

**SPRINT 1: 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 del sprint 1](#_tpsqqt5wzxbn) 4

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

[1.3. Què ha fet cadascú](#_lz540fqujrl7) 6

[1.4. Avaluació de companys](#_ml0o1p9m8f55) 10

[**2. Requisits**](#_70rwu5sjobpj) **12**

[2.1. Not List](#_4orhgki8j36o) 12

[2.2. Backlog](#_92cvirpr68d3) 14

[2.3. Definició tasques Sprint 1](#_gjhzk1z9t2iq) 16

[2.4. Requeriments no funcionals](#_kritwj1w3384) 19

[2.5. Aspectes transversals](#_8te3grf70uxe) 22

[2.6. Serveis](#_yoyozg65c9ij) 24

[**3. Agile ceremony**](#_e8a3kifqkjat) **25**

[3.1. Estadístiques de l'avanç del projecte](#_lhhufi25mkua) 25

[**4. Metodologia**](#_7xs8m5g4iwq) **28**

[4.1. Visió general](#_helrsp46qtqd) 28

[4.2. Gestió del projecte](#_gemsxi2jg73q) 29

[4.3. Gestió de versions](#_v826808c0dbc) 30

[4.4. Comunicació equip](#_wmna8oxsbbhw) 32

[4.5. Gestió qualitat](#_cz4kiacszyln) 34

[4.6. Gestió de proves](#_gju8rbl5nr8g) 34

[4.7. Gestió configuracions](#_uqzvqqmrfdo4) 35

[4.8. Integració amb altres grups](#_d00yh67p8xss) 36

[4.9. Gestió de bugs](#_h5xnena1qtw2) 37

[4.10. IDEs](#_8b8ucmtqslf6) 38

[4.11. Llenguatges](#_w2q204gcyufl) 38

[4.12. Gestió base de dades](#_mymzkp6mq33h) 41

[**5. Descripció tècnica**](#_cdmxgeylv1j5) **42**

[5.1. Diagrama arquitectura física](#_5yljauks3cjs) 42

[5.2. Diagrama de classes UML](#_mrt5b96507zs) 50

[5.3. Diagrama de Base de dades](#_66urdmcenkm7) 53

[5.4. Mockups](#_gqjvxx57jze1) 54

[5.5. Stakeholders](#_l3acf5dzr5l6) 62

[**6. Referències**](#_prupja6a9c1f) **64**

# 

# 

# Introduction

## Resum del 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 que 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, per a que cadascú pugui adaptar-se als seus temps.

## 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, que 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 com a *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 realitzar 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 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, 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. Predim 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 l’equip, seleccionar les històries d’usuari que s’hauran de fer durant l’sprint, dividir aquestes històries d’usuari en tasques i distribuir aquestes tasques el més equitativament possible entre els membres del grup i organitzar reunions que s’han realitzat periòdicament durant l’sprint mitjançant el Discord per a organitzar-nos i mantenir-nos al dia dels avesos de l'equip. 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?

Primer de tot abans de començar l’sprint vaig fer l'organització d’aquest mitjançant el *Taiga*. Per a fer-ho, vaig 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, vaig definir diferents tasques per a cada història d’usuari per a poder repartir la feina amb més facilitat. A continuació, vaig consultar tota la documentació que s’havia d’entregar, preparant el document amb la documentació de l’última entrega i afegint els apartats que encara no s’havien fet al *Taiga* com a *storyless tasks* per tal de mantenir un control sobre l’estat de la documentació.

Un cop tot preparat vam procedir a realitzar la reunió del *sprint planning* on vam revisar les tasques creades per les històries d’usuari i vam assignar les diferents tasques als membres dels grups, aquesta assignació es va dur a terme entre tots per tal d’assignar a cada membre les tasques que li motivessin més i així poder obtenir el millor resultat possible.

Com a sprint master vaig 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.

Un cop acabada la setmana de formació vam dedicar la *weekly*, reunions periòdiques que realitzem 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 seguin la metodologia especificada en aquest document més endavant.

Finalment, es va prendre la d'edició 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.

## Què ha fet cadascú

### Isslam Benali

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.

### Arnau Giménez

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

### Miguel Gutiérrez

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.

### Cristina Migó

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.

### Daniel Oliveras

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 *utils* per agafar instàncies de la base de dades més fàcilment. Finalment vaig crear les rutes necessàries per poder fer la crida des de *frontend*.

### Àlex Ollé

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 *utils* 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.

### Andreu Orensanz

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.

## Avaluació de companys

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 discusió 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 coincidíem 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.

