet l’accés. |
| **Justificació del requisit** | Cal que les dades introduïdes pels usuaris siguin correctament emmagatzemades a la nostra aplicació. |
| **Condició de satisfacció** | * El sistema validarà les dades introduïdes pels usuaris abans de modificar la base de dades per tal de comprovar que siguin correctes. |

| **Número** | **6** |
| --- | --- |
| **Tipus de requisit** | 15c. Requisits de privacitat |
| **Descripció** | Especifica com el sistema garanteix la privadesa de la informació que emmagatzema dels clients. Així doncs, és imprescindible tenir en compte totes les lleis relacionades amb la privadesa de les dades tant d’un usuari com d’una empresa. |
| **Justificació del requisit** | Un dels valors fonamentals de la nostra empresa és el compromís de mantenir la informació protegida i només a l'abast del personal amb autorització d’accés, per tal d’evitar atacs a la seguretat de les dades del sistema i de la informació dels usuaris. |
| **Condició de satisfacció** | * El producte ha de protegir la informació privada d’acord amb les lleis de privadesa pertinents i la política d’informació de l’empresa. * S’encripta tota la informació sensible, com les contrasenyes a la base de dades, i no es mostraran públicament. |

### Requisits de rendiment

| **Número** | **7** |
| --- | --- |
| **Tipus de requisit** | 15a. Requisits de longevitat |
| **Descripció** | L’aplicació s’ha de mantenir funcional en tot moment. |
| **Justificació del requisit** | Cal mantenir el sistema actiu durant un període llarg de temps per tal de complir els objectius que ens proposem amb el desenvolupament d’aquesta *App*. |
| **Condició de satisfacció** | * L’aplicació es mantindrà funcional fins a acabar el projecte que consisteix aquesta assignatura. |

Ara que ja tenim una estructura bàsica desenvolupada del que serà la nostra aplicació, podem mostrar quines tasques es veuen influides en els requisits no funcionals.

| **Número RNF** | **Número**  **Volere** | **Tipus de requisit** | **Història d’usuari** | **Estat** |
| --- | --- | --- | --- | --- |
| 1 | 11a | Facilitat d’ús | - | In progress |
| 2 | 11b | Personalització i internacionalització | [#37](https://tree.taiga.io/project/arnau147-pes-green-whee/us/37) | In progress |
| 3 | 11c | Aprenentatge | - | In progress |
| 4 | 14b | Suport | [#60](https://tree.taiga.io/project/arnau147-pes-green-whee/us/60), [#64](https://tree.taiga.io/project/arnau147-pes-green-whee/us/64) | In progress |
| 5 | 15a | Accés | [#29](https://tree.taiga.io/project/arnau147-pes-green-whee/us/29), [#73](https://tree.taiga.io/project/arnau147-pes-green-whee/us/73), [#46](https://tree.taiga.io/project/arnau147-pes-green-whee/us/46), [#39](https://tree.taiga.io/project/arnau147-pes-green-whee/us/39) | Completed |
| 6 | 15c | Privacitat | [#28](https://tree.taiga.io/project/arnau147-pes-green-whee/us/28), [#29](https://tree.taiga.io/project/arnau147-pes-green-whee/us/29) | Completed |
| 7 | 15a | Longevitat | - | In Progress |

## Aspectes transversals

Al llarg d'aquest primer Sprint hem aconseguit un bon progrés en alguns dels criteris transversals.

### Objectiu Final

* **Geolocalització**: Aquest aspecte transversal és el més important de la nostra aplicació mòbil i és un aspecte que hem d'aplicar i garantir. Volem tenir tot un sistema de geolocalització on tindrem tots els carregadors tant públics com privats dels llogaters en el mapa, marcadors personalitzats, podrem utilitzar el navegador de cerca per anar al mapa, generar rutes, filtrar etc...
* **Xarxes socials**: El nostre objectiu és tenir un sistema de log-in utilitzant serveis com el Racó o el servei de Google.
* **Xat**: Creiem que tenir un xat és molt important pels nostre usuaris per comunicar-se, llavors la nostra idea és desenvolupar un sistema de xat amb historial perquè es pugui realitzar negociacions.
* **Gamificació**: Sobre Gamificació volem implementar un sistema tant de valoracions com de nivells i trofeus i aquest ser mostrat en el perfil d'un usuari.
* **Stakeholders Reals**: Volem que al final de cada sprint mostrar el resultat d’aquest a un possible usuari final i que ens doni feedback de la usabilitat i sensacions generals de l'applicació.
* **Refutació**: Volem tenir un sistema de notificacions alhora que es generin reserves i tindrem un control bàsic sobre els usuaris que realitzin un mal ús de l'aplicació mitjançant el sistema de valoracions dels nostres usuaris o altres factors detectables.
* **Calendari**: Ara mateix dins del nostre abast, tenim planejat tenir un calendari intern a l'aplicació, però no pronostiquem realitzar sincronització amb google calendar.
* **Web-app admin:** Teníem planejat realitzar aquesta funcionalitat en el tercer sprint per tal de gestionar de manera més ràpida i eficient el sistema com el de reports per exemple.
* **MultiIdioma**: Volem una aplicació que pugui ser traduïda i utilitzada per diferents llengües. Llavors incorporarem un sistema que seleccioni un llenguatge dels que tinguem incorporats, es guarda la configuració de l'idioma.

### Estat Sprint 1

* **Geolocalització**: Ha estat implementat al llarg d'aquest sprint mitjançant les històries d'usuari #51 Inciar Ruta, #54 Mostrar ubicació actual com altres tasques com l'edició del marcador en el mapa o mostrar els carregadors públics en el mapa. Volem una aplicació que pugui ser traduïda i utilitzada per diferents llengües. Llavors incorporarem un sistema que seleccioni un llenguatge dels que tinguem incorporats, es guarda la configuració de l'idioma.
* **Xarxes socials**: En aquest primer sprint hem decidit no implementar res sobre xarxes socials, però tenim pensat implementar-ho en futurs sprints.
* **Xat**: En aquest primer sprint només hem implementat el botó de xat, però tot el sistema de comunicació d'usuaris amb el xat ho hem deixat pels següents sprints.
* **Gamificació**: Sobre Gamificació el sistema de valoracions com trofeus està il·lustrat però no implementat.
* **Stakeholders Reals**: Un cop acabat el primer sprint vam consultar a un membre de la família d'un company que posseeix un cotxe elèctric perquè intentes fer servir la nostra aplicació. Amb aquesta prova vam arribar a la conclusió de què la nostra app en aquest moment li faltaven meses funcionalitats per a ser usable, els resultats d’aquest test es comentaran durant la reunió de la planificació del sprint 2.
* **Refutació**: En aquest sprint hem decidit no implementar res sobre bloquejos de compte i notificacions.
* **Calendari**: En aquest sprint hem decidit no implementar res sobre calendaris.
* **Web-app admin:** No hem implementat res en aquest sprint sobre web-apps.
* **MultiIdioma**: Aquesta funcionalitat s'ha implementat mitjançant la història d'usuari #37 Seleccionar Idioma on podem seleccionar l'idioma, guardar aquesta elecció d'idioma i generar el sistema d'internacionalització.

| **Aspecte transversal** | **Estat actual** |
| --- | --- |
| Geolocalització | ✔ |
| Xarxes Socials | **╳** |
| Xat | **╳** |
| Gamificació | **╳** |
| Stakeholders Reals | **╳** |
| Refutació | **╳** |
| Calendari | **╳** |
| Web-app admin | **╳** |
| MultiIdioma | ✔ |

### Estat Sprint 2

* **Geolocalització**: Va estar implementada al primer sprint
* **Xarxes socials**: En aquest primer sprint hem decidit no implementar res sobre xarxes socials, però tenim pensat implementar-ho al següent sprint.
* **Xat**: Al primer sprint es va implementar el botó de xat, però tot el sistema de comunicació d'usuaris amb el xat ho hem deixat per l’últim sprint.
* **Gamificació**: Sobre Gamificació el sistema de valoracions com trofeus està il·lustrat però no implementat. S’implementarà al sprint següent
* **Stakeholders Reals**: Un cop finalitzem aquest sprint, abans del planing del tercer sprint demanarem a diversos usuaris que utilitzin la nostra App i ens donin feedback per poder millorar la usabilitat de cara al següent sprint.
* **Refutació**: En aquest segon sprint hem decidit no implementar res sobre bloquejos de compte i notificacions.
* **Calendari**: S’ha implementat un calendari per a determinar les hores i els dies que es vol reservar un carregador i una bici. Es intern del sistema, no es sincronitza amb eines externes
* **Web-app admin:** No hem implementat res en aquest segon sprint sobre web-apps.
* **MultiIdioma**: Aquesta funcionalitat es va implementar internament al primer sprint. Funciona internament però no està feta la implementació completa encara. Es realitzarà al següent sprint

| **Aspecte transversal** | **Estat actual** |
| --- | --- |
| Geolocalització | ✔ |
| Xarxes Socials | **╳** |
| Xat | **╳** |
| Gamificació | **╳** |
| Stakeholders Reals | ✔ |
| Refutació | **╳** |
| Calendari | ✔ |
| Web-app admin | **╳** |
| MultiIdioma | ✔ |

## Serveis

Durant aquest sprint la comunicació amb els altres grups ha estat suficient, hem anat comunicant-nos o cada dues setmanes aproximament sobre les funcionalitats i sobre noves notícies com ens ho enviaran. Respecte a l'últim sprint no hi ha casi canvis apart de que ara mateix sabem mes o menys com rebrem o enviarem les funcionalitats.

Anteriorment (Inception 2)

* **Funcionalitat rebuda incepció:** El valor de contaminació de la zona on es troba una coordenada qualsevol en format API.
* **Funcionalita donada incepció:** Donada una zona o coordenada, retornar el conjunt de punts de càrrega o llocs de lloguer de bicicletes propers de la zona.

Anteriorment(Sprint 1)

* **Funcionalita rebuda:** Amb el el grup d’aquesta funcionalitat s’ha parlat poc més sobre la funcionalitat, només ens hem ficat d’acord a recordar la funcionalitat en concret que es l’esmentada a l’incepció 2 però amb un retoc de precisió de la magnitud de la crida.

**Funcionalitat** : El valor de contaminació de la zona on es troba una coordenada qualsevol en format API (Dins de Catalunya).



* **Funcionalita donada:** Amb l’altre grup ”Cultura” hem parlat sobre la magnitud de la funcionalitat que els hi donem. Ens han comunicat que ells voldríen primer de tot una crida mitjançant un filtre, sigui els radi de n km propers de la zona o una quantitat de reserves properes a la zona.(Per exemple els 40 punts propers a la zona.)

Finalment al parlar amb el grup hem decidit ampliar i actualitzar la proposta de donar-lis 2 endpoints.

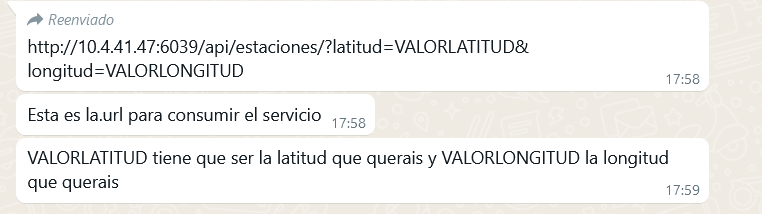
**Funcionalitat 1** : Donada una zona o coordenada i un número indicant km, retornar el conjunt de punts de càrrega o llocs de lloguer de bicicletes propers de la zona per criteri de km.

**Funcionalitat 2** : Donada una zona o coordenada i un número, retornar el conjunt de punts de càrrega o llocs de lloguer de bicicletes, on es retornen exactament el numero esmentat de punts més propers a la zona.



Actualment (Sprint 2)

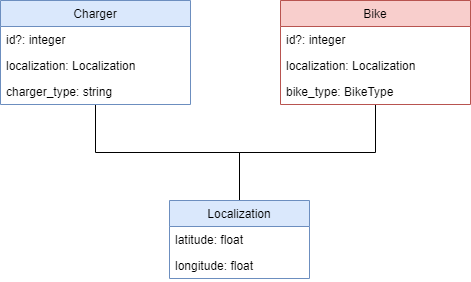
* **Funcionalita rebuda:** La funcionalitat que rebem d’aquest grup ja està establerta i ja sabem més o menys com ens la enviaran.



Seguint aquest exemple, ells ens enviaran un usuari que té els permisos per accedir al seu servei desde una VPN. Desde aquesta VPN accedirem a la crida enviant-lis els query parameter de latitud i longitud. Això retorna una response amb el valor de contaminació d’aquella zona.

**Funcionalitat** : En conclusió, ens donen el valor de contaminació de la zona on es troba una coordenada qualsevol en format API (Dins de Catalunya).



* **Funcionalitat donada:** Amb el grup de Cultura, les funcionalitats que donem no hi ha hagut casi canvis, s’han concretat mes o menys els filtres. Aquestes funcionalitats que li donem seran mitjançant la api que tindrem hostejada a Amazon Web Services, !IMPORTANT: Els hi donarem un document on els hi especificarem pas a pas com es conectaran amb la nostre API. Quin usuari accedira, amb quines credencials per accedir-hi etc… i amb totes les crides i paràmetres que tindran a disposició per tal de realitzar les nostres funcionalitats. En el nostre cas les crides els hi retorna lo que es mostra en aquest UML del punts(localitzacions dels carregadors i bicis de la nostra aplicació).
* **Funcionalitat** : Aquestes crides serán GETs on es podran passar filtres com a query parameters. Les crides seguiran exemples com aquests:
* [http://ip:port/api/bikes](about:blank) http://ip:port/api/chargers/public?query\_param=valor&&query\_param2=valor2 http://ip:port/api/chargers/public
* 



## Stakeholders

A tot projecte de qualsevol empresa hi participen unes parts interessades, que són persones o entitats afectades per les activitats de la companyia. En el cas de GreenWheel comptem amb un llistat de quinze *stakeholders*, als que els afecta les nostres activitats ja sigui des del negoci de carregadors elèctrics fins a les tecnologies que emprem per a desenvolupar la nostra *App*:

### 2.6.1. Conductors de cotxes elèctrics endollables

Usuaris que fan servir la nostra aplicació per traçar rutes fins als punts d’estacionament de recàrrega (públics/privats) per carregar el seu automòbil elèctric.

### 2.6.2. Usuaris que lloguen bicis

Persones que volen realitzar una ruta o un desplaçament en bicicleta i volen llogar-ne una mitjançant la nostra aplicació.

### 2.6.3. Posseïdors de punts de càrrega

Usuaris que posseeixen un punt de carregar per a vehicles elèctrics endollables, i els volen llogar a canvi d’una remuneració.

### 2.6.4. Posseïdors de bicicletes

Usuaris que posseeixen una bicicleta que no li donen ús i la volen llogar mitjançant la nostra plataforma a canvi d’una remuneració.

### 2.6.5. Administrador/moderador del sistema

Persona o grup de persones que s'encarreguen de gestionar les queixes i reports mitjançant la pàgina de moderació.

### 2.6.6. Equip de desenvolupament

Equip compost pels desenvolupadors que seran els encarregats de crear la plataforma i assegurar el seu correcte funcionament.

### 2.6.7. Competència

Totes les altres empreses que ofereixen plataformes que gestionen punts de càrrega o lloguers de bicicletes com les que hem esmentat.

### 2.6.8. Generalitat

Proporcionen una base de dades dels punts de càrrega públics que hi ha per tots el territori Català.

### 2.6.9. Altres equips de PES

L'equip de AB&B proporciona una *API* que indica la contaminació en diferents punts del mapa.

### 2.6.10. *Product Owner*

Són els inversos de l'aplicació i els que indican quina és la finalitat d’aquesta i les funcionalitats que hi ha d’incorporar. En aquest cas són els professors de PES.

### 2.6.11. *Scrum Master*

És l’encarregat de dirigir cada una de les iteracions, definir el que s’ha de fer en cada una i assegurar-se de què la feina es fa.

### 2.6.12. *Google*

*Google* ens oferirà dos serveis, l'*API* de maps que la utilitzarem per a tota la interfície del mapa en la nostra app i *Google authenticator* per iniciar sessió. A part la nostra aplicació s’executarà sobre *Android* sistema operatiu que *Google* ofereix.

### 2.6.13. El Racó

Ens oferirà mitjançant la seva api una forma d'autenticació mitjançant les credencials UPC per als alumnes de la FIB.

### 2.6.14. *Github*

Ens donarà les eines per a la gestió de codi i pel desplegament d’aquest CI/CD.

### 2.6.15. *AWS*

Ens proveirà de diferents serveis per gestionar l’allotjament de la nostra aplicació. Més endavant expliquem quins són els serveis emprats.

# 

# Agile ceremony

En general quedem tots satisfets de tot el que s’ha treballat durant aquest segon *sprint*. Ens reunim, com a mínim, tres vegades a la setmana i en aquestes reunions s’ha gestionat molt bé el progrés de cadascuna de les tasques que tenia cadascú.

Hem dedicat bastantes hores a tasques que no tenien punts i per tant no es veurà representat tot l’esforç dedicat al gràfic de velocitat.

## Reunions de Sprint Planning, review i retrospectiva

A continuació es pot veure, sprint per sprint, com ha anat el projecte i el que ens cal millorar per tal d’anar millor en els següents sprints. La idea és aplicar millores a tot allò que veiem que es negatiu, i valorar allò que fem bé per tal de seguir-ho fent. També desestimar totes aquelles iniciatives que no han acabat funcionant o no han anat tal i com esperàvem.

### 3.1.1. Sprint Planning

#### Sprint 1

En aquest primer sprint, el primer que vam fer va ser configurar totes les eines per començar a programar i així començar a desenvolupar les primeres funcionalitats que haviem plantejat. La manera com ens vam dividir la feina va ser per històries, de manera que cada membre de l’equip es dedicava a realitzar totes les tasques d’aquesta. Aquestes podien ser tasques de *backend* o de *frontend*, així que tots els membres del grup aprengués una mica dels dos *frameworks* amb els que haviem de desenvolupar l’aplicació (*Flutter* i *Django*).

Per a assignar les tasques, en primer lloc vam decidir quines històries serien del primer sprint i, seguidament quines tasques tindrien. Les tasques es vam anar assignant en grup, de manera que tots els membres tinguéssim més o menys la mateixa càrrega de feina. Més aviat no vam assignar tasques si no històries per a cada membre del grup, de manera que totes les tasques de la història d’usuari anirien assignades al que s’encarrega d’aquesta.