# 

# 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 estan 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 en 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** |
| --- | --- | --- | --- |
| **Gestió usuaris** | Login natiu | 8 | Mitjana |
| Registre d’usuari | 13 | Mitjana |
| Tancar sessió | 2 | Mitjana |
| Edició perfil | 8 | Mitjana |
| Veure perfil | 5 | Mitjana |
| Login Google | 3 | Baixa |
| Login Racó | 2 | Baixa |
| Canviar a perfil de llogater | 8 | Alta |
| Veure trofeus | 13 | Baixa |
| Seleccionar idioma | 8 | Alta |
| **Gestió Carregadors** | Veure carregador concret | 13 | Alta |
| Afegir carregador | 21 | Alta |
| Eliminar carregador | 3 | Alta |
| Mostrar contaminació en punt | 13 | Mitjana |
| Mostrar punts de càrrega públics | 34 | Alta |
| Edició carregador | 8 | Alta |
| Mostrar punts de càrrega privats | 13 | Alta |
| **Gestió Bicicletes** | Veure bicicleta concreta | 13 | Mitjana |
| Afegir bicicleta | 21 | Mitjana |
| Mostrar contaminació en punt | 8 | Mitjana |
| Eliminar bicicleta | 5 | Mitjana |
| Mostrar punts de bicicleta | 13 | Mitjana |
| Editar bicicletes | 8 | Mitjana |
| **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 |
| Veure les meves reserves | 8 | Alta |
| Modificar reserva | 13 | Alta |
| **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 |

## Definició tasques Sprint 1

| **Èpiques** | **Històries d'usuari** | **Points** | **Front/Back** | **id Taiga** | **Tasks** | **Punts Tasques** | **Temps real** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Gestió Usuaris** | Canviar de perfil | 8 | F | 102 | Pàgina principal llogater | 2 | 8 |
| F | 128 | Pàgina base perfil (pàgina en blanc) | 2 | 4 |
| F | 159 | Afegir drawer de l'aplicació amb el seu menú. | 1 | 15 |
| F | 137 | Botó + navegació per a canviar d'un perfil a l'altre | 3 | 2 |
| Seleccionar idioma | 8 | F | 103 | Interacció de seleccionar idioma | 1 | 2 |
| F | 104 | Creació de sistema d'internacionalització | 4 | 4 |
| B | 138 | Guardar selecció usuari/dispositiu per defecte | 2 | 2 |
| B | 139 | Obtenir idioma de l’usuari | 1 | 1 |
| **Gestió Carregadors** | Veure carregador concret | 13 | F | 89 | Interacció de selecció punt de carregador (fer click sobre icona) | 1 | 6 |
| B | 140 | Obtenir informació del punt | 2 | 2 |
| F | 90 | Mostrar tarjeta amb informació bàsica (1/3 pantalla) | 5 | 10 |
| F | 141 | Mostrar informació detallada (pantalla completa) | 5 | 8 |
| Afegir carregador | 21 | F | 142 | Interacció botó afegir publicació | 1 | 1 |
| F | 105 | Recaptació informació punt de càrrega (formulari) | 8 | 15 |
| F | 143 | Guardar imatges a storage | 5 | - |
| B | 106 | Guardar informació al sistema | 7 | 10 |
| Eliminar carregador | 3 | F | 91 | Botó + interacció d'eliminar | 2 | - |
| B | 144 | Eliminar carregador de la base de dades | 1 | 5 |
| Obtenir punts de càrrega públic | 21 | B | 145 | Fer crida a l'API per tal d'obtenir els carregadors | 8 | 1 |
| B | 135 | Guardar una copia local per tal d'evitar caigudes o problemes amb l'API | 6 | 12 |
| B | 136 | Fer sistema de sincronització entre API i còpia local | 7 | 5 |
| Mostrar punts de càrrega públics | 13 | F | 147 | Mostrar mapa | 5 | 2 |
| F | 146 | Obtenir els punts | 4 | 4 |
| F | 94 | Dibuixar punts sobre el mapa (ja s'haurà fet el de privats) | 4 | 7 |
| Edició carregador | 8 | F | 95 | Formulari editar carregador | 4 | - |
| B | 130 | Guardar dades al sistema | 4 | 10 |
| Mostrar punts de càrrega privats | 13 | B | 108 | Obtenir els punts privats | 4 | 4 |
| F | 107 | Dibuixar punts sobre el mapa | 9 | 7 |
| **Gestió rutes** | Iniciar ruta | 13 | F | 131 | Dibuixar ruta sobre el mapa | 5 | 6 |
| F | 157 | Obrir ruta a google maps | 1 | 1 |
| F | 132 | Calcular ruta (distància, temps, etc.) | 8 | 7 |
| Mostrar ubicació actual | 8 | F | 96 | Mostrar ubicació actual (Geolocalització) | 8 | 10 |
| **Gestió reserves** | Reservar un carregador | 21 | F | 98 | Seleccionar i reservar un time slot | 12 | - |
| F | 97 | Mostrar time slots lliures | 6 | - |
| B | 148 | Guardar reserva al sistema | 1 | - |
| F | 99 | Notificar propietari | 2 | - |
| Eliminar reserva carregador | 5 | B | 100 | Actualitzar la reserva del carregador amb estat cancel·lat | 2 | 10 |
| F | 149 | Eliminar visualment la reserva | 3 | 3 |
| Veure les meves reserves | 8 | B | 150 | Obtenir les reserves | 2 | 8 |
| F | 101 | Mostrar les reserves | 6 | 8 |
| Modificar reserva | 13 | B | 110 | Guardar els canvis al sistema | 6 | - |
| F | 109 | Mostrar informació reserva | 7 | - |
| **Gestió cerca** | Filtrar Carregadors | 21 | F | 151 | Creació dels filtres sobre el mapa | 6 | - |
| F | 134 | Fer cercador d'ubicacions | 3 | - |
| F | 112 | Afegir recomanacions de direccions al cercador | 4 | - |
| B | 113 | Obtenir els carregadors tenint en compte els filtres passats | 4 | 8 |
| F | 114 | Mostrar resultats en format llistat (depèn de veure els punts de propietari) | 4 | - |
| F | 156 | Bottom bar amb botó de cotxe si bicicletes + icono xat | 4 | - |
| Ordenar punts de càrrega | 13 | F | 116 | Mostrar possibilitats d'ordre | 7 | - |
| B | 115 | Obtenir els carregadors tenint en compte l'ordre passat | 5 | - |
| Veure els meus punts (propietari) | 13 | B | 117 | Obtenir punts de l'usuari loguejat | 5 | - |
| F | 152 | Veure llistat dels meus punts (propietaris) | 8 | - |
|  |  |  |  |  | **TOTAL** | **227** | **208** |

## 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** | **2** |
| --- | --- |
| **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** | **3** |
| --- | --- |
| **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** | **4** |
| --- | --- |
| **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** | **5** |
| --- | --- |
| **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** | **9** |
| --- | --- |
| **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** | **10** |
| --- | --- |
| **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. * El producte només ha de revelar informació privada d’acord amb la política d’informació de l’empresa. * S’encripta tota la informació sensible amb claus privades i no es mostraran públicament |

### Requisits de rendiment

| **Número** | **9** |
| --- | --- |
| **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, a partir dels següents sprints es senyalarà a quines històries del Taiga es tenen en compte els requisits no funcionals anotats anteriorment i com es mesuren.

## 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 en un sprint futur on ja tinguem un producte més complet mostrar-li a algun possible stakeholder per tal que ens doni feedback.
* **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 Actual

* **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**: No s'ha consultat cap stakeholder real (no ens contem), ja que creiem que ens trobem en una fase d'implementació inicial.
* **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 | ✔ |

## Serveis

La negociació ha seguit sent mitjançant el Whatsapp i s’han anat acabat tancant certs dubtes sobre les funcionalitats.

Com bé havíem dit en la ultima fase de la incepció ja es van negociar les funcionalitats que volem i ens donàvem.

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.

Actualment (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 aquest grup 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.



# Agile ceremony

En general quedem tots satisfets de tot el que s’ha treballat durant aquest primer *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.

Com a tot projecte, sempre hi ha aspectes a millorar. De bon principi ens vam proposar realitzar tests per a cada funcionalitat del sistema, però en aquest sprint no n’hem pogut realitzar cap. De cara al segon sprint haurem de considerar què fem en aquest aspecte.

També hem pogut veure que les hores que teníem proposades com a càrrega esperada de les històries ha sigut sempre més temps en realitat. Per tant, es considerarà també la granularitat d’aquestes històries en més o menys tasques per tenir encara més clar de bon principi la càrrega que hi ha darrera d’una funcionalitat.

## 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

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 3. Sprint burndown

### 3.2.2. Release Burndown

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 4. 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 en aquest sprint i els punts que finalment hem realitzat. Com veiem no hem arribat a acomplir-los tots però no ens preocupa ja que, com s’ha explicat prèviament, som conscients que ens havíem posat molts punts. Al següent sprint intentarem posar-nos al dia dels punts que haguem de tenir fets abans de començar el tercer sprint.