També es van definir a partir d’etiquetes al Taiga quines tasques eren purament de *frontend* i quines de *backend* de manera que ni hi hagués cap mena de confusió del que s’ha de desenvolupar. Finalment, vam puntuar cadascuna de les tasques d’acord a la puntuació total de la història, així doncs la suma de les puntuacions de les tasques era igual a la puntuació de la història la qual formen part.

| **Èpiques** | **Històries d'usuari** | **Points** | **Front/Back** | **id Taiga** | **Tasks** | **Temps real** |
| --- | --- | --- | --- | --- | --- | --- |
| **Gestió Usuaris** | Log In natiu | 8 | F | 164 | Mostrar pàgina Login | 2 |
| B | 165 | Autenticació de l'usuari | 1 |
| F | 166 | Configuració API KEY als serveis | 1 |
| F | 175 | Persistència usuari loguejat | 2 |
| B | 222 | Protecció del backend amb la api key | 2 |
| Registre d'usuari | 13 | F | 168 | Mostrar opció recuperar contrasenya | 1 |
| B | 223 | Enviar correu de recuperació de contrasenya | 2 |
| F | 224 | Mostrar pantalla per a canviar la contrasenya | 7 |
| B | 225 | Guardar canvi de contrasenya | 1 |
| B | 169 | Guardar usuari | 1 |
| F | 167 | Mostrar pàgina registre | 11 |
| Edició perfil | 8 | F | 170 | Mostrar edició perfil | 9 |
| B | 226 | Obtenir informació usuari | 1 |
| B | 171 | Guardar canvis edició perfil | 1 |
| Veure Perfil | 5 | B | 173 | Obtenir informació de l'usuari a mostrar | 1 |
| F | 272 | Mostrar punts de l'usuari | - |
| F | 172 | Mostrar perfil | 1 |
| Tancar Sessió | 2 | F | 174 | Habilitar tancar sessió | 1 |
| **Gestió Carregadors** | Edició carregador | 8 | F | 95 | Formulari editar carregador | 10 |
| B | 227 | Obtenir informació carregador | 1 |
| B | 130 | Guardar dades al sistema | 10 |
| **Gestió rutes** | Cancel·lar ruta | 5 | F | 190 | Mostrar opció cancel·lar ruta | - |
| **Gestió Bicicletes** | Veure bicicleta concreta | 13 | B | 178 | Obtenir informació bibicleta completa | 10 |
| F | 177 | Mostrar informació bicicleta completa | 10 |
| F | 176 | Mostrar informació bicicleta reduïda | 8 |
| Afegir bicicleta | 21 | F | 179 | Fer formulari per penjar bicicleta | 3 |
| B | 180 | Fer model bicicleta | 2 |
| B | 181 | Guardar bicicleta | 3 |
| B | 182 | Guardar imatges a S3 | 5 |
| F | 183 | Passar imatges al backend | 3 |
| Eliminar bicicleta | 5 | B | 185 | Eliminar bicicleta | 1 |
| F | - | Mostrar opció eliminar | - |
| Editar bicicleta | 8 | F | 188 | Mostrar formulari per editar bicicleta | 4 |
| B | 228 | Obtenir informació de la bicicleta | 3 |
| B | 189 | Guardar edició bicicleta | 2 |
| Mostrar punts de bicicleta | 13 | F | 186 | Mostrar tots els punts | 8 |
| F | 245 | Mostrar tots els punts en format llista | 5 |
| B | 187 | Obtenir punts bicicleta | 2 |
| **Gestió reserves** | Reservar un carregador | 21 | F | 99 | Notificar propietari | - |
| F | 98 | Seleccionar i reservar un time slot | - |
| F | 229 | Mostrar calendari | 4 |
| F | 230 | Mostrar selector d'hora | 5 |
| F | 97 | Mostrar time slots lliures | 10 |
| B | 148 | Guardar reserva al sistema | 5 |
| Reservar bicicleta | 13 | B | 208 | Crear reserva bicicleta | 2 |
| F | 231 | Seleccionar i reservar una bicicleta | 5 |
| F | 207 | Mostrar opció reserva bicicleta | 5 |
| Administració reserves (propietari) | 21 | F | 233 | Mostrar reserves pendents | - |
| F | 234 | Mostrar reserves passades | - |
| F | 235 | Mostrar reserves actuals (acceptades però que encara no han finalitzat) | - |
| B | 217 | Acceptar/denegar petició reserva | 2 |
| F | 236 | Mostrar botons acceptar/ denegar reserva | - |
| Bloquejar franja | 13 | B | 218 | Bloquejar franja | 3 |
| F | 237 | Mostrar selecció de franjes a bloquejar | - |
| F | 238 | Mostrar franges cancelades | - |
| F | 239 | Botó de cancel·lar bloqueig de franja | - |
| B | 240 | Cancel·lar bloqueig de franja | 1 |
| Calendari reserves (propietari) | 13 | B | 215 | Cancel·lar reserva d'un usuari | 1 |
| F | 241 | Mostrar reserves sobre el calendari | - |
| F | 242 | Canviar vista calendari de mes a hora | 5 |
| B | 273 | Mostrar franjes bloquejades | 5 |
| F | 214 | Mostrar calendari | 2 |
| Eliminar reserva bicicleta | 5 | B | 199 | Eliminar reserva bicicleta de la base de dades | 1 |
| F | 198 | Mostrar opció eliminar reserva de bicicleta | 4 |
|  | Tasques sense història | 34 | F | 219 | Refactoring Codi Frontend | 41 |
| B | 220 | Refactoring Codi Backend | 22 |
| B | 160 | AWS EC2 | 20 |
| B | 161 | AWS S3 | 5 |
| B | 162 | AWS Cloud Formation | 20 |
| B | 163 | CI/CD | 21 |
|  | **TOTAL** | **195** |  |  | **REAL** | **305** |

#### Sprint 2

En aquest segon sprint, es va enfocar de manera diferent la planificació de les tasques aprenent dels errors comesos a l’anterior sprint. Vam decidir de fer que tots els membres actuessin de *full-stack developers* i vam acordar separar l’equip en *backend* i *frontend*, 2 i 5 membres a cadascun respectivament. Els de *backend* s’encarregarien de configurar totes les eines necessàries per a l'aplicació estigués desplegada i realitzar tots els endpoints que fessin falta a *frontend*, mentre que l’equip de *frontend* es dedicaria a desenvolupar les vistes i les funcionalitats que tocaven.

Pel que fa a *frontend* ens vam adonar que calia fer un *refactoring* general de les vistes per tal d’evitar la repetició de codi i que tots els *widgets* estiguessin creats en fitxers diferents i que altres membres els poguessin reutilitzar. L’equip de *frontend* es va reunir i va decidir que la creació de widgets estaria documentada al xat de *widgets* del nostre servidor de Discord fent una captura del widget que es va a desenvolupar i el nom del fitxer. Amb aquest mètode s’evitaria que dos membres treballin en el mateix *widget* i que altres membres els poguessin utilitzar per a la funcionalitat que estaven programant.

Com al sprint anterior, vam decidir entre tots quines històries treballaríem en aquest sprint, recollint algunes que no s’havien realitzat al sprint 1. Aquesta vegada es van crear moltes més tasques respecte a l’anterior sprint, ja que vam trobar important granularitzar més la funcionalitat que representa la història en tasques més específiques. Totes aquestes també anaven etiquetades segons si eren de *frontend* o de *backend*, tot i que aquesta vegada no les vam puntuar ja que el professor ens va dir a la *review* del primer sprint que no calia.

Les tasques no han quedat ben repartides i això ens ha portat alguna conseqüència, i és que no s’han pogut acabar totes les tasques proposades per a aquest sprint

| **Èpiques** | **Històries d'usuari** | **Points** | **Front/Back** | **id Taiga** | **Tasks** | **Temps real** |
| --- | --- | --- | --- | --- | --- | --- |
| **Gestió Usuaris** | Log In natiu | 8 | F | 164 | Mostrar pàgina Login |  |
| B | 165 | Autenticació de l'usuari |  |
| F | 166 | Configuració API KEY als serveis |  |
| F | 175 | Persistència usuari loguejat |  |
| B | 222 | Protecció del backend amb la api key |  |
| Registre d'usuari | 13 | F | 168 | Mostrar opció recuperar contrasenya |  |
| B | 223 | Enviar correu de recuperació de contrasenya |  |
| F | 224 | Mostrar pantalla per a canviar la contrasenya |  |
| B | 225 | Guardar canvi de contrasenya |  |
| B | 169 | Guardar usuari |  |
| F | 167 | Mostrar pàgina registre |  |
| Edició perfil | 8 | F | 170 | Mostrar edició perfil | : |
| B | 226 | Obtenir informació usuari |  |
| B | 171 | Guardar canvis edició perfil |  |
| Veure Perfil | 5 | B | 173 | Obtenir informació de l'usuari a mostrar |  |
| F | 272 | Mostrar punts de l’usuari |  |
| F | 172 | Mostrar perfil |  |
| Tancar Sessió | 2 | F | 174 | Habilitar tancar sessió |  |
| **Gestió Carregadors** | Edició carregador | 8 | F | 95 | Formulari editar carregador |  |
| B | 227 | Obtenir informació carregador |  |
| B | 130 | Guardar dades al sistema | 10 |
| **Gestió rutes** | Cancel·lar ruta | 5 | B | 191 | Implementar cancel·lació ruta |  |
| F | 190 | Mostrar opció cancel·lar ruta |  |
| **Gestió Bicicletes** | Veure bicicleta concreta | 13 | B | 178 | Obtenir informació bibicleta completa |  |
| F | 177 | Mostrar informació bicicleta completa |  |
| F | 176 | Mostrar informació bicicleta reduïda |  |
| Afegir bicicleta | 21 | F | 179 | Fer formulari per penjar bicicleta |  |
| B | 180 | Fer model bicicleta |  |
| B | 181 | Guardar bicicleta |  |
| B | 182 | Guardar imatges a S3 |  |
| F | 183 | Passar imatges al backend |  |
| Eliminar bicicleta | 5 | B | 185 | Eliminar bicicleta |  |
| F | 184 | Mostrar opció eliminar |  |
| Editar bicicleta | 8 | F | 188 | Mostrar formulari per editar bicicleta |  |
| B | 228 | Obtenir informació de la bicicleta |  |
| B | 189 | Guardar edició bicicleta |  |
| Mostrar punts de bicicleta | 13 | F | 186 | Mostrar tots els punts |  |
| F | 245 | Mostrar tots els punts en format llista |  |
| B | 187 | Obtenir punts bicicleta |  |
| **Gestió reserves** | Reservar un carregador | 21 | F | 99 | Notificar propietari |  |
| F | 98 | Seleccionar i reservar un time slot |  |
| F | 229 | Mostrar calendari |  |
| F | 230 | Mostrar selector d'hora |  |
| F | 97 | Mostrar time slots lliures |  |
| B | 148 | Guardar reserva al sistema |  |
| Reservar bicicleta | 13 | B | 208 | Crear reserva bicicleta |  |
| F | 231 | Seleccionar i reservar una bicicleta |  |
| F | 207 | Mostrar opció reserva bicicleta |  |
| Administració reserves (propietari) | 21 | F | 233 | Mostrar reserves pendents |  |
| F | 234 | Mostrar reserves passades |  |
| F | 235 | Mostrar reserves actuals (acceptades però que encara no han finalitzat) |  |
| B | 217 | Acceptar/denegar petició reserva |  |
| F | 236 | Mostrar botons acceptar/ denegar reserva |  |
| Bloquejar franja | 13 | B | 218 | Bloquejar franja |  |
| F | 237 | Mostrar selecció de franjes a bloquejar |  |
| F | 238 | Mostrar franges cancelades |  |
| F | 239 | Botó de cancel·lar bloqueig de franja |  |
| B | 240 | Cancel·lar bloqueig de franja |  |
| Calendari reserves  (propietari) | 13 | B | 215 | Cancel·lar reserva d'un usuari |  |
| F | 241 | Mostrar reserves sobre el calendari |  |
| F | 242 | Canviar vista calendari de mes a setmana |  |
| F | 214 | Mostrar calendari |  |
| Eliminar reserva bicicleta | 5 | B | 199 | Eliminar reserva bicicleta de la base de dades |  |
| F | 198 | Mostrar opció eliminar reserva de bicicleta |  |
|  | Tasques sense història | 34 | F | 219 | Refactoring Codi Frontend |  |
| B | 220 | Refactoring Codi Backend |  |
| F | 221 | Widget Search Bar |  |
| B | 160 | AWS EC2 |  |
| B | 161 | AWS S3 |  |
| B | 162 | AWS Cloud Formation |  |
| B | 163 | CI/CD |  |
|  | **TOTAL** | **195** |  |  | **REAL** | 10 |

### 3.1.2. Retrospectiva

#### Sprint 1

Aquest ha estat el nostre primer sprint programant en equip, i ens hem trobat en varies coses que no han funcionat o hauríem de millorar de cara als següents sprints. A continuació, explicarem les diferents conclusions que hem tret a partir de la retrospectiva.

A millorar

La retrospectiva ha començat parlant d’allò que no ha anat gaire bé durant l’sprint i caldria que milloréssim de cara als següents sprints. Una de les coses que ens cal millorar, és la gran diferència d’hores que hi ha hagut entre els diferents membres, i ja només d’hores, sinó de feina útil realitzada, on alguns dels membres havien fet fet un volum bastant més gran que altres. S’ha dit d’intentar fer la mateixa quantitat de feina, o almenys que s’equilibri més entre els diferents membres i no hi hagi tanta càrrega pels que dediquen més hores.

També s’ha comentat que hem treballat de manera molt individual en general, la qual cosa ha fet que faltés contacte continu entre nosaltres i es fessin coses per duplicat, com per exemple widgets en el frontend, o no s’implementessin coses pensant que ja ho faria un altre. Per això s’ha dit d’intentar treballar més en equip i canviar la distribució de les coses, com per exemple, distribuir-se widgets en ves d’històries d’usuari o pantalles.

Un altre problema que hi ha hagut ha estat que hem perdut molt temps aprenent, ja que calia aprendre les dos tecnologies i fins i tot els llenguatges de programació en les que estaven fetes, fent que gran part de l’sprint s’hagués d’invertir en això. Això també ha fet que, per desconeixement de com utilitzar correctament les tecnologies i falta d’experiència, s’organitzessin malament els codis i es fessin sense seguir les pràctiques friendly dels frameworks. Per un costat, ha anat bé per aprendre a utilitzar els dos frameworks i tenir unes bases de com es fan les coses, així com el cost que té implementar una cosa al frontend i backend. També ha servit per a que la gent que no havia programat mai en aquesta divisió de frontend/backend, pogués veure de que anava i quina preferia. De cara als pròxims sprints, si que creiem que és millor que uns es centrin en frontend i uns altres en backend, i s’especialitzin en la tecnología que están utilitzant. Respecte el desordre de codi, també s’ha comentat que cal arreglar-ho el primer de tot de cara al pròxim sprint, ja que sinó ens farà les coses molt complicades.

La gestió del taiga i el github no ha estat la millor tampoc, tancant tasques molt més tard que quan es van realitzar, o el no assignar-se les tasques quan tocava. Pel pròxim sprint esperem que això millori i cadascú sigui més responsable de com gestiona les seves tasques.

L’últim problema així a destacar, ha estat que ens han faltat moltes tasques per fer de les que vam plantejar inicialment, sobretot degut al temps invertit en aprenentatge. De cara al següent planning, tenim clar que cal agafar menys tasques i plantejar-les millor.

A mantenir

Creiem que cal mantenir sobretot el bon rotllo que hi ha nosaltres i el treball en equip, podent-nos dir les coses que cal millorar i en que hem fallat. Aquest bon ambient ens ajudarà molt de cara als següents sprints, on al aplicar les millores, segur que ens va millor.

També cal mantenir l’estratègia de reunions per a parlar de l’estat de l’aplicació cada 2 o 3 dies a la setmana, fent que no passi massa temps i es puguin parlar els problemes en un temps raonable, ja que si un problema es bloquejant, pot fer que una persona vagi molt més lenta en el seu desenvolupament.

Creiem que les reunions que hem anat fent també han estat molt ben organitzades i ens han servit per a tenir una visió crítica de com anava avançant el projecte, així com d’ajuda-nos entre nosaltres.

Per últim, també creiem que cal mantenir l’alt nivell de documentació i ajuda que anem desenvolupant mentres anem programant, fent que entre uns i altres ens sigui més fàcil entendre què està passant.

A evitar

Cal evitar repetir codi entre varis membres, fent que es perdi molt de temps, duplicant les coses sense motiu. Per a solucionar-ho, cal parlar les coses més.

També cal evitar les llargues funcions i fitxers de moltes línies, reestructurant el codi ara que en sabem una mica més.

Per altra banda, també hem d’evitar anar a última hora i estar desenvolupant fins l’últim moment, la qual cosa afecta molt al desenvolupament de codi i al testeig d’aquest.

Relacionat amb el punt anterior, tenim el tema de treballar sense tests o sense gairebé fer proves, la qual cosa fa que puguin sorgir errors modificant una altra part del codi, o que es pengin parts del projecte amb errors senzills de trobar.

Per últim, cal evitar utilitzar males pràctiques a l’hora d’utilitzar els frameworks i documentar-se per a veure com és millor fer les coses.

#### Sprint 2

En aquest segon sprint, ja veniem amb l’experiència del anterior, i per tant, ja hem intentat millorar tot el que vam comentar en l’anterior. Això ha fet que milloressin les praxis a l’hora de treballar i el desenvolupament fos més eficaç.

A millorar:

En aquest sprint el nostre major problema ha estat la falta de temps i que vam poder començar a 2 setmanes de l’entrega, la qual cosa va fer que tot fos molt just i s’hagués de treure temps de tot arreu. Aquesta falta de temps ha estat provocada per varis motius, però sobretot per la setmana d'exàmens i haver-se de posar al dia d’algunes assignatures, ja que pes té un pes molt gran i fa que durant el curs no poguem dedicar el temps necessari a altres assignatures. De cara al tercer sprint, al tenir nadal, creiem que pot anar millor i podem anar més tranquils, però que si ens cal fer feina de manera regular.

Hem tingut un problema bastant gran amb la granularitat de les tasques, ja que al ser tasques una mica grans en alguns casos, ha fet que hi haguessin branques que duressin tot l’sprint, cosa que no volem que es repeteixi de cara als següents sprints. per això, caldrà fer un esforç extra al principi de l’sprint i fer la granularitat de les tasques més petites. El nostre objectiu és que una branca mai durés més de 1 setmana, i que aquesta duració fos molt excepcional.

Per altra banda, també hem vist que un problema que hem tingut es que alguns dels membres ens costa preguntar dubtes i perdem molt temps buscant solucions a problemes que altres ja han resolt anteriorment, i per això s’ha afegit a les “daily” la pregunta de si tens algun problema que et bloquegi i com et podem ajudar a solucionar-lo.

Cal organitzar millor les rutes, sobretot del frontend, ja que sinó es fa molt complicat que unes pantalles cridin a altres, fent que es monti un caos bastant gran. Això també es solucionarà de cara al següent sprint amb l’experiència guanyada i no fent branques de llarga duració.

Per últim, un dels problemes amb els que ens hem trobat, ha estat que hi havia branques que no han tingut pràcticament commits i que tot es feia en local, la qual cosa feia molt difícil pels altres membres saber com podrien utilitzar les funcionalitats dels companys. Això es solucionarà fent commits més sovint.

Encara que hem millorat respecte el passat sprint, la gestió del taiga i el github segueix sent bastant dolenta i en moltes ocasions no es segueixen les metodologies pactades, ja sigui per pressa o per fer-ho ràpid i sense adonar-se’n. Si anem millor de temps, no caldrà que ho fem tant ràpid i poguem posar la nomenclatura millor.

A mantenir:

En aquest sprint hem fet moltes coses millor que en l'anterior sprint, i que ara cal mantenir de cara al tercer i últim sprint del projecte.

Primer de tot, ha anat molt bé el tema de repartir-se widgets del frontend en ves de treballar tant per històries d’usuari, fent que el codi estigui molt millor organitzat, sigui més reutilitzable i no hi hagi risc de duplicar codi.

També cal destacar la iniciativa que va néixer sense voler de, quan ens posem a treballar en el projecte, posar-nos en un dels xats de veu del nostre discord, i quedar-se allà, per si algú coincideix amb tu a l’hora de programar, pugui entrar o pugui venir a preguntar-te dubtes, fent així que hi hagi més treball en equip i els dubtes puguin ser resolts molt més ràpidament.

Lligat amb el punt anterior, en aquest sprint hem començat a aplicar el pair programming en varies ocasions, com per exemple el refactoring de codi de frontend aplicant serializers o la configuració del servidor de aws. Això ens ajuda a avançar més ràpid i és una tècnica d’agile que ens ha donat molt bon resultat.

La incorporació de fer un excel amb els diferents endpoints que teniem (semblant a una documentació d’openapi de Swagger), ha ajudat molt a organitzar-se als encarregats de frontend per tal de no estar preguntant tota la estona als de backend com funcionaven les coses. Hem quedat que es continuarà mantenint de cara al següent sprint.

La creació d’un diagrama que descrigui la capa de domini a l’hora que s’anava fent, ha ajudat molt a l’equip de frontend a poder realitzar la seva seguint la mateixa forma, ja que es requeria que fossin iguals o molt semblants per a poder crear automatismes. També es mantindrà de cara al següent sprint.

A evitar:

Tot i que hem millorat respecte el primer sprint, encara hi ha coses que hem d’evitar fer de cara als pròxims sprints.

Primer de tot, caldria deixar de fer branques que englobin moltes coses i que s’allarguin durant tot l’sprint, ja que dificulta molt la integració de l’aplicació, sobretot si té dependències.

També cal seguir evitant l’escriure codi hardcodejat en el frontend, i en cas que s’hagi de fer, que es faci en el backend.