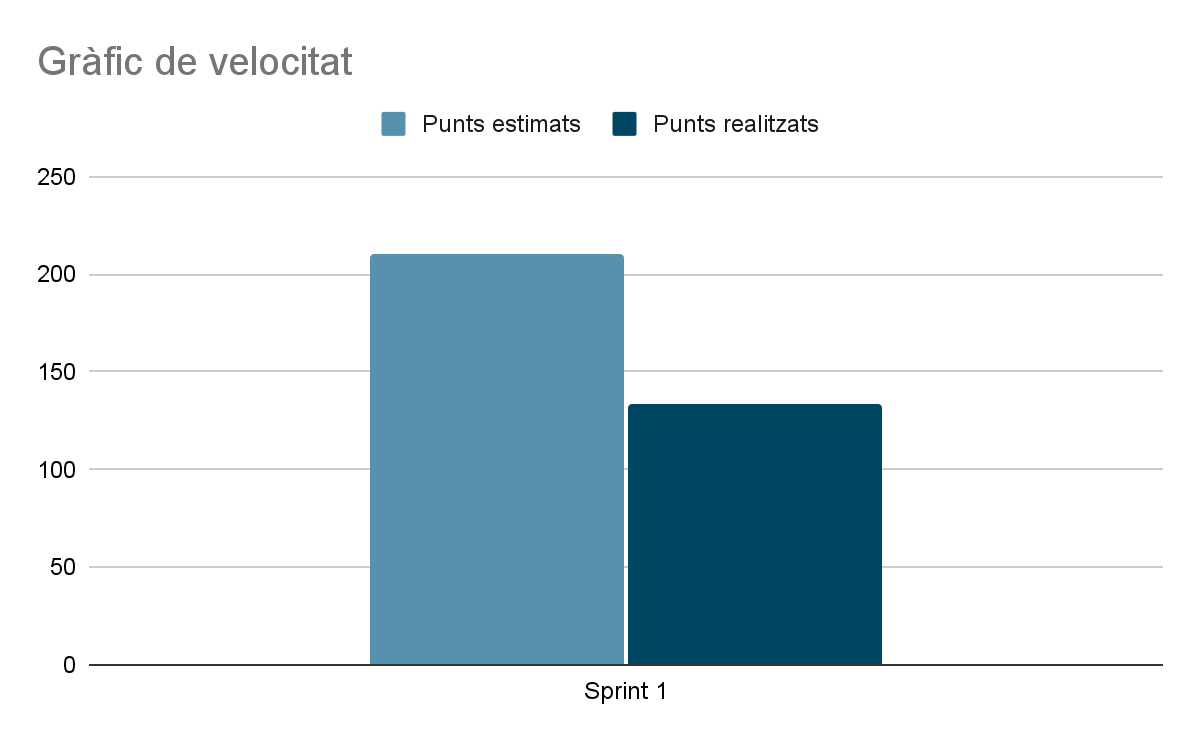


Figura 5. Velocity chart

### 3.2.4. Effort chart

Per últim mostrarem un gràfic que representa l’esforç estimat i l’esforç realitzat. S’han eliminat els punts corresponents a les tasques que no s’han pogut fer en aquest sprint del total de punts de l’esforç estimat. Com podem veure hem sobrestimat les nostres habilitats, hem fet trigat 72 hores més (1 punt equival a 1 hora) de les que teníem previstes. També hem de tenir en compte que en aquest primer sprint tots hem invertit temps en aprendre com funcionava tant Django com Flutter, al següent sprint ja podrem anar directament a fer les tasques com a tal.

### GràficFigura 6. 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, aquestes 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.

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.

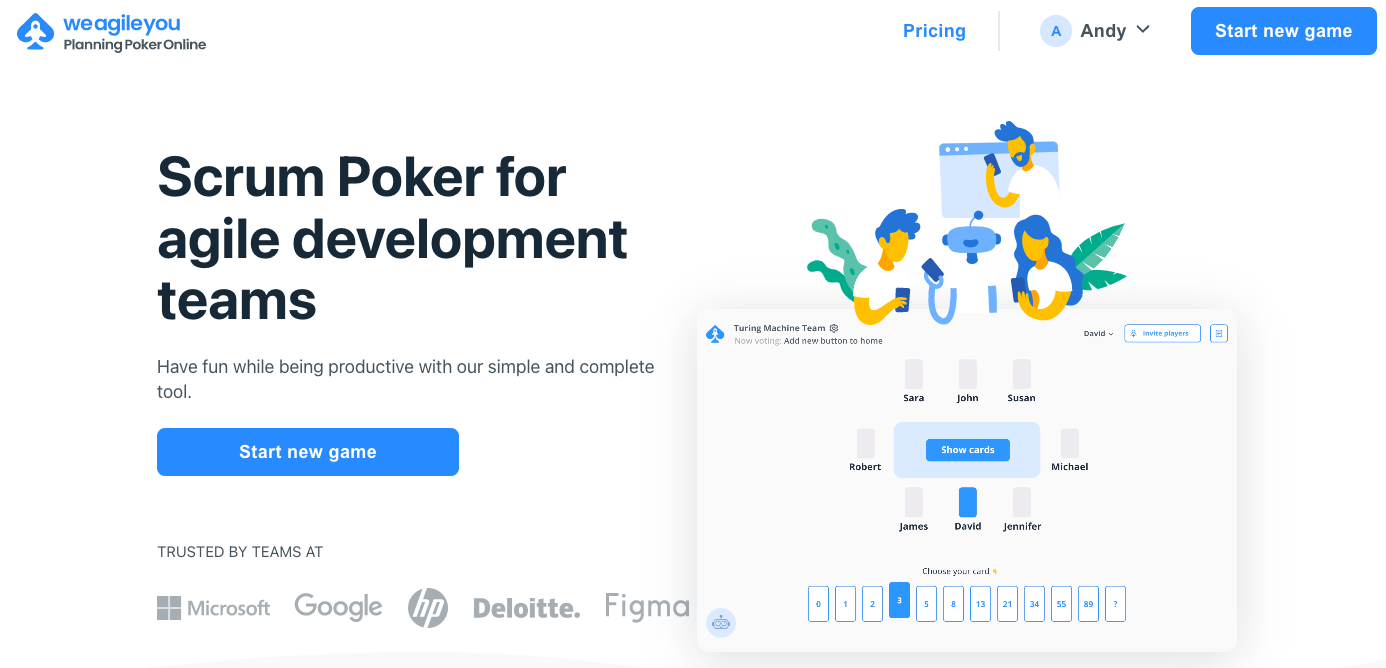
Pel que fa als *sprints*, ja hem decidit que setmanalment cadascun dels membres del grup tindrà diverses tasques a realitzar. Aquestes poden tenir relació a una història d’usuari i, per tant, aniran relacionades amb el desenvolupament de l’*App*, i d’altres poden ser tasques externes relacionades amb la documentació. 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 hàgem revisat tots els detalls que podrien faltar o els aspectes que podríem millorar, així com els errors que s’hagin pogut cometre. 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*.

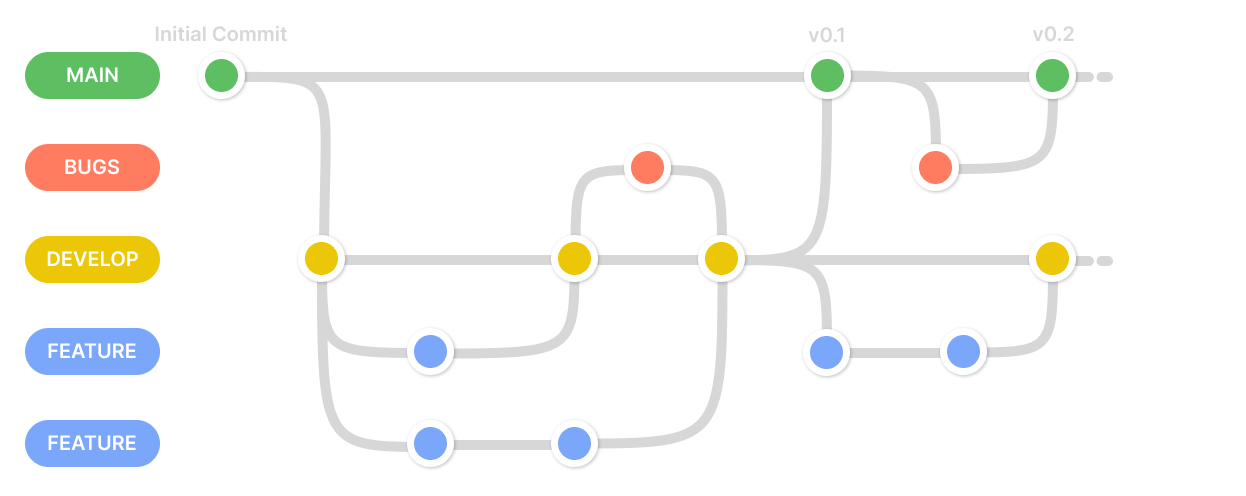
Pel que fa al cicle de vida d’una història d’usuari, aquesta seguirà les fases que ens determina la metodologia *Scrum* al *Taiga*: Quan una història creada s’assigna a un *sprint* determinat es defineix com a nova (*New*). Un cop s’assigna a un membre de l’equip i aquest comença a treballar en alguna 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 tots els tests unitaris d’aquella història d’usuari s’han superat i, per tant, 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:

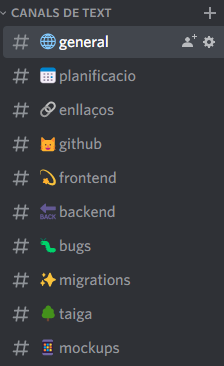
* Què he fet des de l’última reunió: On cada membre del grup explica quins progresos ha fet des de l’última reunió.
* Què té pensat fer fins a la pròxima reunió: On cada membre del grup explica que te planificat en els pròxims 3-4 dies i si necessita qualsevol cosa dels altres.
* Quines dificultats s’ha 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 permés 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 9. 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*.

Ademés, 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 (mitjançant Continuous integration) de manera en que, per cada commit, es faci la comprovació i ens pugui donar els resultats. La idea és utilitzar sonarQube o algun per l’estil, que no tingui necessitat de self hosted, ja que és un problema haver de allotjar una tecnologia més, degut a la poca facilitat per obtenir servidors de manera gratuita.

Per últim, tots els companys estem treballant a una, i veiem els codis dels companys continuament 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 comentar les diferents funcions utilitzant l’estructura de javadoc.

## Gestió de proves

Per a fer els tests del nostre sistema, ja siguin els unitaris o els d’integració, els farem 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 contínua (CI), farem servir l’apartat d’*Actions* dels nostres repositoris de *GitHub*. 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.

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 algun 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.

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.

Per altra banda, en local treballarem amb el fitxer *.env*, on 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 que ens anava millor el servei de contaminació i polució 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 haviem 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 i es manté tot l'esmentat en la inception anterior.

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

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.

En el segon sprint acabarem de concretar exactament a 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. Per altra banda, amb el grup de la polució, la comunicació ha estat més complciada, ja que a part de que el representant del seu grup per tal de gestionar els serveis externs mai assistia a classe, era complicat contactar amb ell per whatsapp, fent una comunicació lenta i poc clara.

Tot i així, finalment hem pogut tancar els diferents serveis sense gaires problemes.

## 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.

## 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, perque s’està col·locant allà l’element, etc. També ens ofereix eines d’anàlisi del rendiment com el *Profiler*, el qual ens dona, 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…).