També caldria evitar deixar les coses per l'últim moment i estar fins l’últim dia desenvolupant funcionalitats, ja que dificulta molt les proves i porta a que es puguin provocar molts errors.

Per últim, cal evitar el treballar sense fer proves o fer-ne molt poques, ja que porta a que els companys puguin tenir molts errors.

### 3.1.3. Review

#### Sprint 1

La *review* del primer sprint no va anar gaire bé. En primer lloc, cap dels membres del grup era conscient de que la *review* es tractava d’una petita *demo* davant el professor, per tant no vam estar prou preparats. Només el *sprint master* tenia un mòbil operable per a fer la *review*, de manera que vam trobar molts errors d’*overflow* a la part programada en *Flutter*, fet que alguns botons estiguessin sobreposats o objectes sortits de la pantalla. Tampoc vam connectar-ho bé amb el *backend* i algunes funcionalitats no funcionaven, com mostrar les reserves o crear una ruta cap a un punt del mapa (funcionalitat la qual no funcionava perquè no teníem connectada l’API de pagament que ho realitzava).

En segon lloc, el professor ens va fer veure que moltes històries que estaven tancades al *Taiga* no ho podien estar ja que totes les tasques no estaven fetes al 100%. També vam veure que el que es documentava als criteris d’acceptació havia d’estar reflectit a la funcionalitat de l’aplicació, fet que amb el que es tenia per aquest sprint no estava ben fet.

Per últim, el professor ens va comunicar que ens havíem passat amb les hores treballades per persona i que hauríem d’afegir al *project record track* una opció per a afegir la tasca a la qual havíem treballat, per tal de controlar que ens centressim en fer feina que prèviament havia estat planejada.

Com a conclusió d’aquesta primera *review* tindrem en compte anar més preparats per a la del següent sprint i així anar millorant amb la nostra metodologia de treball.

## Estadístiques de l'avanç del projecte

Els gràfics que informen l’avanç del sprint i el *burndown* han estat extrets del *Taiga*. Aquestes van determinades segons quan els membres del grup posen a l’apartat de *closed* totes les tasques de la respectiva història que els ha tocat treballar.

### 3.2.1. Sprint Burndown

#### Sprint 1

En aquest *burndown* que hi ha a continuació podem veure l’evolució amb la què hem anat completant tasques al llarg de l’sprint de color verd clar i l’evolució que representa anar completant tasques d’una manera constant amb verd fosc. Tot i que no hem pogut acabar totes les tasques que vam assignar en un principi per aquest primer sprint, erem conscients que ens havíem posat bastanta feina, així que estem contents de les tasques realitzades. També podem veure que sembla que fins a mitjans del sprint no hem fet gaire feina, però no és així, això és degut a que havíem dividit les històries d’usuari en varies tasques, i fins que no es completen totes les tasques, l’història no es considera com a tancada i, per tant, no és fins aquest moment que es veu reflexada l’evolució al *burndown*.

### 

Figura 5. Sprint 1 burndown

#### Sprint 2

Amb el següent *burndown* podem veure reflexat el que s’ha comentat al document i és que degut a exàmens i treballs, la majoria de l’equip va començar a treballar després d’una setmana de la inicialització de l’sprint. També és veritat que com bé vam dir al primer sprint, que el burndown no comenci a disminuir fins el 20 de novembre no vol dir que no haguem fet res fins llavors, simplement que no s’han començat a tancar històries sanceres fins llavors. Les tasques no s’han pogut realitzar totes ja que vam balancejar malament els punts i algun company es va quedar amb masses punts i no els ha pogut acabar. També hem de tenir en compte que vam heretar unes quantes tasques del primer sprint degut a que no havia hagut temps de fer-les.

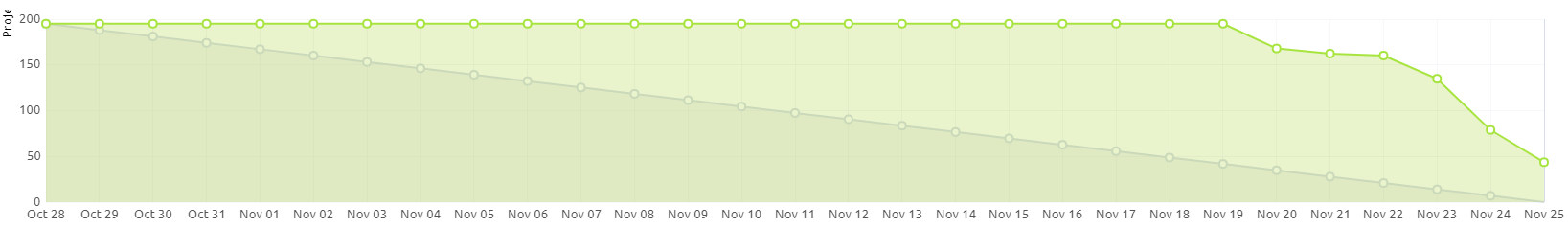


Figura 6. Sprint 2 burndown

### 3.2.2. Release Burndown

#### Sprint 1

Amb aquest segon *burndown* veiem que 474 són els punts que ens queden restants i 405 són els que ens haurien de quedar per fer, així que podem veure que tampoc anem gaire endarrerits segons la nostra planificació. Això sí, al següent sprint ens tocarà fer més punts dels que teníem planejats per tal de començar el tercer sprint amb els punts restants que vam estimar.

### 

Figura 7. Sprint 1 release burndown

Amb el següent gràfic podem veure que ens hem quedat endarrerits respecte als punts que hauríem de portar fets fins al moment, concretament distem de 160 punts dels punts estimats. Tot i que som conscients que són bastants punts, també contem amb que el pròxim sprint tindrem tots més temps per dedicar al projecte. Fem reflexió dels problemes que hem tingut en aquest sprint i pel següent ens posarem encara més les piles.

### 

Figura 8. Sprint 2 release burndown

### 3.2.3. Velocity chart

Al gràfic de barres que es veu a continuació es mostren els punts estimats que havíem calculat que hauríem de fer als prints i els punts que finalment hem realitzat. Com veiem aquest segon sprint tampoc hem arribat a complir-los tots degut als entrebancs explicats prèviament.

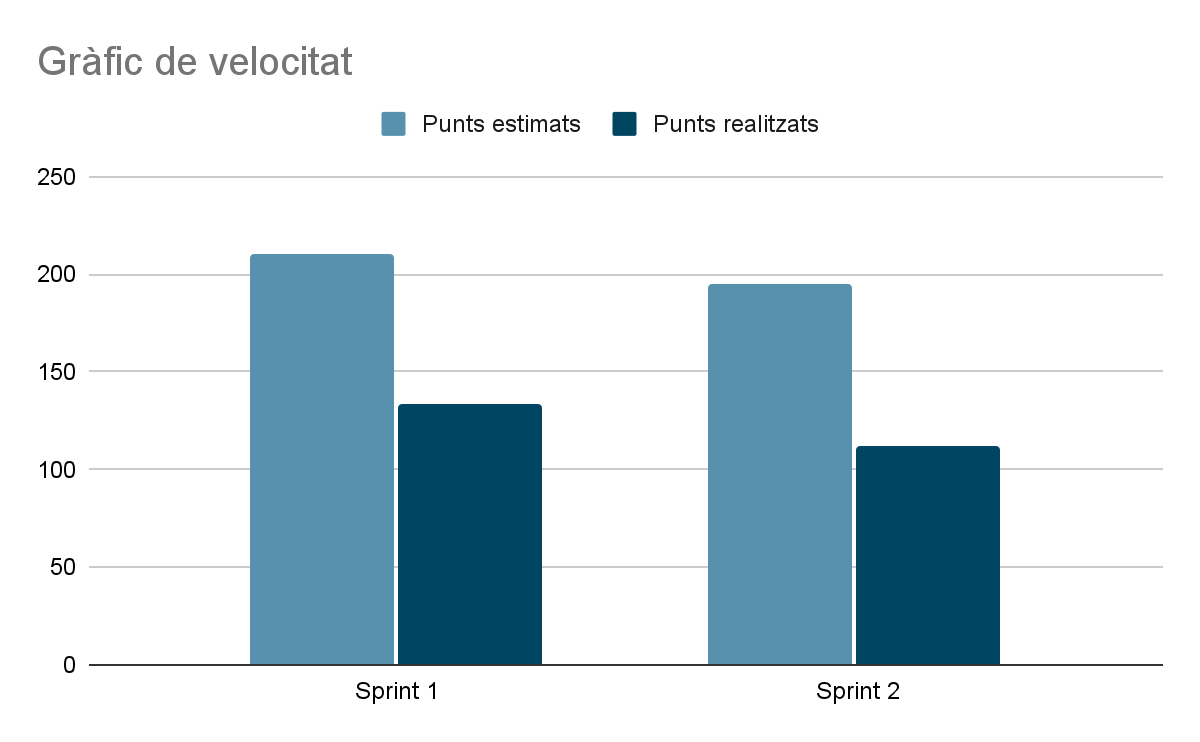


Figura 9. Velocity chart

### 3.2.4. Effort chart

Per últim mostrarem un gràfic que representa l’esforç estimat i l’esforç realitzat al llarg dels sprints. Amb el gràfic veiem que en el segon sprint hem realitzat més o menys l’esforç estimat però en realitat hem dedicat moltes més hores, però aquestes pertanyien a *storyless tasks*, entre elles les tasques de refactoritzar el codi i montar la configuració del servidor.

### GràficFigura 10. Effort chart

# Metodologia

En aquest apartat explicarem la metodologia que estem emprant en aquesta fase d’Incepció i una visió general de com ens organitzarem quan comencem els sprints. Aquest apartat s’anirà actualitzant a mesura que anem avançant el projecte, on detallarem quines metodologies hem fet servir en els diferents aspectes del treball.

## Visió general

El marc de treball que hem escollit per a gestionar el nostre projecte és la metodologia *Scrum*. En aquesta tenim una fase d’*Inception*, on realitzarem tota la documentació que ens ajudarà a començar el desenvolupament, i tres fases de *Sprint*. Cadascuna d’aquestes tindrà un o diversos responsables que s’encarregaran de supervisar que tots els membres de l’equip realitzen les tasques que toquen en aquella fase. A part, tenim un responsable que s’encarrega de les relacions amb els altres grups per tal d’acordar quins serveis hem de proporcionar al seu projecte i quins ens ofereixen al nostre sistema.

Pel que fa a les reunions, a banda de les reunions generals que duem a terme a classe, quasi cada dia estem en contacte via el xat de Discord i diverses vegades a la setmana fem reunions, no de tot l’equip a la vegada, però si uns quants membres que estan realitzant una tasca concreta. Cada dissabte al matí, fem una *weekly* amb tot l’equip per tal de repassar el que s’ha portat a cap al llarg de la setmana i el que es farà la setmana següent. De moment, durant primeres setmanes de la fase d’*Inception* hem anat executant conjuntament tasques que tots els membres de l’equip han de tenir present per tal d’entendre el funcionament del sistema: èpiques, històries d’usuari, *planning poker*, UML, entre d’altres. Més endavant als sprints hem anat seguint el mateix model d’anar creant les tasques conjuntament, així com assignar-les.

En estar en contacte molt sovint, ens proporcionem *feedback* entre nosaltres en totes les tasques que du a terme cada membre. Donat això, respectem els principis de les metodologies àgils, on ens adaptem als canvis i ajustem i debatem les tasques fetes contínuament.

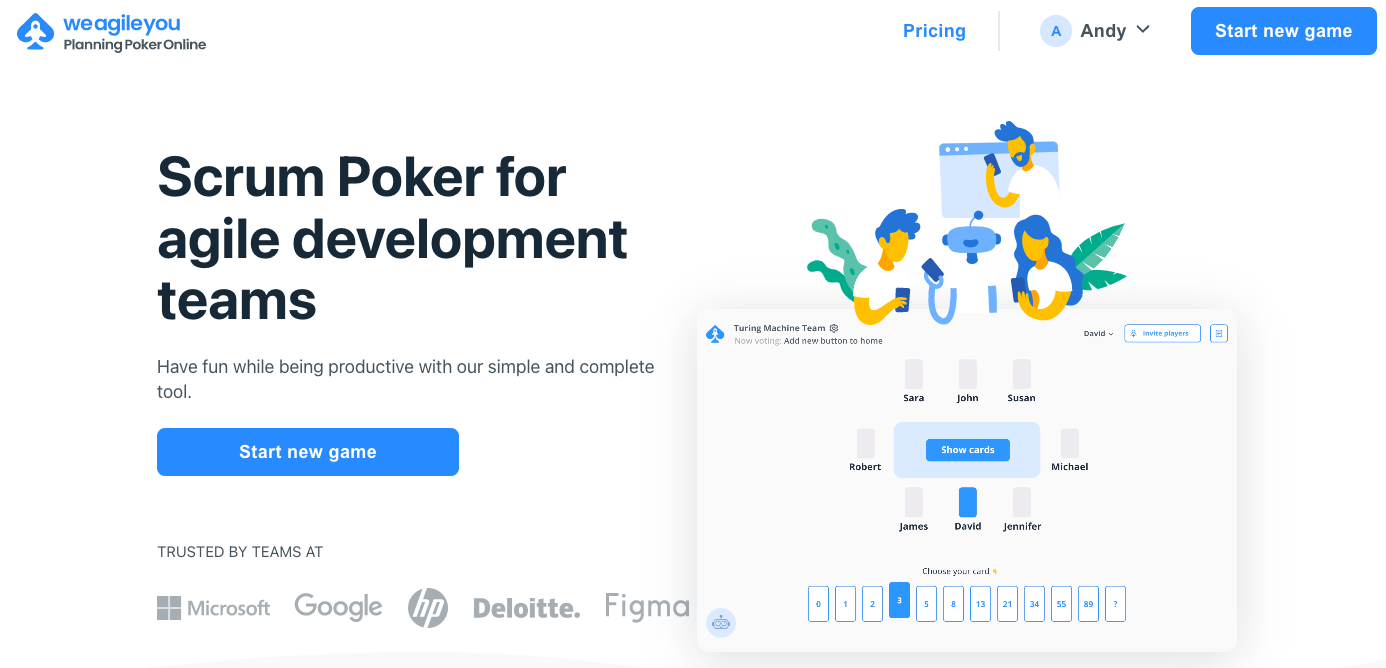
Pel que fa als *sprints*, ja hem decidit que setmanalment cadascun dels membres del grup tindrà diverses tasques a realitzar. Aquestes poden tenir relació a una història d’usuari i, per tant, aniran relacionades amb el desenvolupament de l’*App*, i d’altres poden ser tasques externes relacionades amb la documentació, *deployment* o gestió del *Git*. Tot això, estarà acordat i assignat a través de la plataforma *Taiga*.

A continuació expliquem per apartats, amb més profunditat, les diverses tecnologies i metodologies que emprem en cada part del nostre projecte:

## Gestió del projecte

Per a seguir la metodologia *Scrum*, fem servir l’eina *Taiga*, que ens permet crear tasques i assignar-les a determinats membres del grup. També ens permet crear un *backlog* amb totes les històries d’usuari, classificades per èpiques, que el nostre sistema suporta. A mesura que anem desenvolupant, anirem classificant aquestes històries en cadascun dels 3 *sprints* segons ens convingui. Cadascuna d’aquestes conté una puntuació, referent a la càrrega de treball que aproximem que suposarà. *Taiga* permet dividir la puntuació de les històries en la càrrega que suposa pel desenvolupament del *frontend*, *backend*, disseny i *user experience*, de les quals considerem més importants, de moment, el pes que implicarà el *frontend* i el *backend* per la qual cosa hem eliminat les altres opcions.

Aquesta puntuació de cadascuna de les històries d’usuari va ser decidida entre tots els membres realitzant un *planning poker*, on vam utilitzar el web Planning Poker Online[**[4]**](#kix.bpjbt7hocose). En aquesta tècnica vam anar discutint cadascuna de les històries d’usuari que havíem acordat i analitzant tot el que comportava a l’hora de desenvolupar. Seguidament, cadascun dels membres triava a través del web mencionat anteriorment, la puntuació que crèiem que suposaria la respectiva història d’usuari. Un cop revelades aquestes puntuacions el web en deia la mitjana i nosaltres vam acordar d’arrodonir aquest nombre cap amunt.



Abans d’acordar el pes de cadascuna de les històries d’usuari, vam establir primer què representava una unitat d’aquesta puntuació. Vam decidir que 1 punt de càrrega equivalgués a 1 hora de treball.

A banda de la càrrega que suposa una història també hem acordat què representa la realització d’una història o un sprint, també conegut com a *Definition of Done* (DoD). Aquest, pel que fa a les històries, anirà determinat per una sèrie d'elements clau: s’han finalitzat totes les tasques relacionades en aquesta història, tots els tests generats d’acord els criteris d’acceptació s’han superat i tenir un codi sense errors ni *bugs*. Pel que fa als sprints, considerem que estan completats quan s’ha finalitzat el desenvolupament de totes les històries d’usuari correctament, no hi ha errors ni *bugs*, els tests d’integració han estat superats i la documentació ha estat completada. Un cop acabat el *sprint*, no comencem el següent fins que no haguem revisat tots els detalls que podrien faltar o els aspectes que podríem millorar, així com els errors que s’hagin pogut cometre (*retro*). Això ens facilita a començar els pròxims *sprints* de manera més àgil i sense haver-nos de preocupar del funcionament de les tasques fetes a l’anterior *sprint*.

Quan canviem de *sprint* gestionem el canvi d’iteració recollint totes aquelles tasques que no s’han pogut completar. Aquestes com que formen part d’una història, aquesta es passarà altre cop al backlog fins a decidir si s’assigna al pròxim sprint, ja que no queda tancada fins que totes les tasques estan completades.

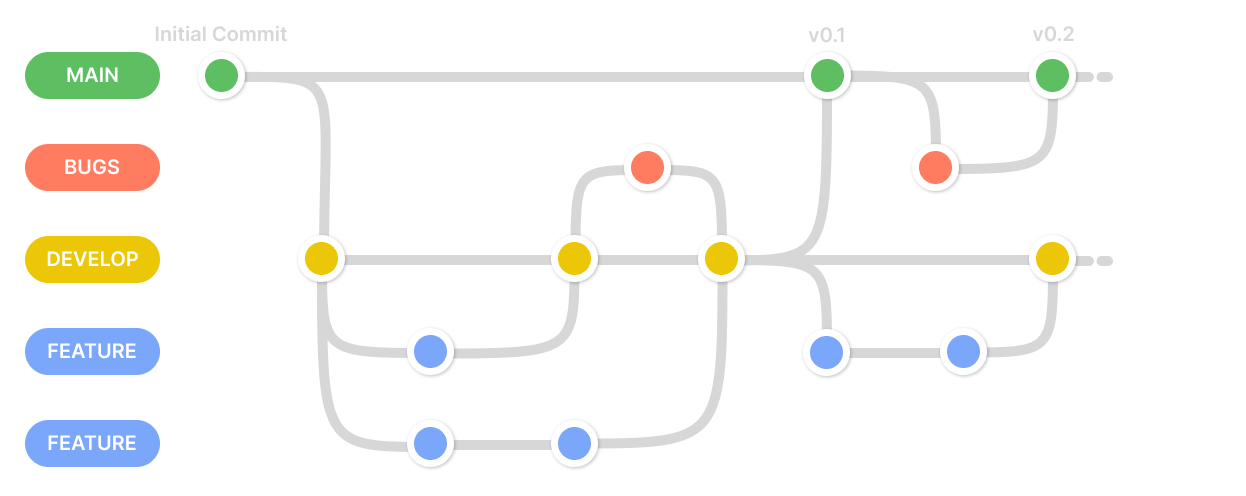
Pel que fa al cicle de vida d’una història d’usuari, aquesta seguirà les fases que ens determina la metodologia *Scrum* al *Taiga*: Quan una història creada s’assigna a un *sprint* determinat es defineix com a nova (*New*). Un cop s’assigna a un membre de l’equip i aquest comença a treballar en alguna de les tasques de la història es diu que està en progrés (*In progress*) i un cop s’han acabat totes tasques la definim com a preparada per a ser testejada (*Ready for test*). Quan es comprova que la funcionalitat és vàlida i es tenen en compte tots els criteris d’acceptació, llavors considerem que la història s’ha tancat (*Closed*), i en conseqüència el seu cicle de vida acaba.