# 

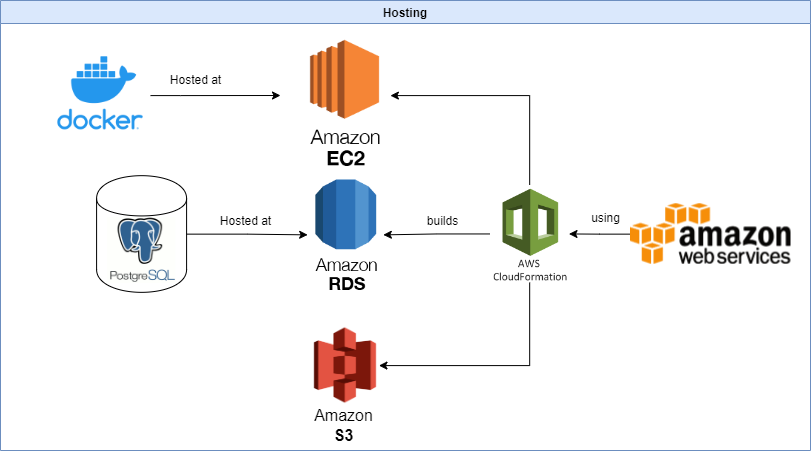
# Descripció tècnica

## 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 instància 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.

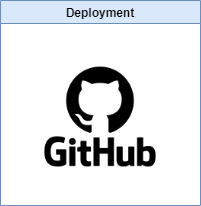
Per a crear un contenidor on corri la nostra aplicació *Docker*:

* **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 vàries 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 tindrien 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 escollíssim 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.

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 necessitar 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.

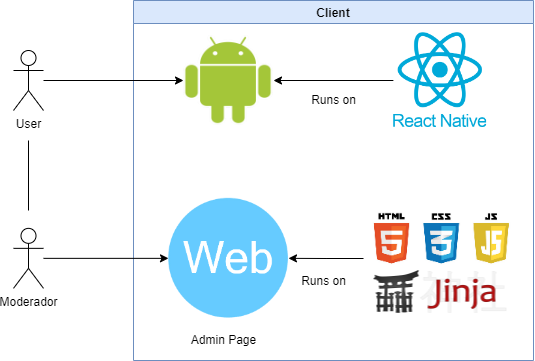
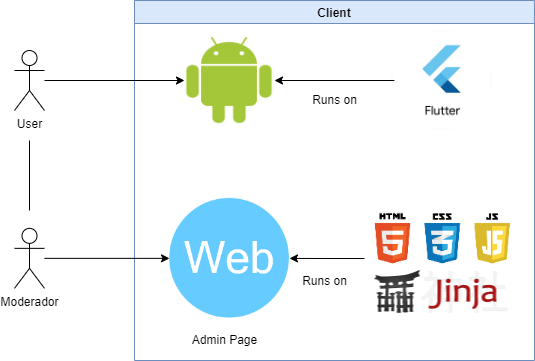
Les aplicacions de les quals obtenim dades o utilitats són:

* **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.

### 

### 

### 

### 

### 5.1.4.1 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 construiríem 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 totrnar 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 vàries 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.

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.

### 5.1.4.2 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 tecnologia 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.

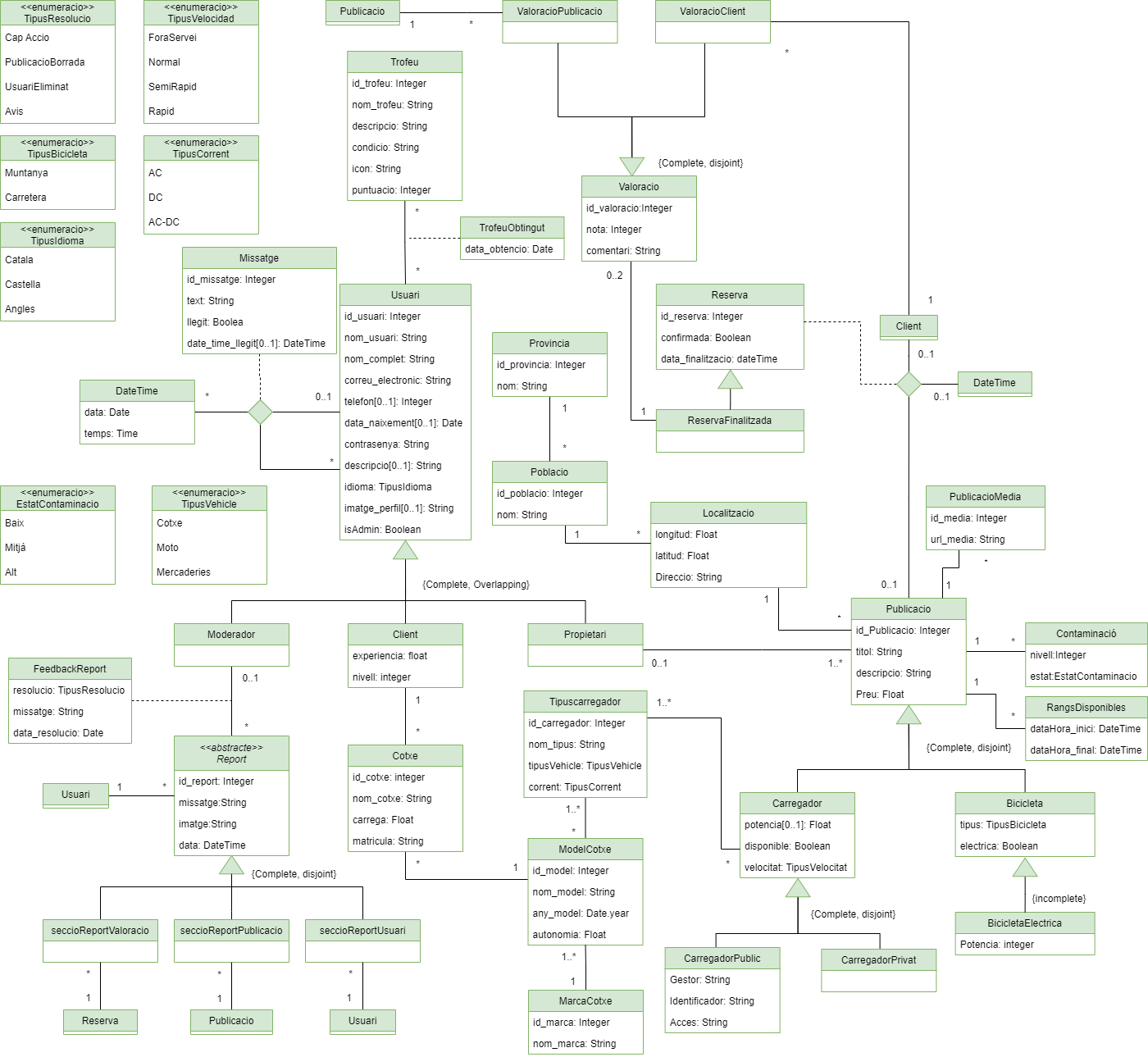
El nombre d’aplicacions que es fan servir, sobretot en *frameworks* de *JavaScript*, és molt gran, volem destacar dues llibreries i serveis que ens són de gran ajuda per a la realització del projecte.