## Gestió de versions

Per tal de gestionar el control de versions del codi de la nostra aplicació, hem utilitzat el servidor *GitHub*, el qual utilitza git. En aquest servidor, hem creat una organització anomenada com el projecte, GreenWheel. Dins d’aquesta, hem creat dos repositoris, un per al *frontend* de l’aplicació (l’aplicació mòbil) i un per al *backend* (l’*API* i la pàgina d’administració).

Els dos repositoris funcionen de manera molt semblant i, per tant, a partir d’ara, se’n parlarà conjuntament. Els repositoris estan formats per diferents branques, les quals, ens ajudaran a tenir un millor control dels repositoris. Cadascun, tindrà:

* **Branca main:** Branca la qual estarà preparada per a fer *releases* de l’aplicació i on cal que el codi estigui polit. En pujar en aquesta branca, s’aplicarà l’acció de CI/CD del codi i es muntarà el *Docker* en l’ambient de producció. En aquesta branca, només podrà fer *merge* el responsable de l’*sprint*, el qual tindrà la funció de comprovar que la branca de la qual s’hereta el codi funciona correctament i no causa problemes.
* **Branca Dev:** En aquesta branca el codi serà estable, tot i que funcionarà en entorn de desenvolupament. Serà la branca central del projecte sobre la qual s’aplicaran els canvis que funcionen de les branques. Un cop es pengi a la branca, s’executarà el CI/CD de producció.
* **Branques d’històries:** Cada història serà declarada com una *issue* de tipus *feature*, la qual cosa farà que quan es vulgui implementar una història d’usuari es faci una branca lligada a la *feature*. Tot el desenvolupament de la *feature* es farà a través de la branca, així com el testing. Un cop finalitzada la seva implementació, s’ajuntarà amb la branca *dev* i es donarà la feature com a completada. Per convenció, el nom d’aquestes branques serà: *feature-<num\_issue>-<nom\_historia>*. *Exemple: feature-#35-login\_raco.* Durant el desenvolupament de la branca és possible que es depengui d’alguna altra feature (encara que les històries han de ser el màxim independents possibles). En aquest cas, es farà *merge* de la branca *dev* en cas que hagi estat finalitzada, i merge de la branca de la història en última instància.
* **Branques de correcció d’errors:** Funcionaran semblants a les branques de features o històries d’usuari. Aquestes, s’obriran per a solucionar problemes de la branca *dev* o de la branca *main*, depenent de si l’error trobat ja es troba en la branca main o pel contrari només està detectat amb les noves versions de la branca *dev* i, per tant, no afecta la versió de producció. Si afecta la versió de producció, caldrà fer *merge* de la branca tant a producció com a *dev*, per tant, no s’esborrarà la branca al fer *merge*. Per convenció, el nom d’aquesta branca serà: *bug-<titol\_issue>*. *Exemple: bug-#135-Error-Al-Entrar-Al-Perfil.*



Per altra banda, caldrà mantenir la coherència dels *commits*. Per a fer-ho, farem el conveni següent: Per cada tasca realitzada, actualitzada o creada, es posarà el número de la tasca amb el seu títol i a continuació la descripció dels canvis realitzats tal com es mostra a continuació:   
*Tasca <#><titol\_tasca>: <descripcio\_canvis> \n Tasca <#><titol\_tasca>: <descripcio\_canvis>.*   
*Exemple: Tasca #35 Login Usuari: S’ha afegit el camp contrasenya i la seva validacio.*

Per últim, el directori *root* de cada repositori, contindrà un fitxer anomenat *Readme* on caldrà que estigui explicat tot el necessari per a executar el codi en local de forma correcta, així com una descripció de què s’hi trobarà i de com està organitzat el projecte. No caldrà que cada directori del projecte tingui aquest fitxer detallant el que es troba al directori, però si s’aconsella fer-ho per als directoris que puguin resultar més confusos.

## Comunicació equip

La nostra comunicació és diària, ja que a diari surten dubtes o inquietuds que anem parlant pels nostres canals de comunicació. Ademés, també fem reunions de tot l’equip de 3 a 4 cops per setmana, les quals, cada cop intentem que durin menys, apropant-nos al concepte de daily de 15-20 minuts de reunió, i seguidament, ens dividim per petits grups per tal de seguir amb la programació de l’aplicació. En aquestes reunions,plantegem les següents questions:

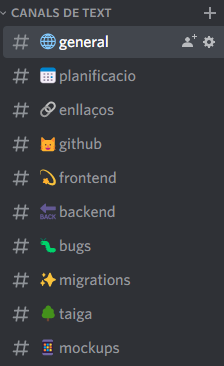
* Què he fet des de l’última reunió: On cada membre del grup explica quins progresos ha fet des de l’última reunió.
* Què té pensat fer fins a la pròxima reunió: On cada membre del grup explica que te planificat en els pròxims 3-4 dies i si necessita qualsevol cosa dels altres.
* Quines dificultats s’ha trobat o preveu que trobarà: Es descriuen aquelles dificultats que s’han trobat o que es preveuen trobar i que poden ser elements bloquejants del projecte, per tal de poder-los discutir i trobar-los una solució.

El nostre equip té dos canals de comunicació principals:

* Un servidor de Discord on tenim classificats diversos canals de veu i de text per a diferents assumptes, com un xat exclusiu per a parlar del *backend*, el *frontend*, la planificació de les tasques, enllaços d’interès, entre d’altres. Tots els membres del grup estem familiaritzats en emprar aquesta eina per a comunicar-nos, que ens ajuda a realitzar les reunions, compartir pantalla, compartir enllaços i avisar-nos d’errors o canvis rellevants que s’hagin realitzat al projecte. A més a més, comptem amb dos bots, un que ens notifica pel canal pertinent dels canvis a taiga i un altre dels canvis a github.
* Un grup de *WhatsApp,* que utilitzem per a comunicar-nos de manera més informal i per a resoldre dubtes puntuals que poden sortir al dia a dia. Ens ajuda a acordar les hores de les reunions i a avisar d’algun inconvenient que hagi sorgit a última hora.

De tots dos, el que fem servir més és Discord, ja que, com que tenim els canals de text classificats per temes, ens és més fàcil de deixar constància de les tasques que ens van sortint cada dia.

Aquest discord, està distribuït amb canals de text i canals de veu. Pels canals de text trobem la següent distribució:

* General: Canal per on ens comuniquem normalment per tal de parlar o discutir alguna cosa general de l’aplicació, així com per fer avisos i comunicar-nos.
* Planificació: És un canal pel qual parla l’sprint master i on va fent indicacions de com va l’evolució del sprint, que falta per fer i com ens hauriem de planificar de manera general per als pròxims dies.
* Enllaços: En aquest canal s’hi troben molts enllaços d’interès, com apis, llibreries, exemples de codi, enllaços a coses importants del projecte, entre d’altres. Només està permès enviar enllaços, per evitar saturar el canal.
* Github: En aquest canal només pot parlar un bot que tenim de discord, el qual, cada cop que hi ha un nou commit en algun dels dos repositoris, ho comunica per aquest canal, així tenim un log dels diferents logs que s’han fet de manera cronològica, així com també podem saber quan algú ha fet commit.
* Frontend: Canal on es parlen coses sobre el procés de programació de l’aplicació mòbil (en flutter) i es pregunten dubtes específics de frontend. També es comparteixen consells i qualsevol cosa que pugui ser d’utilitat a l’hora de programar frontend. Figura 13. Captura del server de Discord
* Backend: El canal de backend funciona de manera molt semblant al de frontend, amb la diferència en que els dubtes i el motiu pel qual s’escriuen missatges està relacionat amb la part de backend (django) de la nostra aplicació.
* Bugs: En aquest canal es on es reporten els bugs de manera ràpida, on indiques quan t’ha passat i imatge del error o de que ha passat. Un cop reportat aquí, qui la vagi a resoldre, ja la introduirà a taiga de manera formal.
* Migrations: Canal pel qual es notifica si cal fer alguna migration o actualitzar les llibreries del front, ja que el codi actualitzat a dev ho requereix, per tant, quan l’agafin, cal que ho tinguin en compte.
* Taiga: En aquest canal rebem els avisos dels moviments de tasques en el taiga, gràcies a un bot que s’encarrega de fer les notificacions.
* Mockups: Canal on anem passant captures de les interfícies que anem fent per a veure que els sembla als companys i puguin valorar-les. També ens serveix per a anar tenint un petit repositori de captures del progrés de l’aplicació.

Per la part dels canals de veu, tenim 3 canals bàsics, que ens serveixen per a dividir-nos en grups a l’hora de treballar.

## Gestió qualitat

La qualitat del nostre software, més concretament del codi, la durem a terme fent ús de les eines d’anàlisi de codi que proporcionen els IDEs de la companyia JetBrains, que són els que farem servir per a programar la nostra *App* (ja sigui *PyCharm* o *WebStorm*).

Aquestes eines que contenen els IDEs detecten i suggereixen correccions al codi del projecte. Ressalten *bugs*, *dead code*, problemes d’ortografia i milloren l’estructura general del codi. Els problemes estan ressaltats de diferent manera de forma que nosaltres, els programadors, podem distingir entre els problemes més i menys importants.

Per altra banda, com que tenim accés a *GitHub for Students* a partir del nostre correu de la UPC, alguns dels membres del grup farem servir *GitHub Copilot*, que és una eina que recomana codi ben estructurat a partir del que s’està editant en programar. Això ens ajudarà a estalviar temps i a evitar alguns *bugs*.

A més, la nostra idea és afegir algun analitzador de codi a l’hora de penjar a la branca dev i a la branca main del *git* de manera en que, per cada *commit*, es faci la comprovació i ens pugui donar els resultats.

Tots els companys estem treballant a una, i veiem els codis dels companys contínuament quan ho publiquen a les seves branques, i ens anem avisant entre nosaltres si veiem quelcom que no és correcte o podria tenir millor qualitat. La nostra idea és fer el codi més entenedor possible i dividit per funcions, evitant en tot moment repeticions de blocs de codi que puguin dificultar la lectura del codi. També el compromís de l'últim sprint és comentar les diferents funcions utilitzant l’estructura de *javadoc* (no *javadoc* en si, si no un format de text *markup* que segueixi una estructura semblant per a llenguatge *Python*).

Pel que fa la imatge de les vistes i comprovar que el que es desenvolupa al *Frontend* queda clar i està acordat per tothom, fem servir l’eina de *Figma*. Del model de totes les pantalles l’equip es reuneix i acorda tots els widgets, es numeren, s’anoten i s’assignen. D’aquesta manera es redueix l’acoblament, la repetició de codi i els altres membres poden reutilitzar els widgets ja desenvolupats.

## Gestió de proves

Per a fer els tests del nostre sistema, ja siguin els unitaris o els d’integració, els podrem fer manualment amb les eines que ens proporcionen *Django i Flutter*. El *testing* ens permetrà veure si el nostre codi funciona (des de casos trivials fins a casos extrems) i que, per tant, les funcionalitats s’executen correctament. També ajudarà a comprovar que el codi ja existent, no pateixi cap error a causa dels canvis realitzats en l’últim commit.

Pel que fa a la integració continua (CI), farem servir l’apartat d’*Actions* dels nostres repositoris de *GitHub* un cop els tests estiguin fets. Aquesta ens permet a cada *push* executar els tests unitaris i d’integració que haguem programat abans tant pel *frontend* o *backend*.

Els tests de cada història d’usuari (per tant, de cada branca), cal que es realitzin abans de fer *merge* a la branca *dev*, per tal que la branca *dev* pugui executar els nous tests d’integració contínua i el codi implementat fagi l’esperat. Aquests esperem desenvolupar-los a l’últim sprint i així tenir ja una integració continua real.

A part dels tests automàtics, també farem moltes proves manualment, per tal de veure el correcte funcionament i que el programa fa l’esperat, així com que compleix tots els criteris d’acceptació de les històries d’usuari. Aquestes proves, es faran amb dades de prova tant en la branca mentre es realitzen les funcionalitats com un cop acabada la funcionalitat utilitzant les dades locals. També es faran proves un cop es publica a la branca *dev*, en les quals s’utilitzaran les dades que tindrem a la base de dades de desenvolupament. Finalment, es faran proves un cop el codi arribi a producció i on les dades ja seran reals. En cas que en alguna d’aquestes proves es detecti algún error, es farà el *rollback* per tal de no tenir codi erroni, i s’intentarà d’arreglar l’abans possible la incidència, tornant a fer tot el procés de revisió de codi.

## Gestió configuracions

Al principi del projecte, ens vam plantejar utilitzar *Jenkins*, la qual és una eina molt potent per tal de desplegar codi a servidors. Tot i així, l’hem acabat descartant degut a que requereix d’un servidor amb uns requisits mínims bastant alts, i el qual ens era impossible aconseguir de manera gratuïta i senzilla.

**Github Actions**

Per tant, finalment vam decidir que, per tal de fer el desplegament de les nostres aplicacions, utilitzem *GitHub Actions*, el qual permet fer *scripts* d’automatització que s’executen quan es fa un *push* a la branca seleccionada. Aquesta eina és propia de *Github* i, per tant, està totalment integrada amb aquest, fent que funcioni molt bé quan s’utilitza aquest repositori. Conté moltes plantilles les quals ajuden a muntar els *scripts*, fent que la corba d’aprenentatge per tal d’utilitzar aquesta tecnologia no sigui tant alta.

Per tal de desplegar la nostra aplicació, tindrem un *script* que contindrà les ordres per tal que s’executin els tests d’integració contínua, la revisió de codi i el desplegament al servidor d’*Amazon Web Services* dins d’un Docker.

Hi haurà un únic *script* de desplegament, el qual depenent de si es fa *commit* a *dev* o a producció agafarà les variables d’entorn que pertoquin i desplegarà al compte d’*AWS* de preproducció o de producció.

El nostre objectiu és tenir preparada la integració contínua per al segon *sprint*, per tal de poder treballar amb l’arquitectura al complet.

Als primers sprints treballem en local fins que no hi hagi el deployment fet, i en aquest entorn local amb el fitxer *.env*, tindrem aquelles variables d’entorn que, en pre producció i producció es posaran automàticament, però que en local cal que tinguem per tal de poder utilitzar l’aplicació. En aquestes configuracions hi haurà totes aquelles claus a *APIs*, connexions a bases de dades i *url* que puguin canviar (com per exemple la connexió des del *frontend* a la nostra *API*).

## Integració amb altres grups

Des d’un bon principì, vam tenir molt clar quin servei i funcionalitat volíem. Era millor agafar una funcionalitat del servei de contaminació i pol·lució de l’aire que el d’actes culturals, degut a que era una utilitat que realment aportava valor a la nostra aplicació, a diferència dels actes culturals, la qual no tenia una relació directa o indirecta sobre la nostra aplicació.

Per aquest motiu, des del primer dia ens vam posar a parlar amb els dos grups per tal d’aconseguir obtenir la contaminació i nosaltres oferir el nostre servei de carregadors per a cotxes i bicicletes al grup de cultura i festes.

Al principi, no estava del tot clar que havíem d’oferir, i en primer moment vam plantejar la possibilitat d’oferir el traçat d’una ruta sobre un mapa. De seguida vam valorar que no era una bona idea, ja que segurament utilitzariem tecnologies diferents, i el millor era oferir-los dades a partir de la nostra api que els hi dongués valor.

La interacció amb els altres equips de PES sobre oferir i rebre els serveis ha continuat sent mitjançant l'aplicació WhatsApp .

Encara se sosté que en cas que aquesta aplicació falli, llavors la comunicació es realitzarà mitjançant el correu de la universitat o Discord.

Periòdicament, s'han realitzat i rebut peticions dels dos grups amb els quals estem negociant i és on es manté la constància de la negociació de les funcionalitats que han esdevingut en el projecte.

Durant aquest segon sprint hem acabat de concretar quins endpoints podran accedir, encara que ja està concretat el que els oferim i com els hi oferirem, tal i com podem veure descrit en l’apartat de la documentació dedicat a això.

Per últim, si valorem la comunicació amb els dos grups, podem dir que amb el grup relacionat amb la cultura i festes a Catalunya, no hem tingut cap problema per a fer-ho i ens ha contestat al moment, fent la comunicació fluida i fàcil, ademés de com que venia a classe, si no eren coses urgents, es podien parlar de manera presencial. Respecte el grup de contaminació, hi ha hagut comunicació amb més freqüència que el primer sprint i Inception, on hem pogut establir finalment els serveis, i on els últims dies ens han explicat.

## Gestió de bugs

Els *bugs* del nostre sistema els gestionem a través de l’apartat d’*issues* de cadascun dels nostres 2 repositoris. Emprant aquest sistema ens permet crear una *issue* de *GitHub* amb l’etiqueta *bug* i es descriu quin és el problema i quin membre de l’equip s’ha d’encarregar de solucionar-lo, que normalment serà el mateix que s’ha encarregat de desenvolupar aquella part on hi ha el *bug*.

Per altra banda, en *bugs* de més gran importància es proporcionarà un informe a través del canal de text de *bugs* del nostre servidor de Discord, on es detallarà el problema més a fons per tal de comunicar-lo a tots els membres del grup perquè el tinguin present.

També documentem els diferents errors utilitzant l’apartat issues de *Taiga*, el qual és molt complet i és un bon mètode per a tenir tots els *bugs* documentats i apuntats, així com per assignar-los i mantenir un control de l’evolució d’aquest.

La resolució de les incidències serà una prioritat per al nostre equip on, en el moment en que es detecti, s’intentarà de solucionar el més ràpid possible, sobretot si pot afectar altres parts de l’aplicació. Les incidències estaran etiquetades amb tres tipus de prioritats, les quals designaran l’afectació que té a l’aplicació i si només afecta visualment o és algo que no deixa seguir. Les tres prioritats que tenim son:

* Alta: La incidència impossibilita l’evolució de l’usuari utilitzant l’aplicació o afecta altres sistemes, fent que no puguin seguir amb la seva evolució de forma nominal. Aquestes incidències cal que siguin tractades de manera urgent i el programador assignat deixi de fer l’evolutiu per tal de resoldre-la.
* Mitjana: La incidència dificulta l’evolució de l’usuari utilitzant l’aplicació, fent que es quedi encallat en algun punt i que hagi de tornar endarrere o que no pugui completar una funcionalitat. També pot ser categoritzada amb aquesta etiqueta si provoca errors en altres apartats de l’aplicació, sense ser bloquejants. En aquest cas, caldrà que el programador ho resolgui l’abans possible, però sense l’urgència que tindriem en cas que fos alta.
* Baixa: La incidència només té afectacions visuals, les quals no impossibilitant l’ús de l’aplicació, però sí que el fan més complicat o no el fan agradable. En aquest cas no afecta a cap altre sistema de l’aplicació. També és possible que siguin incidències que podrien estar etiquetades com a mitjanes però que només es donen en casos molt concrets i localitzats. En aquest cas, es resoldran abans de que acabi l’sprint, però i l’abans millor, però sense la necessitat que el programador hagi de deixar de fer el seu evolutiu per solucionar aquest error.

Aquestes incidències s’etiqueten al *Taiga* quan estan en progrés, tancades, rebutjades, preparades per a testejar, entre d’altres.

## IDEs

Per tal de realitzar les aplicacions de la nostra arquitectura, hem decidit utilitzar els *IDE*sde la marca de *software JetBrains*, concretament *PyCharm* per a *backend* i *IntelliJ* i *Android Studio* per a *Flutter*.

La desenvolupadora de software *JetBrains* ofereix llicència d’estudiant per a tots els seus *IDE’s*, la qual desbloqueja l’accés a totes les seves tecnologies i a tots els seus avantatges.

L’aspecte positiu és que al generar fitxers de configuració, quan la persona que crea el projecte el publica al repositori, els altres integrants del grup ja tenen el projecte configurat i preparat per a començar a programar.

Un altre dels avantatges d’aquests *IDEs* és la quantitat d’eines que porten incorporades, com per exemple *intellisense*, la qual fa recomanacions de codi, un analitzador de codi molt potent, entre altres.

Altres característiques que trobem dels *IDE*s de *JetBrains* és les bones gestions que tenen de tecnologies com *git* o *Github*, les quals son molt completes i permeten gestionar tot el repositori des del *IDE* i sense necessitat d’haver d’accedir a *Github* o a altres plantaformes. Una altra que també ens ha estat molt útil, és el sistema gestor de base de dades que incorpora *PyCharm*, el qual ens permet examinar i/o modificar la base de dades des del mateix *IDE* i sense necessitat de tenir un altre programa per tal de fer-ho.

Per altra banda, *IntelliJ* i *Android Studio* ofereixen una sèrie d’eines molt útils a l’hora de desenvolupar en *Flutter*, com per exemple el tenir l’emulació del dispositiu mòbil integrat, així com inspectors de codi específics per a *Flutter* i *Dart*, els quals permeten veure com s’ha construït els components i veure quin espai ocupen, perquè s’està col·locant allà l’element, etc. També ens ofereix eines d’anàlisi del rendiment com el *Profiler*, el qual ens dóna, a una granularitat de variable, el cost que té cada operació realitzada en el codi i quin impacte té sobre el total de càlcul, molt útil a l’hora d’analitzar les càrregues lentes.

Per últim, també farem ús del *plugin* que incorporen els *IDEs* de *GitHub copilot*, el qual és un recomanador de codi desenvolupat per *GitHub* que té una gran potència i fa recomanacions de codis complets a partir d’un comentari. Aquest *plugin*, ens ajudarà a poder implementar més ràpidament el codi, tot i que sempre revisant el que ens ha escrit i modificant-ho perquè s’adapti a les nostres necessitats.

## Llenguatges

Per tal de programar les nostres aplicacions hem dividit el nostre codi en *backend* i *frontend*. Seran dues tecnologies ben diferenciades.

Al *backend*, que s’executa en el servidor, hi tindrem un *Docker* amb el *framework Django*, el qual està programat en *Python*, un llenguatge interpretat. Aquest *framework* ens permet crear una *API* a través de la seva llibreria *Django Rest Framework*, la qual ens ofereix moltes possibilitats per tal de realitzar la nostra *API* desitjada.

Per altra banda, tenim el *frontend*, que s’executa en el dispositiu de l’usuari. En el nostre cas, hem decidit fer que l’aplicació s’executi exclusivament en dispositius *Android*, i hem escollit *Flutter* com a *framework* per a desenvolupar-la. Tot i que està dissenyat per a sistemes multiplataforma i no natius d’*Android*, hem triat aquest *framework* per la quantitat de documentació existent i per la simplicitat que suposa *Dart* (el llenguatge principal del codi d’aquest *framework*) respecte a llenguatges que havíem treballat anteriorment a altres classes. Durant la fase d’*Inception* vam pensar emprar *React Native*, un *framework* desenvolupat per *Facebook* i realitzat usant *JavaScript*, ja que inicialment està pensat per a sistemes *Android* natius. Però tot i tenir una gran comunitat amb milers de llibreries vam acabar decantant-nos per *Flutter* pel problema que suposa *React Native* amb la incompatibilitat entre versions.

El *frontend* de la pàgina d’administració utilitzarà la capa de vista del *framework Django*, on podrem utilitzar *jinja* per a utilitzar codi dins de l’*HTML*.

A continuació, es descriuran algunes de les convencions a l’hora d’escriure codi que seguirem, per tal de generar un estàndard i que el codi sigui més fàcil de llegir.

### 4.11.1 Variables

Les variables caldrà que s’escriguin seguint el conveni *Lower Snake Case*, el qual especifica que totes les lletres han de ser minúscules i les paraules han d’estar separades per ‘\_’.

Exemple: comptador\_de\_paraules.

Les variables cal que siguin descriptives, i en cap cas siguin una sola lletra, exceptuant que s’usi de manera temporal en un bucle o semblant. El text de les variables cal que estigui escrit en anglès, utilitzant un llenguatge entenedor i un vocabulari senzill.

Per últim, caldrà que les constants segueixin el conveni *Upper Snake Case*, la qual determina que totes les lletres han de ser majúscules i han d’estar separades per un ‘\_’.

Exemple: PI\_NUMBER

### 4.11.2 Funcions / mètodes

Les funcions caldrà que s’escriguin seguint el conveni *Lower Camel Case*, el qual especifica que totes les paraules menys la primera, han de començar en majúscula i en cap cas, se separaran les paraules amb símbols.

Igual que en les variables, cal que els noms siguin tan clars com sigui possible i escrits en anglès, especificant l’acció que realitzen, si és el cas (*getter* o *setter*).

Exemple: getCarByUserId(param1, param2).

Cal que cada funció o mètode tingui un comentari seguint el conveni de *javadoc*, de què fa la funció o mètode, explicació dels paràmetres i del que retorna. Aquest comentari ha d’estar escrit en català. En cas necessari, durant el codi, també poden existir comentaris de línia.

Per últim, caldrà que les claus de les funcions, condicionals, bucles i altres, estiguin en línia en el cas d’obrir i estiguin en un salt de línia en cas de tancar.

Exemple:

if(i==1){

print(“i es igual a 1);

}

### 4.11.3 Classes

El nom de les classes cal que s’escriguin seguint el conveni *Upper Camel Case*, el qual indica que cal que la primera lletra de cada paraula sigui majúscula.

Les classes estaran ordenades de la següent manera:

1. Propietats
2. Constructor
3. *getters*
4. *setters*
5. altres mètodes

En el cas de les classes relacionades amb rest s’ordenaran els mètodes de la següent manera:

1. get
2. post
3. put
4. delete

### 4.11.4 Fitxers i directoris

Els noms dels fitxers (que no siguin classes) caldrà que estiguin escrits utilitzant el conveni *Lower Snake Case*, el qual, com ja s’ha explicat a dalt, consisteix a escriure totes les paraules en minúscula, separades per una barra baixa. Finalment, cal que tinguin una extensió. Si són classes, s’utilitzarà la mateixa nomenclatura que la classe.

Exemple: cotxes\_usuari.py

Per altra banda, els noms dels directoris caldrà que estiguin escrits utilitzant el conveni Lower Camel Case, el qual defineix que totes les paraules han de començar en majúscula excepte la primera.

Exemple: aixoEsUnDirectori

Per a qualsevol altre element no mencionat en aquest apartat, se seguirà el conveni de *PEP8*[**[5]**](#2xkdfceq17vf) per a *python* i el conveni de *Mozilla*[**[6]**](#vm3hnw8k844g) per a javascript.

## Gestió base de dades

La base de dades serà *PostgreSQL* tal com hem comentat anteriorment i estarà com un servei dins *d’Amazon Web Services, tenint-ne una* per a producció i una altra per a preproducció. Per a treballar en local, utilitzarem un *Docker* amb una base de dades *PostgreSQL* amb la mateixa versió que la que es trobarà a producció, però que serà local, podent provar així valors estranys i dades sense sentit sense fer malbé les dades autèntiques de producció. Aquest *Docker* es desplegarà amb un *Docker composer*, que es troba al repositori del *backend*.

Per tal de construir la base de dades, usarem les *migrations* de *Django*, les quals ens permeten poder fer canvis en el model de la base de dades fàcilment, a més de proporcionar-nos tota la potència dels models de *Django* per a crear bones estructures de bases de dades.

L’script que fa el desplegament se n’encarregarà d’executar les comandes per a fer les *migrations* cada vegada que pugi a producció o preproducció, per assegurar-se que la base de dades conté tots els camps que es necessiten.

En local, caldrà executar-ho en cas que algú hagi fet alguna modificació, que en aquest cas, ho haurà notificat pel xat de discord anomenat *migrations* i posant el següent missatge al comentari del *commit*: Migrations: (nom\_taula1, nom\_taula2…).

# 

# Descripció tècnica

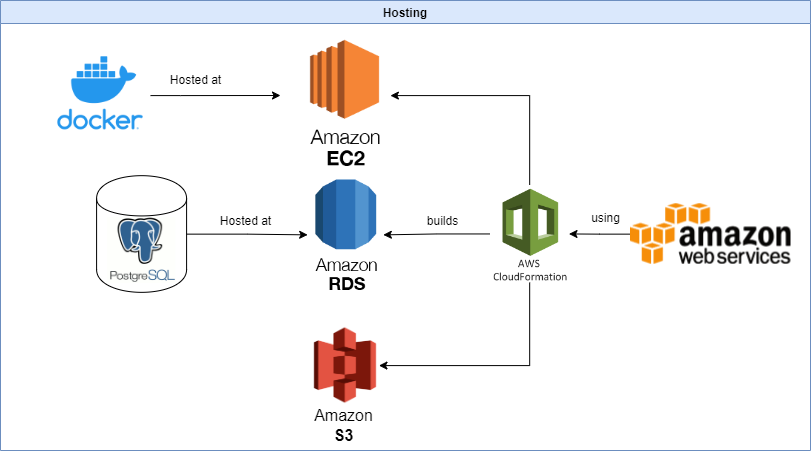
## Arquitectura

### 5.1.1 Diagrama arquitectura física

A continuació mostrem el diagrama de l’arquitectura física que tindrà el nostre projecte. En aquest apartat expliquem tot el que utilitzem i el perquè d’emprar un servei i no un altre.

## 

#### **Hosting**



Hem estat debatent quina plataforma utilitzar per a allotjar la nostra web. Han anat sortint diferents llocs com *Heroku*, *Virtech*, *OVH* i *AWS* i després d'investigar-los, hem arribat a la següent conclusió: Heroku passa a ser de pagament en breu i el compte gratuït ofereix molt poca cosa, Virtech tampoc ens proporciona tot el que volem, ja que són servidors molt limitats i OVH hem vist que està molt bé, però és de pagament. Així que finalment utilitzarem AWS per la part d'allotjament, a continuació expliquem per què.

Hem decidit emprar diferents serveis d*’AWS*, ja que d'aquesta manera podem tenir tant el servei d'emmagatzematge i d'allotjament en una mateixa plataforma. A més, una companya del grup a la feina fa servir diversos serveis d'Amazon i, per tant, ja tenim a algú que coneix i sap fer anar alguns dels serveis d'*AWS*. També ha ajudat a decidir-nos per usar Amazon el fet que per la infraestructura que necessitem nosaltres, el free tier que ofereix, ja ens serveix. Així que hem creat un compte a *AWS* a la qual podem accedir tots els components del grup. Hem investigat quins són els serveis que més utilitat ens poden oferir, aquests són els següents:

* **Amazon EC2:** Per allotjar la nostra web farem servir el servei d'Amazon EC2, aquest ens permetrà tenir un servidor allotjat al *Cloud*. Per configurar una instància d'EC2, entre diverses coses, podem definir unes regles d'entrada i de sortida al servidor i també podem decidir sobre quin sistema operatiu corre la màquina. En el nostre cas, hem decidit que el sistema operatiu que farem servir serà Ubuntu, el qual ens permet poder desplegar dockers fàcilment.
* **Amazon RDS**: Utilitzarem el servei d'Amazon *RDS* (*Relational Database Service*) per a configurar una base de dades relacional en un entorn aïllat al *Cloud* d'*AWS*. Per a fer-ho crearem una instancia de base de dades *PostgreSQL* a la qual ens podrem connectar a través de *TCP/IP.*
* **Amazon S3:** També farem ús d**’**Amazon *S3* (*Simple Storage Service*), el qual és un servei d’emmagatzemament d’objectes. Ens servirà per guardar totes les imatges que pugin els usuaris siguin del perfil o dels punts de càrrega privats o les bicicletes.
* **AWS CloudFormation**: Tots aquests serveis explicats els crearem a través d’*AWS* *CloudFormation*. Aquest et permet crear una plantilla la qual descriu tots els recursos d'AWSque es vulguin fer servir i *CloudFormation* s’encarrega de l'aprovisionament i la configuració dels recursos descrits.

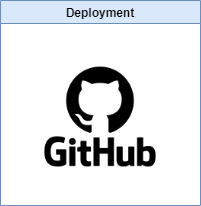
Per a crear un contenidor on corri la nostra aplicació *Docker*:

* **Docker:** Hem decidit també utilitzar-ho, ja que si més endavant necessitem replicar la nostra API ho podrem fer fàcilment gràcies a Docker. En un principi tindrem un únic container i, per tant, ho podríem desplegar directament al servidor, però per un tema d'escalabilitat hem decidit emprar un Docker, pel fet que, en cas de necessitar-ho, podríem muntar un sistema mitjançant *kubernetes*[***[1]***](https://docs.google.com/document/d/1IpiThHwlIW9uYLJfOzsr2eL825wHIHRnZoTb9bvjIk0/edit#bookmark=id.p9yra1ry95o5)o *docker swarm*[**[2]**](https://docs.google.com/document/d/1IpiThHwlIW9uYLJfOzsr2eL825wHIHRnZoTb9bvjIk0/edit#bookmark=kix.f5ga4o917ufl), per tal d'ampliar la capacitat de la nostra API.

Farem servir una base de dades relacional, ja que tots estem més familiaritzats amb les relacionals en comparació amb les no relacionals, a més no creiem que tinguem tantes dades com perquè no ens sigui viable emprar una relacional. També és veritat que *Django* està més treballat per a suportar bases de dades *SQL* i així podrem beneficiar-nos de totes les facilitats que ens doni *Django*. Hem decidit utilitzar el llenguatge *PostgreSQL* per implementar la nostra base de dades:

* **PostgreSQL**: Amazon *RDS* admet instàncies de base de dades que executen versions de diferents llenguatges i entre ells es troba *PostgreSQL*. Del ventall que admet hem triat aquest, ja que tots l’hem utilitzat algun cop.

#### **Desplegament**

Per tal de desplegar el nostre codi a l’aplicació, teníem varies opcions que ens eren vàlides i que ens podien servir per a fer la construcció, testejar i desplegar el servidor a *Amazon Web Services* en un *Docker*.

Una de les opcions, era utilitzar *Jenkins*, el qual és un *software* molt eficaç per a portar a terme les tasques que volíem, permetent fer el *script* d’automatització en *groovy* (un llenguatge molt semblant a Java, el qual inclou algunes millores), a més d’integrar-se molt bé amb els repositoris *git* (sobretot *GitLab*). El gran problema d’aquest software, és que no es pot fer servir sense tenir-lo allotjat en un servidor propi, el qual ha de tenir uns mínims requisits per tal que pugui funcionar correctament. Aquest va ser el punt que ens va ajudar a decidir que aquest no seria el software que empraríem per a desplegar la nostra arquitectura.

Vam estar analitzant i fent proves de concepte amb altres tecnologies com *CircleCI*, *travisci* o *TeamCity*, aquesta última desenvolupada per *JetBrains*, els desenvolupadors dels IDEs que utilitzem i del qual tenim llicència. Però o eren de pagament o necessitàvem adquirir un servidor per tal de poder-lo usar.

Finalment, ens vam quedar amb l'opció de *GitHub Actions*, el qual no requereix cap servidor, sinó que ve integrat amb *GitHub*. Aquesta eina, ens permet fer l’automatització de processos on el *trigger* és quan una branca concreta és actualitzada, permetent executar els passos que vulguis, com per exemple analitzar i testejar, *buildejar* i desplegar a *Amazon Web Services* el *Docker* amb el *backend* de la nostra aplicació.

En el nostre cas, ens agradaria tenir dues infraestructures, una de producció, on estigui l’última versió productiva i d’on l’equip que absorbeix de la nostra *API*, pugui demanar dades sense problemes i un altre de desenvolupament, on un cop hi ha una característica acabada, es pugui provar en aquest entorn de desenvolupament per tal de comprovar que funcionarien en l’entorn productiu, ja que tindran la mateixa arquitectura.

En tenir llicència d’estudiant de *GitHub*, tenim accés a totes les seves possibilitats i a utilitzar tot el que ofereix *GitHub* *Pro*, permetent-nos utilitzar tota la potència de *GitHub Actions*, així com també per poder-ne aprendre.

#### **Server**

Respecte a la nostra aplicació servidor o *backend*, teníem moltes possibilitats. Necessitàvem una tecnologia que ens permetés desenvolupar una *API* amb bastants *endpoints*, organitzar i reaprofitar codi, connectar-nos a bases de dades relacionals i realitzar-hi migracions fàcilment, que la corba d’aprenentatge no fos excessivament gran i tenir una *API rest* i una web amb part visual (per a plantejar-hi la pàgina d’administració). La majoria de l’equip no hem programat mai en web, així que qualsevol tecnologia que escollissim necessitaria un temps d’aprenentatge pels membres de l’equip. Els que ja treballaven en tecnologies web, van presentar les diferents opcions que teníem i seguidament, després de saber els pros i contres de cadascuna, vam elegir.

Com a opcions per tal de poder implementar la nostra tecnologia teníem *Django*, *Spring Boot*, *Express*, *Laravel* i *Ruby on Rails*. Existeixen moltes altres alternatives, però aquestes són les que ens van semblar més encertades.

Finalment, després de documentar-nos[**[3]**](https://docs.google.com/document/d/1IpiThHwlIW9uYLJfOzsr2eL825wHIHRnZoTb9bvjIk0/edit#bookmark=id.x9xr74la5vk8) i estudiar cadascuna de les tecnologies, vam decidir utilitzar *Django*, el qual ens ofereix tot allò que necessitem per al nostre projecte, i s'adapta perfectament a allò que volem aconseguir.

*Django* ens permet desenvolupar una *API rest* completa i complexa gràcies a la seva llibreria *Django Rest Framework*, la qual permet crear *APIs* fàcilment, podent jugar amb les dades i transformant-les al nostre gust, creant genèrics i permetent-nos afegir seguretat de manera senzilla. Un altre dels motius pels quals hem escollit *Django* és per la seva gran documentació, la qual és molt completa i està plena d’exemples, a més que la comunitat que té al darrere és molt gran i activa. La interacció amb les bases de dades usant els models de *Django* és molt agradable i ens permet interactuar amb qualsevol mena de base de dades fàcilment, sent independent de la base de dades. Interactuar amb altres *APIs*, amb *Python* és molt senzill i t’ho posa molt fàcil per poder fer grans consultes a *APIs* externes, cosa que necessitem.

Aquest llenguatge segueix el model MVC, encara que quan s’utilitza com a Rest API, es converteix en MV i la capa de negoci es perd. Per aquest motiu la implementarem nosaltres.

Per últim, el fet de permetre’ns poder realitzar la pàgina d’administració a la vegada que l’*API*, és un avantatge, i en el nostre cas, que estem limitats pels servidors, ens facilitava molt les coses.

#### **External services**



Per a desenvolupar la nostra aplicació, a part de les nostres funcionalitats i dades, també necessitem comptar amb aplicacions externes, les quals ens donin tot allò que ens manca o que no podem aconseguir fàcilment.

Les dades, les agafem des de la part del servidor perquè és més segur que fer-ho des de la part del client, ja que per a obtenir-les se sol necesitar algun tipus d’autenticació, fent que si ho fas a la part del client, aquesta autenticació es pugui veure o aconseguir fàcilment. Per això, tot el que hem pogut ho hem obtingut des de la part del servidor, on les dades d’autenticació estan guardades dins del servidor en variables d’entorn.