* ***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.

## 

## Diagrama de classes UML

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 classes.



**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.

### 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:

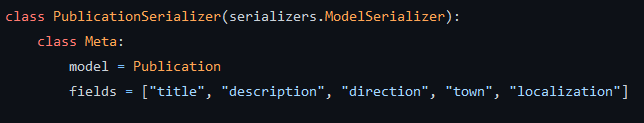
* **Adaptador:** Hem utilitzat aquest patró per tractar la informació dels objectes. Tant la informació obtinguda de les diferents taules són transformades a un format JSON per poder tractar-les en diferents funcionalitats. A més a més, s’han implementat serializers que també fan d’adaptador de les dades, 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 19: Mostra del serializer implementat per a una publicació

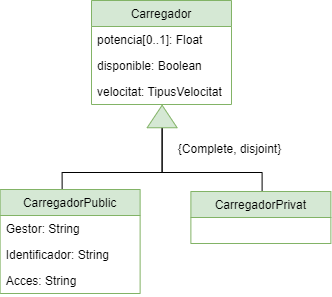
* **Plantilla:** L’ús d’aquest patró el portem a terme per poder encapsular mòduls complexes desenvolupats en l’aplicació. Hi ha metodologies que tenen una funcionalitat molt semblant, però que es diferencien en petits detalls, per tant, es crea una superclasse corresponent a l'objecte comú i es desenvolupa 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 20: Mostra del UML en relació amb el patró plantilla

## Diagrama de Base de dades

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 21: Diagrama de Bases de dades

## 

## Mockups

A continuació mostrem els *mockups* de les pantalles de la nostra aplicació on veiem com es poden realitzar les principals funcionalitats del nostre sistema:

## 5.4.1. Inici de sessió



Aquesta pantalla serà la primera pantalla que es veurà quan un usuari es descarregui l’aplicació o bé tanqui la sessió.

En aquesta pantalla es pot iniciar sessió ja sigui introduint un correu i una contrasenya o bé iniciant sessió amb un compte de *Google* o del Racó de la FIB.

També et permetrà regenerar la contrasenya en el cas que l’usuari l’hagi oblidada i redirigir-te a una altra pantalla en la qual l’usuari es podrà crear un compte en el cas que no en tingui.

## 5.4.2. Registre usuari

Figura 22: Pantalla d’inici de sessió

Aquesta és la pantalla a la qual es redirigeix l’usuari quan selecciona l’opció de crear-se un compte. Com es pot veure s’ha d’introduir un nom d’usuari, un correu i una contrasenya i seleccionar “Siguiente”.

En el cas que l’usuari s’hagi equivocat en seleccionar crear compte i ja tingui un compte podrà seleccionar “Inicia sesión” i se’l retornarà a la pantalla d’inici de sessió.

L’usuari podrà també crear-se un compte de l’aplicació utilitzant un compte de *Google* o del Racó de la FIB.