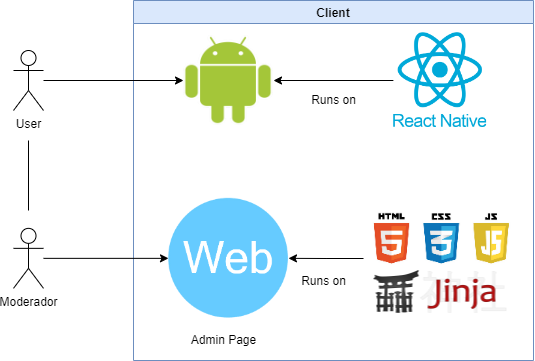
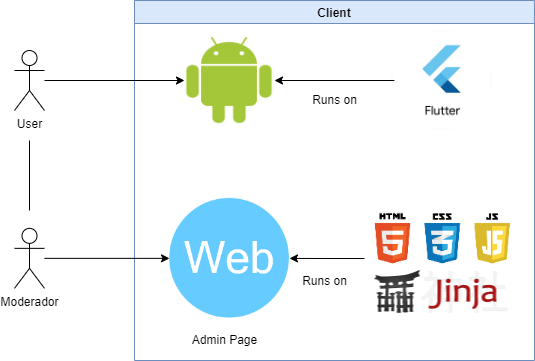
Les aplicacions de les quals obtenim dades o utilitats són:

* **AB&B**: És una aplicació, desenvolupada per un altre projecte del nostre subgrup 12, el qual ens ofereix les dades sobre la contaminació a Catalunya.
* **Dades obertes de Catalunya:** És un repositori de dades i *datasets* actualitzats amb dades sobre Catalunya. En el nostre cas, ens interessa el *dataset* relacionat amb els carregadors per a cotxes elèctrics a Catalunya.
* **Racó Fib:** És el portal d’estudiants de la facultat d’Informàtica de Barcelona, el qual ofereix una API per accedir a les seves dades públiques i les dades de l’usuari loguejat. En el nostre cas, ens interessen les dades de l’usuari loguejat.
* **Google:** *Google* té una gran *API*, de la qual pots obtenir moltes dades de diferents àmbits. Tot i que després en el client obtindrem més dades de *Google*, des del *backend* només ens interessa l’autenticació utilitzant *Google*, la qual es porta a terme mitjançant *OAuth2*.
* ***Socket.io*:** És una llibreria la qual permet enviar esdeveniments entre el *frontend* i el *backend* de manera bidireccional i en temps real. Ens ajudarà a gestionar les notificacions i el xat, de manera que els missatges es rebin instantàniament.

Per últim, tindrem el servei que nosaltres oferirem, el qual constarà de dos *endpoints*. Amb el primer, es podran sol·licitar els punts de càrrega i amb el segon els punts de bicicleta. Aquests *endpoints* podran rebre *queries* com per exemple la quantitat, la ubicació, si es vol per radi o per proximitat, etc.

Aquest servei ens obligarà a tenir sempre una aplicació productiva activa, a la qual puguin accedir sempre que ho necessitin per a utilitzar-la a la seva aplicació.

#### **Client**

La part del client, es divideix en dues aplicacions, l’aplicació per a *Android* i l’aplicació d’administració, les quals estan destinades a públics i dispositius ben diferenciats.

### 

### 

### 

### 

#### **Aplicació Android**

Per un costat, tenim l’aplicació mòbil, la qual només estarà disponible per a dispositius mòbils Android i està enfocada als usuaris que donin ús a la nostra aplicació per al que ha estat dissenyada.

Vam tenir molts dubtes a l’hora d’elegir la tecnologia amb la qual construirem la nostra aplicació per a *Android*. Les opcions que podíem elegir eren moltes, com per exemple *Android* natiu, *Flutter*, *React Native*, *Kotlin* amb *Compose*, entre altres.

Hem de dir que la presentació que van venir a fer de *Basetis* ens va fer replantejar si era millor opció utilitzar *Kotlin* amb *Compose*, ja que ens ho van presentar de manera molt robusta i que de veritat es considerava com una bona opció.

Finalment, a la segona fase d’inception, vam elegir *React Native* pel fet que l’equip de programadors se sentia més segur en aquesta tecnologia (per haver-la usat anteriorment) i vam decidir no arriscar-nos amb una tecnologia inexplorada per nosaltres. *React* ens permet construir l’aplicació i que pugui ser executada nativament per *Android*, així com penjar-la a *Google Play*. A part, ens ofereix escalabilitat, ja que si en algun moment volguéssim ampliar la nostra aplicació a *iOS*, podríem fer-ho sense haver de començar de zero.

Un cop vam iniciar les proves de concepte i a provar react per a desenvolupar aplicacions mòbil, vam veure que ens podia donar molts problemes, ja que vam detectar que era poc sòlid entre versions, i els codis que funcionaven en la versió anterior ja no funcionaven en l’actual, fent que fos molt complicat trobar exemples, documentació correcte o errors, ja que si eren d’altres versions, en la majoria de casos no ens servien. També vam veure que la documentació oficial no era gaire bona a l’hora de programar, i tot el que estava explicat era molt bàsic. Per aquests motius, vam decidir tornar a explorar les possibilitats que ens havíem plantejat anteriorment i, finalment, vam decidir elegir Flutter.

Flutter és un framework escrit en Dart i creat per google per a poder desenvolupar aplicacions multiplataforma de manera senzilla. Compta amb una documentació molt ben feta de part de google, plena d’exemples i consells, així com cursos per a iniciar-se en flutter. Gràcies a la gran informació que aporta, iniciar-se a Flutter se’ns ha fet més senzill i en poques hores d’aprenentatge hem pogut desenvolupar varies pantalles.

A part de la part oficial, flutter també compta amb una gran comunitat al darrera, la qual ha fet llibreries i widgets que ens facilitaran molt més la feina a l’hora de realitzar les tasques.

Segueix una arquitectura MVVM, és a dir, Model-View-ViewModel, la qual ens permet implementar l’arquitectura en 3 capes i separar l’acoplament el màxim possible.

Per últim, al estar desenvolupat per google, que també és el propietari d’android, té tot allò bo d’android, amb facilitats extres que ofereix flutter, fent que, en molts casos, sigui més senzill que treballar amb android natiu.

#### **Aplicació administració**

L’aplicació d’administració és l’altra aplicació del client. Aquesta ha de poder funcionar en web, per tant, necessitàvem elegir les tecnologies que requeriria.

Després de plantejar-nos que utilitzar, vam trobar que podíem utilitzar *React* o utilitzar la tecnología base de la capa de vista de *Django*, i que elegir una o l’altre, depenia de la complexitat de la nostra aplicació. Un cop vam analitzar aquesta aplicació web, vam veure que no requeria un *framework frontend* i que amb la mateixa capa de vista de *Django* en teníem suficient, també facilitant-nos el desenvolupament.

#### **Llibreries externes importants**

En el client, també necessitem utilitzar diverses llibreries per a poder desenvolupar la nostra aplicació.

El nombre d’aplicacions que es fan servir, sobretot en *frameworks* de *JavaScript*, és molt gran, volem destacar dues llibreries i serveis que ens són de gran ajuda per a la realització del projecte.

* ***Google Maps***: Ens ofereix la principal funcionalitat de la nostra aplicació, la qual és la utilització del mapa per tal de marcar els punts i poder navegar per Catalunya. De *Google Maps*, a part del propi mapa, també fem servir *Google Maps* *places*, el qual ens permet obtenir els noms de carrers i poder buscar utilitzant-los i *Google Maps* *directions*, el qual ens permet traçar rutes sobre el mapa.
* ***Socket.io***: És la part del client de la llibreria descrita en l’explicació de servidor, la qual rep els esdeveniments que envia el servidor.

### 5.1.2 Diagrama Domini Frontend

A continuació trobarem el diagrama de domini del frontend de la nostra plataforma. En aquest diagrama hem intentat que es vegin clarament l’organització del nostre sistema i com les classes es relacionen entre elles, el que dona una idea genèrica del funcionament d’aquest.

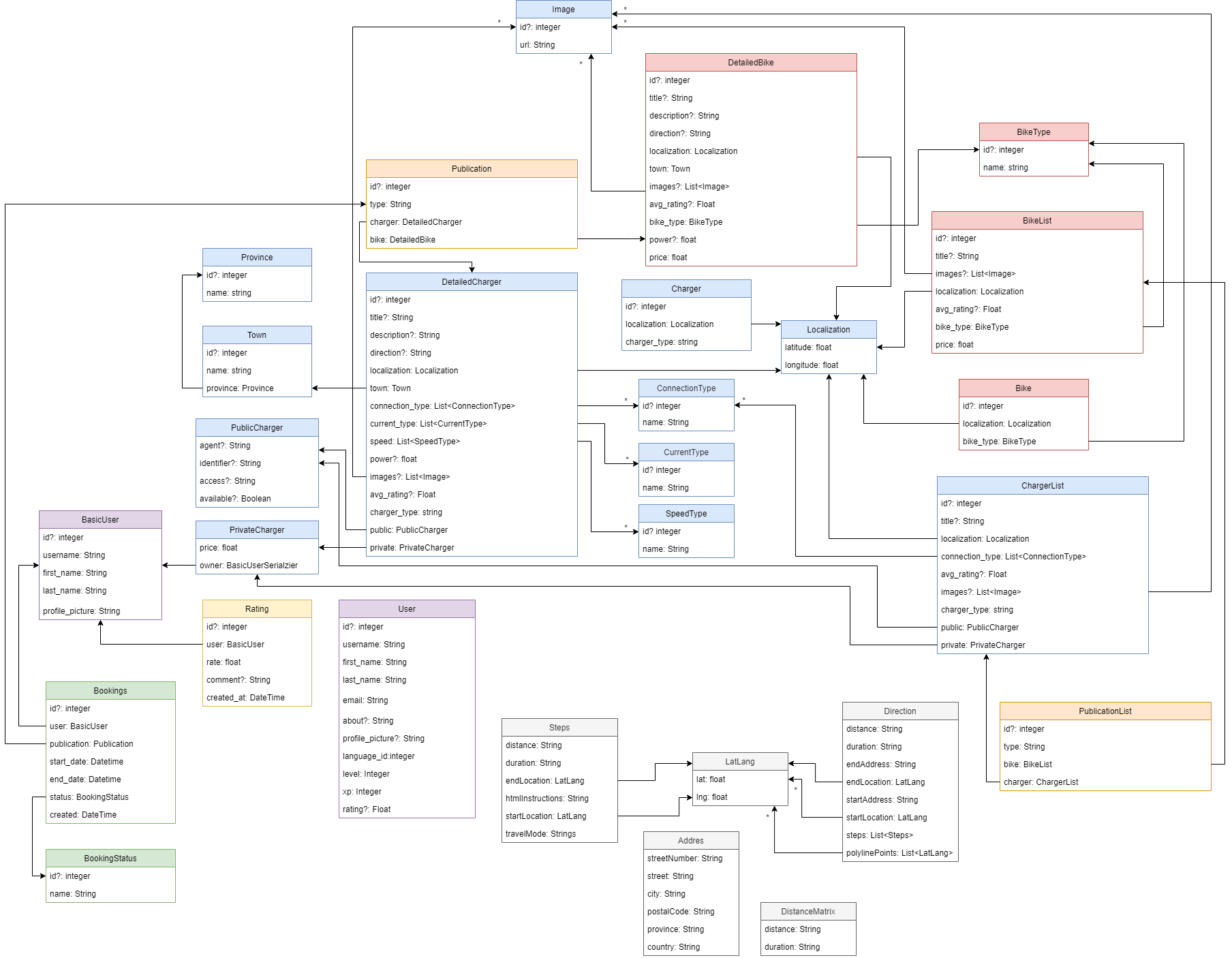


Figura 22: Diagrama domini frontend

### 5.1.3 Diagrama Domini Backend

A continuació trobarem el diagrama de domini del backend de la nostra plataforma. En aquest diagrama hem intentat que es vegin clarament l’organització del nostre sistema i com les classes es relacionen entre elles, el que dona una idea genèrica del funcionament d’aquest.

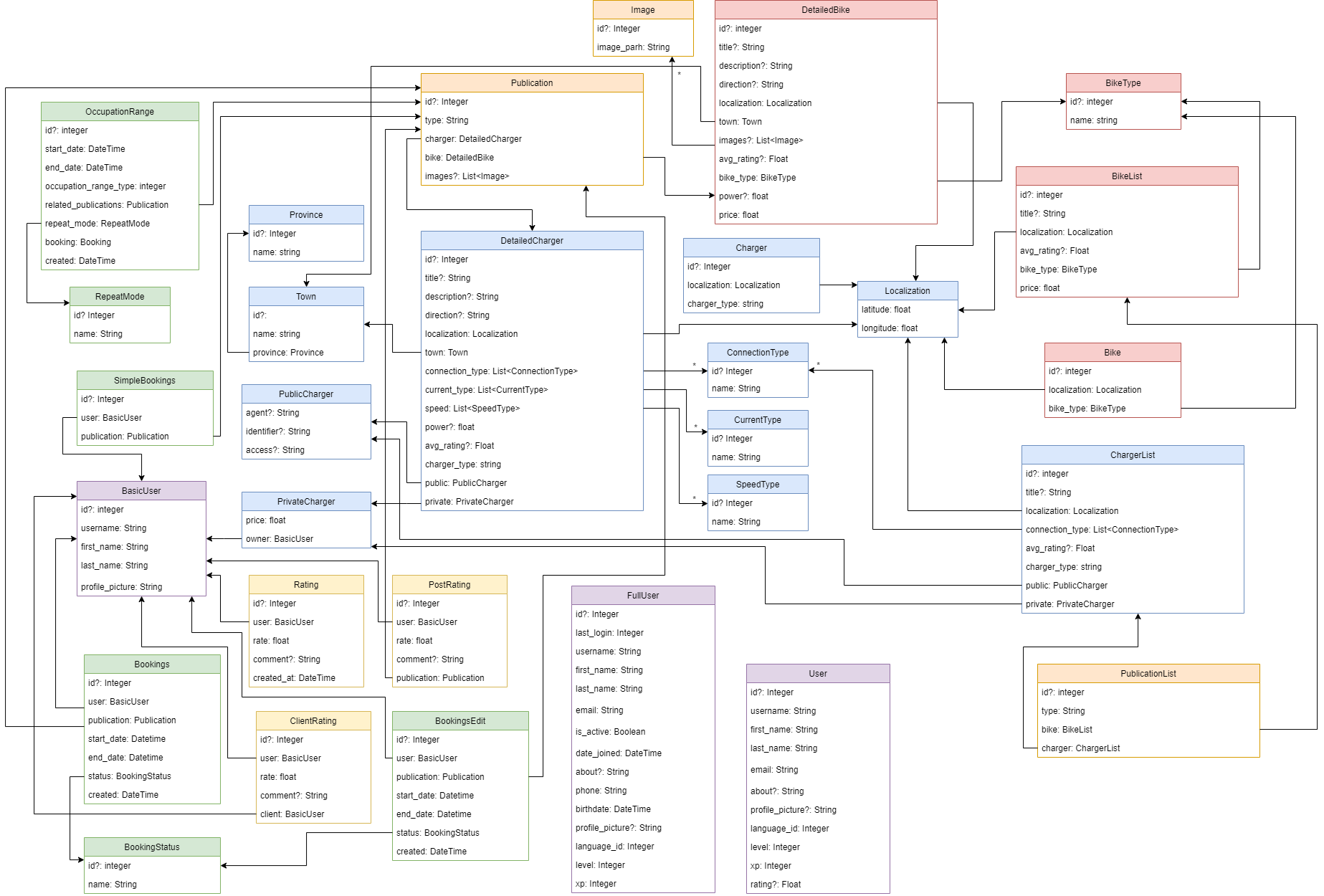


Figura 23: Diagrama domini backend

## Diagrama de components

A continuació trobarem el diagrama de components de la nostra plataforma, el qual il·lustra de manera general com es relacionen els diferents components que formen la nostra aplicació entre ells, i quines dependències tenen. Hem intentat que hi hagués el menor acoplament possible fent els els serveix per aplicació del nostre sistema, la qual cosa ajuda que millori molt, però no és perfecte i, com és inevitable, hi trobem alguns acoplaments.

Les screens estan formades per widgets que les conformen i que són reutilitzats en més d’una screen, però per aquest diagrama no hem volgut filar tant prim i per tant, s’han encapsulat dins de les seves pantalles.

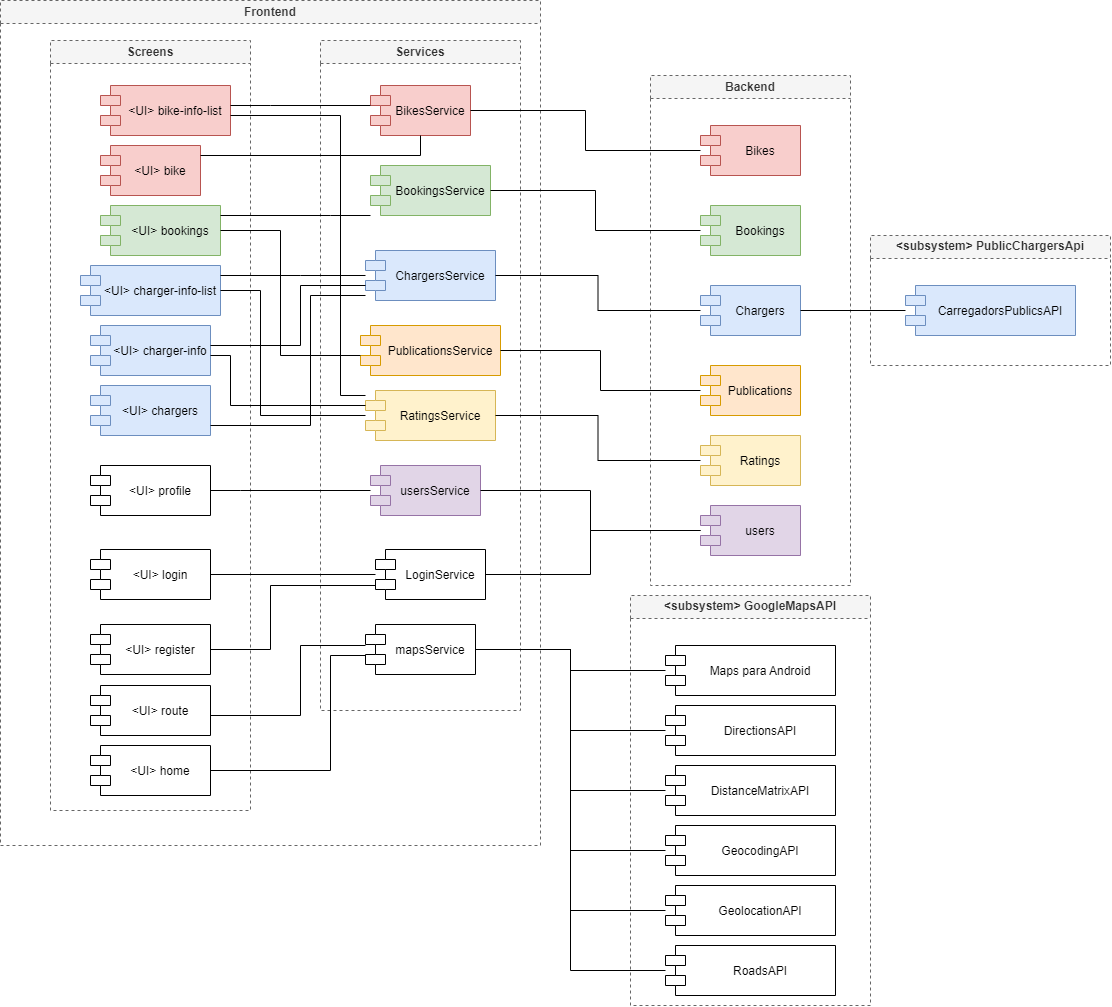


Figura 24: Diagrama de components

## Patrons utilitzats

Entre els patrons de disseny que hem implementat, trobem els més comuns, aquests són: Adaptador i Plantilla. Aquests patrons de disseny ens han ajudat a crear bones pràctiques de cara a la implementació, ja que faciliten el treball al programador i a possibles lectors que a futur llegeixin el codi.

A continuació expliquem amb més detall l’aplicació de cada patró:

* **Adaptador:** Hem utilitzat aquest patró per tractar la informació dels objectes. Tant la informació obtinguda de les diferents taules són transformades a un format JSON per poder tractar-les en diferents funcionalitats. A més a més, s’han implementat serializers que també fan d’adaptador de les dades, ja que ens donen la informació com a objectes i ens proporciona una fàcil manipulació. A la figura 1 podem veure com hem creat el serializer corresponent a una publicació per tal de poder adquirir els atributs com a objectes.

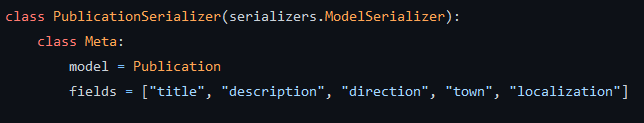


Figura 25: Mostra del serializer implementat per a una publicació

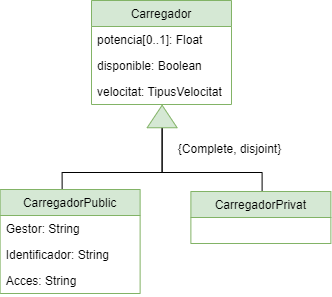
* **Plantilla:** L’ús d’aquest patró el portem a terme per poder encapsular mòduls complexes desenvolupats en l’aplicació. Hi ha metodologies que tenen una funcionalitat molt semblant, però que es diferencien en petits detalls, per tant, es crea una superclasse corresponent a l'objecte comú i es desenvolupa les funcionalitats referents a cada objecte, de forma que integra la part comuna més la diferenciació pertinent. Un possible exemple seria el de carregador, que s'especialitza en públic o privat, tal com mostrem a la figura 2.

Figura 26: Mostra del UML en relació amb el patró plantilla

* **Singleton:** Aquest patró consisteix en limitar el número d'instàncies d’una classe, i s’acostuma a utilitzar per controlar com i quan s’accedeix a una instància o per facilitar l'accés global sense ocupar espais de noms. En el nostre cas l’hem aplicat en algunes classes com per exemple la de LoginService*,*  que s’encarrega de gestionar el login dels usuaris, i considerem que no ha de tenir més d’una instancia.

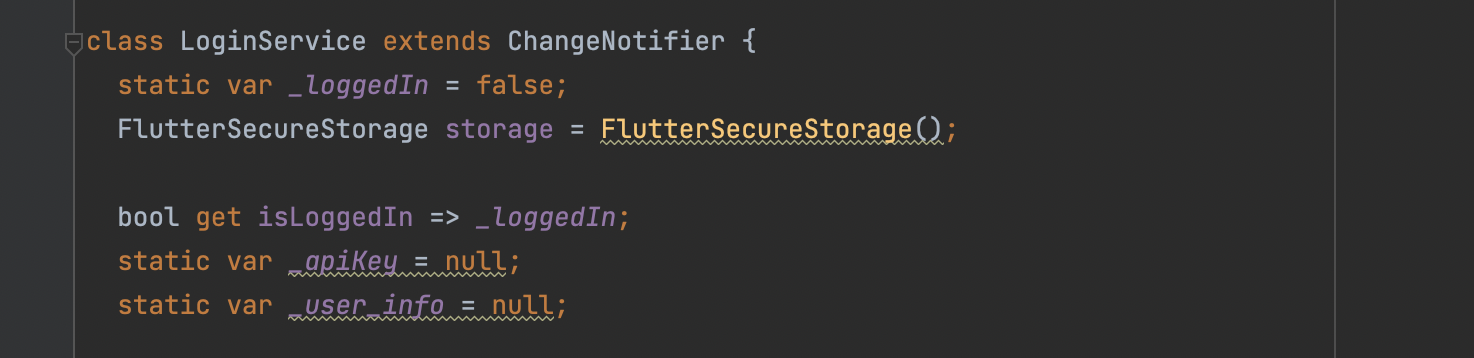


Figura 27: Codi de la classe LoginService

* **Observer:**  S’utilitza quan hi ha un conjunt d’elements dependents d’un altre, que han de ser notificats, quan l’element del que depenen passa a un cert estat. En el nostre projecte, principalment n’hem fet ús d’aquest patró al frontend, ja que cada vegada que volem actualitzar un widget/pantalla, fem ús de la funció *setState*  que ens permet fer modificacions y renderitzar els components amb les dades actualitzades.



Figura 28: Codi de la classe LoginService

* **Command:** Aquest patró permet sol·licitar una operació a un objecte sense conèixer realment el contingut d'aquesta operació, ni el receptor real. Per això s'encapsula la petició com un objecte, de manera que a més facilita la parametrització dels mètodes. En el nostre projecte l’utilitzem al fer crides desde el frontend al backend.



Figura 29: Codi de la classe BackendService, on s’aplica el patró

* **Mediator:** És un patró de comportament en el que s’encapsulen, com a un conjunt, objectes que interactuen. El podem trobar aplicat al frontend amb l’enrutament de les diferents pantalles. D’aquesta manera es facilita la comunicació i redirecció entre diferents pantalles.

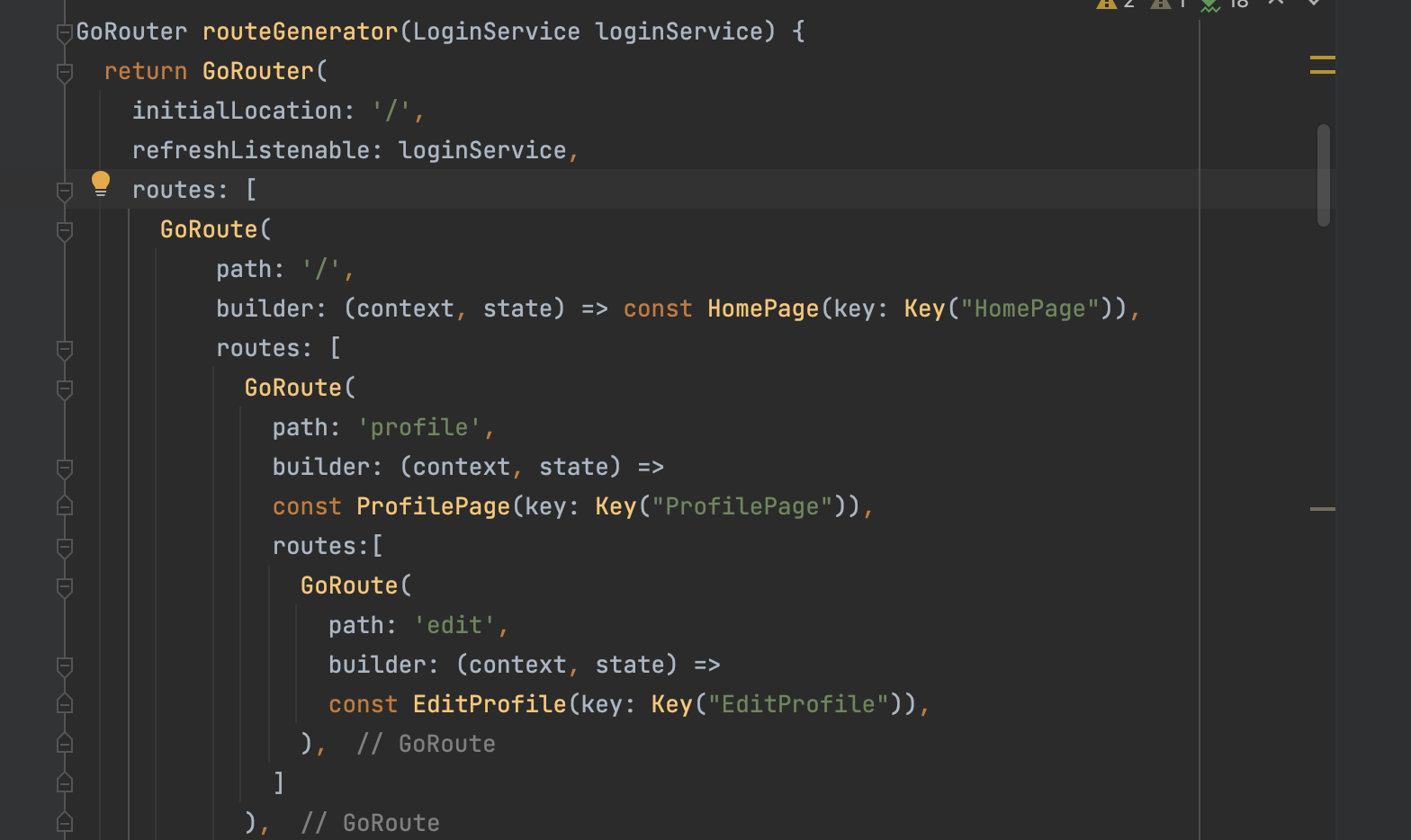
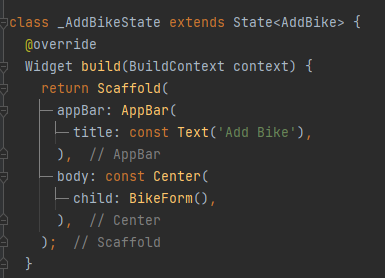
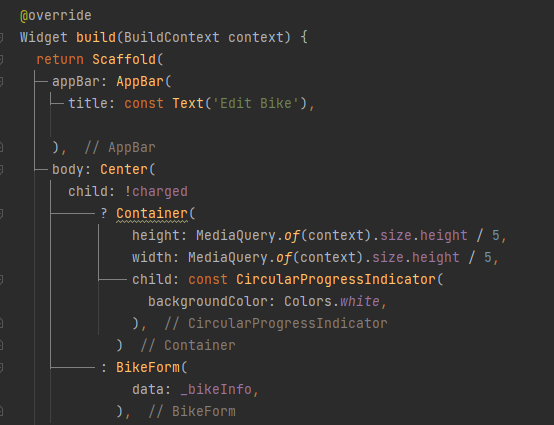


Figura 30: Codi de la classe BackendService, on s’aplica el patró

* **Decorator**: Aquest patró s’aplica quan vols que un component canviï el seu comportament durant el temps d'execució sense canviar l’objecte en si, com pot ser, en el nostre cas en els formularis de creació i d’edició tant de bicicletes com de carregadors, les nostres classes charger\_form i bike\_form, depenent dels atributs amb els quals són cridats tenen un comportament diferent, els d'editar o de crear un nou bike/charger. En la imatge 31 veiem com es crea una bicicleta, cridem a BikeForm() sense cap paràmetre, i en la foto 32 veiem com s’edita una bicicleta, tornem a cridada a BikeForm(), però en aquest cas li passem un paràmetre. Per tant, la classe BikeForm no canvia, és la mateixa, però té un comportament diferent, el seu comportament canvia en temps d’execució.

Figura 31 y 32: Exemple de codi on s’aplica el patró decorator. 

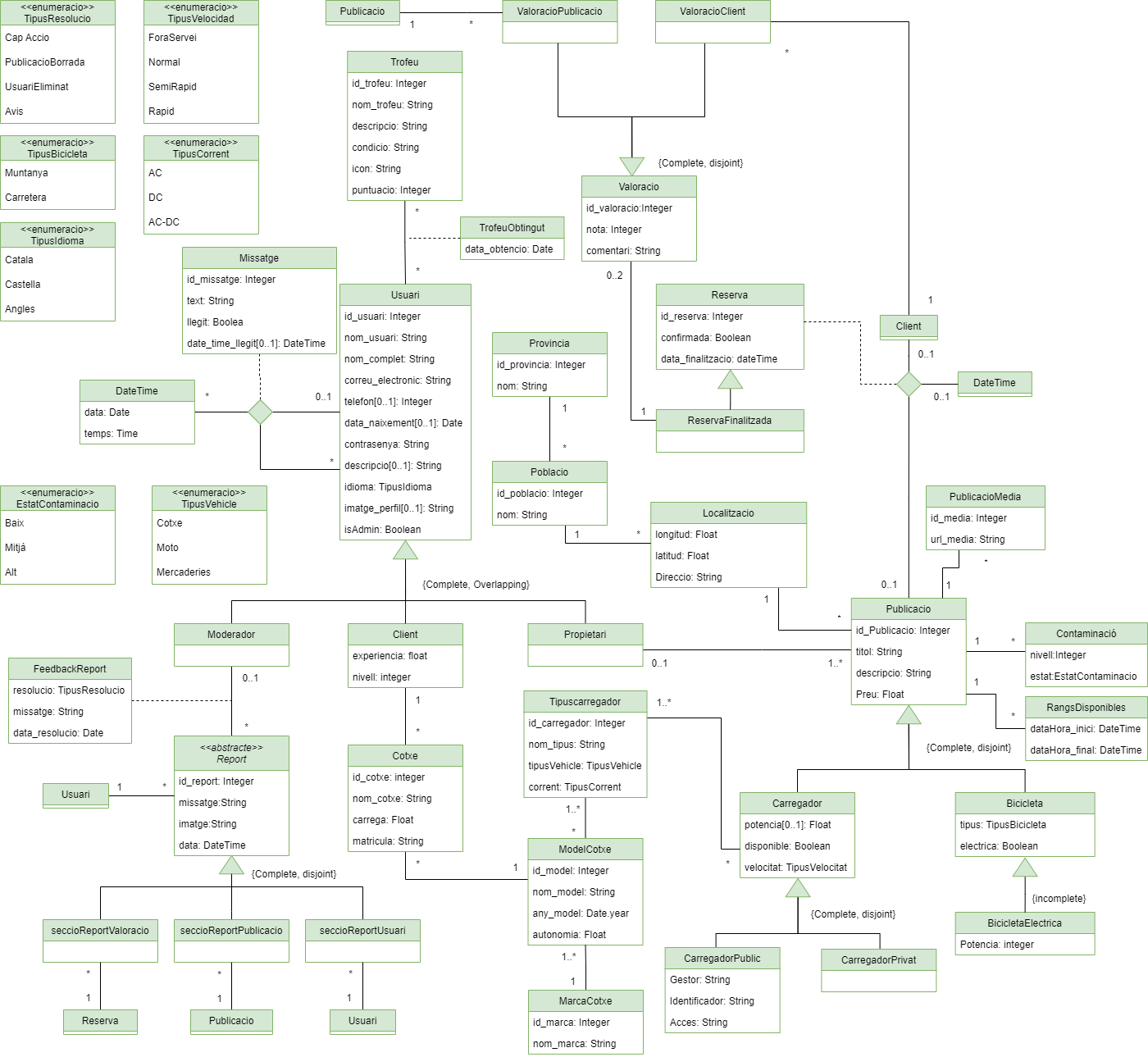
## 

## Data Models

### 5.4.1 Diagrama de classes UML del problema

A continuació mostrem l’*UML* del nostre sistema on mostrem totes les classes que conté el nostre projecte i la relació que hi ha entre elles. Seguidament, indiquem quines són les restriccions textuals d’integritat que esdevenen d’aquest diagrama de cla**sses.**

#### Sprint 1



**Restriccions textuals:**

**R.T.1:** Claus externes:

* (Usuari, id\_usuari)
* (DateTime, data + temps)
* (Report, id\_report)
* (Trofeu, id\_trofeu)
* (MarcaCotxe, id\_arca)
* (ModelCotxe, MarcaCotxe:id\_marca + id\_model)
* (Cotxe, id\_cotxe)
* (TipusCarregador, id\_carregador)
* (Publicació, id\_publicacio)
* (Contaminació, Publicació:id\_publicacio + nivell)
* (PublicacioMedia, Publicació id\_publicacio + id\_media)
* (Localitzacio, longitud + latitud)
* (Poblacio, id\_poblacio)
* (Provincia, id\_provincia)
* (Valoracio, id\_valoracio)
* (RangsDisponibles, Publicacio + dataHora\_inici)

**R.T.2:** Dos reserves amb la mateixa publicació, no es poden solapar temporalment.

**R.T.3:** La data de finalització d’una reserva no pot ser anterior o igual a la data de la reserva a la qual pertany.

**R.T.4:** Les dues valoracions associades a una reserva finalitzada han de ser de tipus diferents.

**R.T.5:** Una bicicleta només és BicicletaElectrica si l’atribut elèctrica és cert.

**R.T.6:** Només es pot reservar un carregador en cas que l’atribut disponible sigui cert.

**R.T.7:** Una publicació ha de tenir un propietari si no és de tipus carregadorPublic.

**R.T.8:** La data de resolució d’un report ha de ser anterior a la data de resolució del seu feedback.

**R.T.9:** La data d’obtenció d’un trofeu ha de ser anterior o igual a la data actual.

**R.T.10:** Perquè una reserva estigui finalitzada, cal que hagi estat confirmada.

**R.T.11:** La valoració d’un client assignada a una reserva, ha d’estar relacionada amb el client que ha fet la reserva.

**R.T.12:** La valoració de la publicació feta a una reserva finalitzada, ha d’estar relacionada amb la publicació en la qual s’ha fet la reserva.

**R.T.13:** Un client no pot reservar una publicació de la qual n’és propietari.

**R.T.14:** L’any del model d’un cotxe ha de ser anterior o igual a l’any actual.

**R.T.15:** Els rangs de disponibilitat d’una publicació no es poden solapar temporalment.

**R.T.16:** Un usuari ha d’estar relacionat com a mínim amb un cotxe per poder fer una reserva d'una publicació del tipus carregador.

**R.T.17**: Una reserva no pot ser d’una publicació de carregador públic

**R.T.18**: La valoració de client ha d’estar feta pel propietari de la publicació del qual pertany la reserva de la valoració.

**R.T.19**: La valoració de publicació ha d’estar feta pel client de la publicació del qual pertany la reserva de la valoració.

**R.T.20**: Un usuari moderador només pot ser del tipus moderador.

#### Sprint 2

Figura 34: Diagrama UML del problema

**Restriccions textuals:**

**R.T.1:** Claus externes:

* (Usuari, id\_usuari)
* (DateTime, data + temps)
* (Report, id\_report)
* (Trofeu, id\_trofeu)
* (MarcaCotxe, id\_arca)
* (ModelCotxe, MarcaCotxe:id\_marca + id\_model)
* (Cotxe, id\_cotxe)
* (TipusCarregador, id\_carregador)
* (Publicació, id\_publicacio)
* (Contaminació, Publicació:id\_publicacio + nivell)
* (PublicacioMedia, Publicació id\_publicacio + id\_media)
* (Localitzacio, longitud + latitud)
* (Poblacio, id\_poblacio)
* (Provincia, id\_provincia)
* (Valoracio, id\_valoracio)
* (RangsDisponibles, Publicacio + dataHora\_inici)
* (Ocupacio,Data\_inici + data\_final)

**R.T.2:** Dos reserves amb la mateixa publicació, no es poden solapar temporalment.

**R.T.3:** La data de finalització d’una reserva no pot ser anterior o igual a la data de la reserva a la qual pertany.

**R.T.4:** Les dues valoracions associades a una reserva finalitzada han de ser de tipus diferents.

**R.T.5:** Una bicicleta només és BicicletaElectrica si l’atribut elèctrica és cert.

**R.T.6:** Només es pot reservar un carregador en cas que l’atribut disponible sigui cert.

**R.T.7:** Una publicació ha de tenir un propietari si no és de tipus carregadorPublic.

**R.T.8:** La data de resolució d’un report ha de ser anterior a la data de resolució del seu feedback.

**R.T.9:** La data d’obtenció d’un trofeu ha de ser anterior o igual a la data actual.

**R.T.10:** Perquè una reserva estigui finalitzada, cal que hagi estat confirmada.

**R.T.11:** La valoració d’un client assignada a una reserva, ha d’estar relacionada amb el client que ha fet la reserva.

**R.T.12:** La valoració de la publicació feta a una reserva finalitzada, ha d’estar relacionada amb la publicació en la qual s’ha fet la reserva.

**R.T.13:** Un client no pot reservar una publicació de la qual n’és propietari.

**R.T.14:** L’any del model d’un cotxe ha de ser anterior o igual a l’any actual.

**R.T.15:** Els rangs de disponibilitat d’una publicació no es poden solapar temporalment.

**R.T.16:** Un usuari ha d’estar relacionat com a mínim amb un cotxe per poder fer una reserva d'una publicació del tipus carregador.

**R.T.17**: Una reserva no pot ser d’una publicació de carregador públic

**R.T.18**: La valoració de client ha d’estar feta pel propietari de la publicació del qual pertany la reserva de la valoració.

**R.T.19**: La valoració de publicació ha d’estar feta pel client de la publicació del qual pertany la reserva de la valoració.

**R.T.20**: Un usuari moderador només pot ser del tipus moderador.

**R.T.21:** Una reserva no pot ser finalitzada si l’estat de la reserva no es confirmada.

## 5.4.1 Diagrama de Base de dades

#### Sprint 1

A continuació mostrem el diagrama de base de dades amb el que hem treballat en aquest primer sprint. Com es pot veure no es veuen representades totes les classes que hi ha l’UML, això és degut a que només hem definit els models que ens han resultat necessaris per a realitzar les funcionalitats del primer sprint, que són els de Chargers, Users i Bookings. En el diagrama es mostren els models amb els seus atributs i el tipus que tenen i també es mostra l’atribut pel qual es relacionen els diferents models.

## 

Figura 35: Diagrama de Bases de dades Sprint 1

#### Sprint 2

El diagrama que hi ha a continuació és el diagrama del 2n sprint. Veiem que ja és bastant més complex que el del primer, tot i que encara no veiem representades totes les classes que tenim a l’UML del problema, aquestes vindran amb el pròxim sprint. Els models afegits han estat el de Ratings, Publications i Bikes.

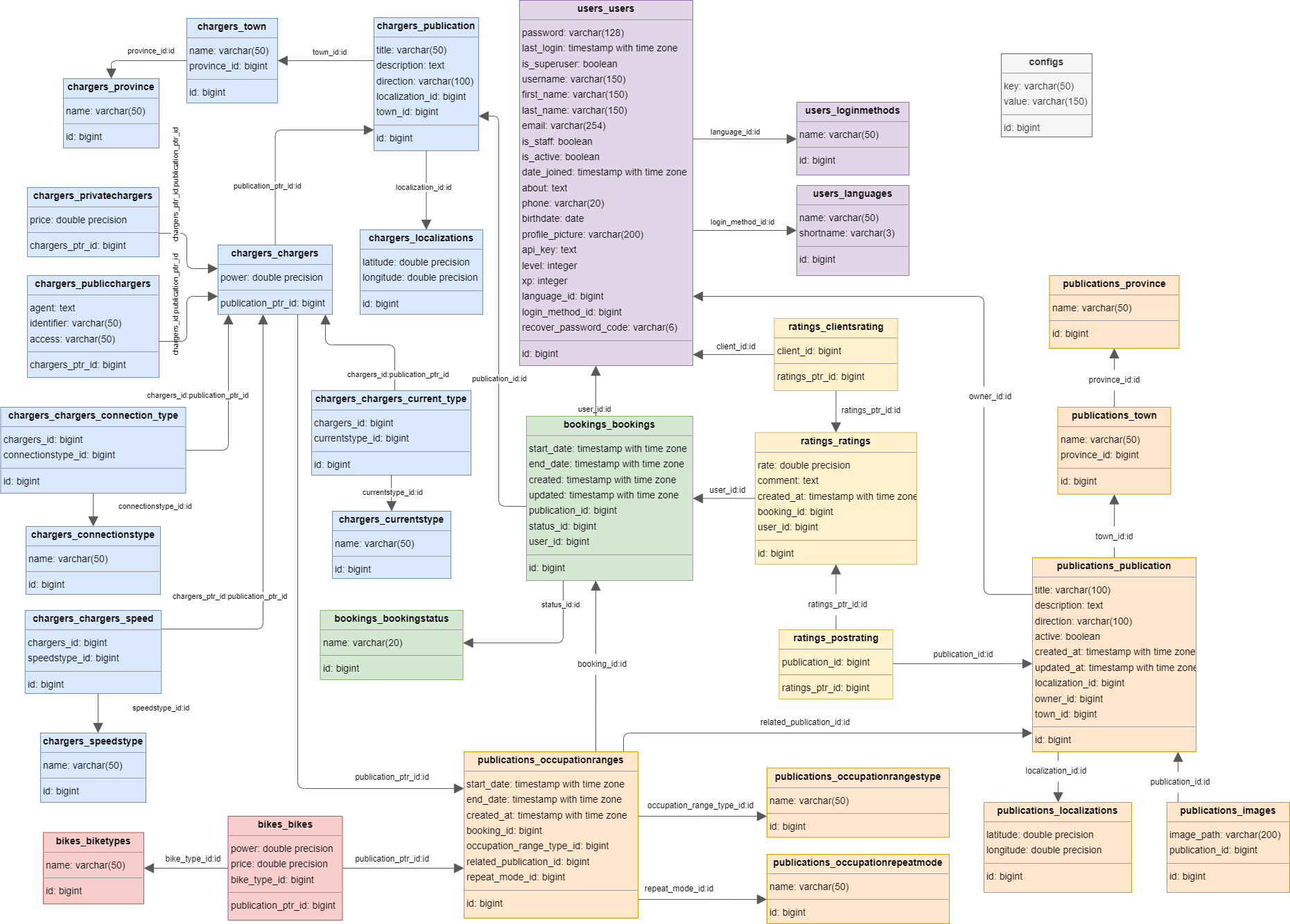
****

Figura 36: Diagrama de Bases de dades Sprint 2

## APIs

Per tal del correcte funcionament de la nostra aplicació, utilitzem vàries APIs que la proveeixen d’informació i li donen tota la informació que necessita. A continuació, repassarem la informació sobre les APIs que utilitzem.

### 5.5.1. API pròpia

La nostra API és el mètode d’informació principal de l’aplicació, ja que és la que s’encarrega de centralitzar, analitzar i proveïr-li la informació.

La idea de documentar la nostra API en format excel i tenir un full de càlcul actualitzat en cada moment, va nèixer per la necessitat de transmetre informació entre frontend i backend, on l’equip de backend haviem d’anar explicant com funcionava cada endpoint, que se li podia donar i que no, etc. També hi havia la problemàtica inversa, on l’equip de frontend ens demanava un endpoint que els dongués certa informació, i ens l’escribien per privat o per algun canal, on s’acabava perdent.

Per aquest motiu, i amb la idea de facilitar la transmissió d’informació, vam decidir fer una documentació en excel que fos fàcil d’entendre i que fos entenedora.

Vam decidir fer-ho en format excel i no amb una documentació d’OpenApi de Swagger degut a la facilitat que ens suposava fer-ho en excel envers a swagger, així com la facilitat de que tots la poguessim editar (google sheets) i poguessin tenir-la actualitzada al instant.

Després de pensar com podiem distribuir aquest excel, vam decidir separar-ho per pàgines, on cada pàgina era una branca principal de la API. En cada pàgina, els endpoints contenen la següent informació, dividida en columnes:

| Endpoint | Mostra el path que cal concatenar al que surt a la part superior per arribar a la ubicació |
| --- | --- |
| Method | Mètode que es pot utilitzar en el endpoint |
| Description | Petita descripció de que fa el mètode de cada endpoint |
| Query params | Mètodes que es poden passar en format de query params. |
| Body | Paràmetres que es poden passar en el body (nomes en put i post) |
| Response Serializer | En cas de que la resposta sigui serialitzada, es diu quina estructura s'utilitza. Les estructures dels serializers es poden trobar a diagrames |
| Response Many | Indica con si o no si devuelve más de un objeto o solo devuelve uno. |
| Paginació | Indica si utilitza paginació o no. En cas que n'utilitzi, donarà la informació de manera pàginada (de 10 en 10). |
| Notes | Indica coses addicionals, de si falta fer alguna cosa, o si cal tenir quelcom en compte |

Per a documentar els serializers, vam fer un altre document on estaven representats aquests serializers que utilitzavem i quins atributs tenien, per tal de que els poguessin utilitzar a frontend quan ho implementessin.

Per altra banda, es va crear un codi de colors per tal d’indicar als desenvolupadors de frontend que un endpoint tenia canvis, estava pendent de fer o s’estava realitzant. Per això, vam decidir utilitzar el següent sistema de colors.

| Verd | Endpoint verificat i testejat. Llest per ser utilitzat |
| --- | --- |
| Groc | Endpoint verificat i testejat, pero amb canvis (mirar notes) |
| Taronja | Endpoint en procés o planificat |
| Vermell | Endpoint pendent de fer |

Es va dir al equip de frontend que, en cas de voler un nou endpoint, només l’havien d’afegir amb els camps que necessitessin i l’equip de frontend ja els hi farien.

Com a última informació que dóna aquest excel, es com fer crides quan és un endpoint amb paginació, on cal fer-ho de la següent manera:

*Per tal d'utilitzar la paginació, caldrà que passeu la pàgina que voleu obtenir. La manera més fàcil, és fer la crida normal al endpoint, i seguidament, anar fent la crida al enllaç de l'atribut next que retorna la resposta. En cas de no voler-ho utilitzar, també podeu passar els següents paràmetres per query params: ?page=1&limit=15. Page es el numero de la pagina que volem obtenir. límit és el nombre d'elements que volem rebre. És important que el límit no es canviï en totes les pàgines, ja que si canvia, pot ser que se us repeteixin elements. En cas de no posar limit, serà de 10.*

Per a realitzar aquesta API, la vam dividir en vàries seccions principals, les quals es ramificaven en altres crides més detallades.

Les diferents seccions que tenim son:

#### 

#### Chargers (/api/chargers)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| / | GET | Permet obtenir els carregadors filtrats per les diferents propietats de charger o per si son de tipus public o privat. També permet ordenar-los per qualsevol de les seves propietats. | current, speed, connection, type, available, price, order, latitude, longitude, radius |  | ChargerSerializer | Si | No |
| POST | Permet crear un nou carregador de tipus privat |  | title, description, latitude, longitude, power, price, speed\_id, connection\_type\_id, current\_type\_id, images | DetailedChargerSerializer | No | No |
| /<int:id>/ | GET | Permet obtenir tota la informació del carregador passat a la url. |  |  | DetailedChargerSerializer | No | No |
| PUT | Permet actualitzar la informació del carregador passat per url |  | title, description, latitude, longitude, power, price, speed\_id, connection\_type\_id, current\_type\_id, images | DetailedChargerSerializer | No | No |
| DELETE | Permet eliminar el carregador privat amb l'identificador passat per url |  |  |  | No | No |
| /list/ | GET | Permet obtenir els carregadors filtrats per les diferents propietats de charger o per si son de tipus public o privat. També permet ordenar-los per qualsevol de les seves propietats. La diferència es que aquest retorna tota la informació, molt util per a mostrar-ho en format llista | current, speed, connection, type, available, price, order, latitude, longitude |  | ChargerListSerializer | Si | Si |
| speed/ | GET | Permet obtenir tots els valors d'speed |  |  | SpeedTypeSerializer | Si | No |
| current/ | GET | Permet obtenir tots els valors de current |  |  | CurrentTypeSerializer | Si | No |
| connection/ | GET | Permet obtenir tots els valors de connection |  |  | ConnectionTypeSerializer | Si | No |

#### Bikes (/api/bikes)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| / | GET | Permet obtenir les bicicletes filtrades per les diferents propietats. També permet ordenar-los per qualsevol de les seves propietats. | bike\_type, price, order, latitude, longitude, radius |  | BikeSerializer | Si | No |
| POST | Permet crear una nova bicicleta amb els atributs passats |  | latitude, longitude, town, title, description, direction, bike\_type, power, price | DetailedBikeSerializer | No | No |
| /<int:id>/ | GET | Permet obtenir tota la informació de la bicicleta passada a la url. |  |  | DetailedBikeSerializer | No | No |
| PUT | Permet actualitzar la informació de la bicicleta passada per url |  | latitude, longitude, speed, connection\_type, current\_type, town, title, description, direction, power, price | DetailedBikeSerializer | No | No |
| DELETE | Permet eliminar la bicicleta amb l'identificador passat per url |  |  |  | No | No |
| /list/ | GET | Permet obtenir les bicicletes filtrades per les diferents propietats. També permet ordenar-los per qualsevol de les seves propietats. La diferència es que aquest retorna tota la informació, molt util per a mostrar-ho en format llista | bike\_type, price, order, latitude, longitude |  | BikeListSerializer | Si | Si |
| types/ | GET | Permet obtenir tots els noms dels tipus de bicicleta |  |  | BikeTypeSerializer | Si | No |

#### 

#### Publications (/api/publications)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| /<int:id>/occupation/ | POST | Permet crear una nova franja de bloqueig on no es podran fer reserves | range: List<startdate,enddate, repeatmode> |  | OccupationRangeSerializer | No | No |
| /<int:id>/occupation/<int:id> | GET | Permet obtenir una franja d'ocupació determinada |  |  | OccupationRangeSerializer | No | No |
| PUT | Permet editar una franja | startdate,enddate, repeatmode |  | OccupationRangeSerializer | No | No |
| DELETE | Permet eliminar una franja |  |  |  | No | No |
| /<int:id>/occupation/<int:year>/<int:month>/ | GET | Permet obtenir totes les franjes bloquejades que té una publicació en un mes determinat |  |  | Map on les claus son dies de la setmana i els valors arrays d'ocupacions | Si | No |
| /<int:id>/occupation/<int:year>/<int:month>/<int:day>/ | GET | Permet obtenir totes les franjes bloquejades que té una publicació en un dia determinat |  |  | array d'ocupacions | Si | No |
| /occupation/repeatTypes/ | GET | Permet obtenir els diferents modes de repetició que pot tenir una ocupació |  |  | RepeatTypesSerializer | Si | No |
| /<int:id>/upload | POST | Permet pujar la foto d'una bici o d'un carregador |  | Multiples imatges en body.files | ImageSerializer | Si | No |

#### Bookings (/api/bookings)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| / | GET | Permet obtenir els bookings, de manera paginada, del usuari loguejat | orderby=["date","town"], year, month, day |  | BookingsDetailedSerializer | Si | Si |
| POST | Permet crear un nou booking |  | publication, startdate, enddate | BookingsDetailedSerializer | No | No |
| /<int:id>/ | GET | Permet obtenir una reserva concreta, la qual correspon a la de la id passada per url |  |  | BookingsDetailedSerializer | No | No |
| PUT | Permet confirmat la reserva (owner) |  | confirmed(boolean on 1 = confirmed i 0 = denied) |  | No | No |
| DELETE | Permet eliminar una reserva |  |  |  | No | No |
| /history/ | GET | Permet obtenir les reserves antigues del usuari loguejat | order, cancelled |  | BookingsDetailedSerializer | Si | Si |
| /owner/ | GET | Permet obtenir les reserves que pertanyen a les publicacions propies de l'usuari, és a dir, de les reserves que li han fet al usuari loguejat | type, order |  | BookingsDetailedSerializer | Si | Si |
| /owner/history/ | GET | Permet obtenir les reserves que li han realitzat a les seves publicacions, les quals son antigues | order, cancelled |  | BookingsDetailedSerializer | Si | Si |

#### Ratings (/api/ratings)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| /publication/<int:id>/ | GET | Permet obtenir les ratings que té una publicació | order |  | RatingsSerializer | Si | Si |
| POST | Permet crear una nova rating de la publicació identificada per id, en la qual la reserva ja esta finalitzada |  | booking\_id, rate, comment | RatingsSerializer | No | No |
| /client/<int:id>/ | GET | Permet obtenir les ratings que té un usuari | order |  | RatingsSerializer | Si | Si |
| POST | Permet crear una nova rating del usuari identificat per id, en la qual la reserva ja esta finalitzada |  | booking, rate, comment | RatingsSerializer | No | No |

#### 

#### Users (/api/users)

| **Endpoint** | **Method** | **Description** | **Query params** | **Body** | **Response Serializer** | **Response Many** | **Paginació** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| / | GET | Permet obtenir la informació de l'usuari loguejat |  |  | UserSerializer | No | No |
| PUT | Permet modificar la informació d'un usuari loguejat |  | first\_name, last\_name, about, profile\_picture | UserSerializer | No | No |
| /<int:id>/ | GET | Permet obtenir la informació d'un usuari concret |  |  | UserSerializer | No | No |
| /language/ | GET | Permet obtenir l'idioma del usuari loguejat |  |  | {int} | No | No |
| PUT | Permet modificar l'idioma del usuari loguejat |  |  |  | No | No |
| /<int:id>/posts/ | GET | Permet obtenir les publicacions de l'usuari loguejat |  |  | PublicationListSerializer | Si | Si |
| /upload/ | PUT | Permet actualitzar la imatge de perfil de l'usuari |  | Multiples imatges en body.files | {url:string} | No | No |
| /register/ | POST | Permet crear un nou compte natiu |  | username, first\_name, last\_name, about, profile\_picture, birthday, password, language, phone | api\_key | No | No |
| /password/recovery/ | GET | En cridar-lo, envia un sms amb un codi de recuperació al usuari. | username |  |  | No | No |
| PUT | En cridar-lo, verifica si el codi es correcte. En cas de ser-ho, permetrà reiniciar la contrasenya |  | username,code | api\_key | No | No |
| /password/change/ | PUT | Permet canviar la contrasenya de l'usuari |  | password |  | No | No |
| /logout/ | DELETE | Permet tancar la sessió de l'usuari loguejat |  |  |  | No | No |
| /login/ | POST | Permet crear un nou inici de sessió |  | username, password | api\_key | No | No |

### 5.5.2. Nombre d’APIs externes

Encara que l’API explicada anteriorment és la principal, la nostra aplicació també s’alimenta de vàries APIs que ens faciliten la programació de la plataforma. En el nostre cas, tenim 3 APIs externes de les quals obtenim informació:

#### Google Maps:

Google maps està format per múltiples APIs que fan la vida més senzilla als programadors, les quals ens ofereixen tota la informació sobre els mapes, les localitzacions, les rutes, distàncies, etc), fent que ens sigui més fàcil mostrar la informació al usuari o guiar-lo.

Algunes de les APIs que hem volgut utilitzar, son de pagament, i per tant, haviem de pagar en cas de voler-les utilitzar. Tot i així, hem jugat en que et donen 200€ de crèdit cada mes (posant la tarjeta) i, si et passes del límit et fan pagar, cosa que no ens passarà ja que no arribem a fer tantes peticions.

Concretament, hem obtingut la informació de les següents APIs:

* Maps SDK para Android: Ens ofereix el widget del mapa i la navegació per aquest mapa, es a dir, el poder-nos moure d’un costat a l’altre, poder fer zoom, poder mostrar marcadors, etc.
* DirectionsAPI: Ens permet trobar les diferents opcions de ruta entre un punt i un altre, ideal per al nostre sistema de ruta. També ens diu pas per pas que hauria de fer el cotxe per tal de conduir d’un lloc a l’altre.
* DistanceMatrixAPI: Ens permet calcular el temps i la distància entre dos punts en cotxe.
* GeocodingAPI: Ens permet convertir direccions a coordenades geogràfiques i a la inversa.
* GeolocationAPI: Permet obtenir la ubicació en temps real del dispositiu.
* Roads: Complementa la directions API a l’hora de dibuixar-nos les rutes, donant-nos alguna funcionalitat que no té la directionsAPI.

#### AB&B

Tot i que la implementació de les crides la farem en el següent sprint, aquesta API ens oferirà la contaminació que hi ha en els carregadors, per a que l’usuari pugui veure la polució que s’hi trobarà i l’aire que respirarà mentres s’espera a que el cotxe es carregui. Per a més informació, es pot consultar l’apartat indicat.

#### Estacions de recàrrega per a vehicles elèctrics a Catalunya (dades obertes)

Aquesta API ens ofereix la informació dels carregadors que podem trobar per a Catalunya i que son per a ús públic.

La informació que ens dona és suficient per a l’ús que li volem donar en la nostra aplicació, encara que un dels problemes en els que ens hem trobat, és que els valors de les dades, sobretot dels que haurien de ser fixes o una elecció entre uns quants valors, els quals solen tenir valors erronis o amb faltes d’ortografia, i fan més difícil l’obtenció de dades.

Per a obtenir la informació, vam decidir que el millor era demanar les dades un cop cada hora i de manera asincrona, fent que no s’afegeixi el delay de demanar les dades a la API cada cop que es fa una consulta. El que es fa es obtenir-los i parsejar-los per a que s’adaptin als nostres models, podent-los filtrar i fer qualsevol cosa amb les dades.

# 

# Referències

1. Planning Poker Online

* + <https://planningpokeronline.com/>

1. Docker Swarm

* + <https://docs.docker.com/engine/swarm/>

1. Kubernetes

* + <https://kubernetes.io/>

1. Documentació frameworks backend
   * <https://www.interviewbit.com/blog/backend-technologies/>
   * [https://doit.software/blog/backend-technologies](https://doit.software/blog/backend-technologies#screen6)
   * <https://blog.back4app.com/backend-technologies/>

1. Conveni PEP8 python
   * <https://peps.python.org/pep-0008/>

1. Conveni Mozilla javascript
   * <https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/JavaScript>