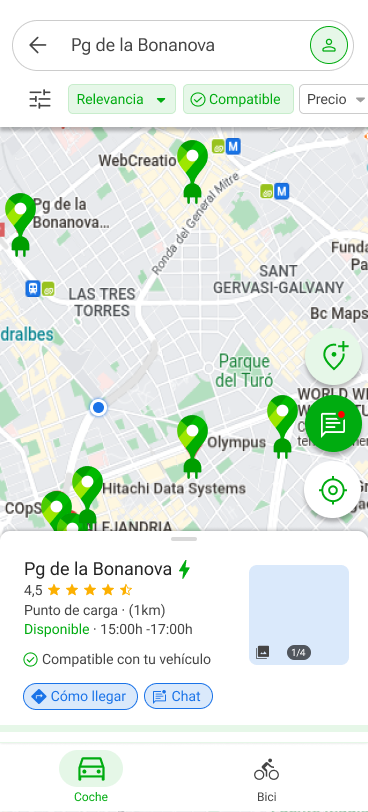
## 5.4.3. Pantalla mapa

Aquesta és la pantalla principal de l’aplicació amb el mapa on es mostraran els diversos punts. A dalt de tot es troba la barra de cerca amb un accés al perfil.

A la part de sota podem veure dues opcions: Cotxe i bici. No són dues opcions excloents, és a dir, si l’usuari selecciona Cotxe es mostren tots els punts de càrrega propers, si selecciona Bici es mostren tots els punts de lloguer de bici de la zona i si els selecciona els dos es mostren ambdós tipus.

A sota a la dreta veiem un botó d’ubicació el qual en seleccionar-lo enfoca el mapa a la localització actual de l’usuari. Just sobre del botó esmentat hi ha un altre botó el qual et redirigeix a la pantalla de xat, la qual mostrarem més endavant.

## 5.4.4. Punts

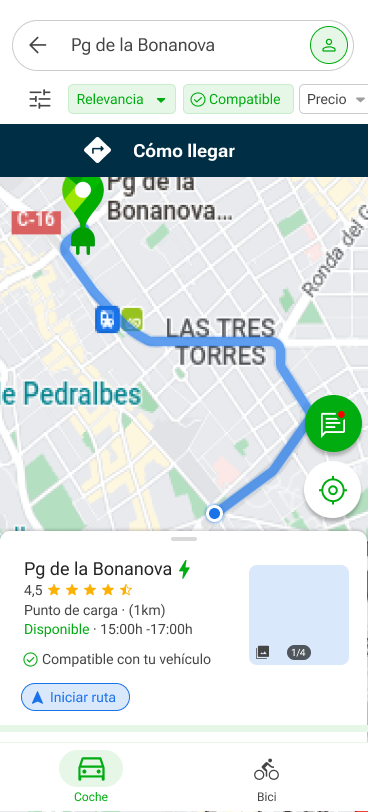


Aquesta és la pantalla que veurà l’usuari en posar una direcció a la barra de cerca. Quan un usuari busca una direcció es mostren tots els punts de càrrega propers a la direcció buscada. Es podrà filtrar els punts que es mostren per rellevància ja sigui de la distància, del preu o, en cas dels punts de càrrega, de si l’endoll és compatible o no amb el cotxe de l’usuari.

En seleccionar un punt apareixerà la informació d’aquest. Amb la localització exacta on es troba, la distància a la qual es troba de l’usuari, i en el cas dels carregadors si és compatible o no amb el cotxe, la disponibilitat que té, imatges per veure el seu estat i per últim es podrà seleccionar el botó de “Como llegar” el qual marcarà una ruta en seleccionar-ho i un botó de xat en el cas que el l’usuari es vulgui posar en contacte amb el propietari.

## 

## 5.4.5. Ruta

Aquesta pantalla mostra com es veurà quan l’usuari seleccioni un punt i del mapa i a continuació prem-hi al botó d’anar-hi.

Com podem veure es traça una ruta entre l’origen i el destí per anar d’un punt a l’altre.

El botó superior de “Iniciar ruta” redirigirà a l’usuari a *Google Maps* per a que pugui veure les diferents opcions de com arribar-hi que proporciona

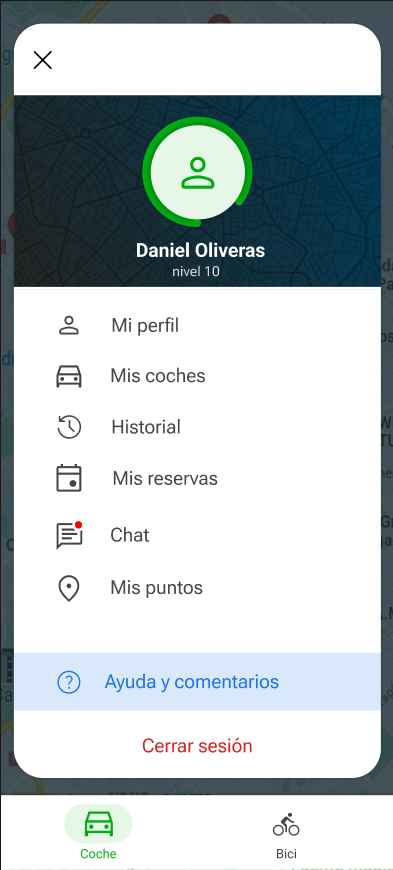
## 

## 

## 

## 

## 5.4.6. Pestanya perfil



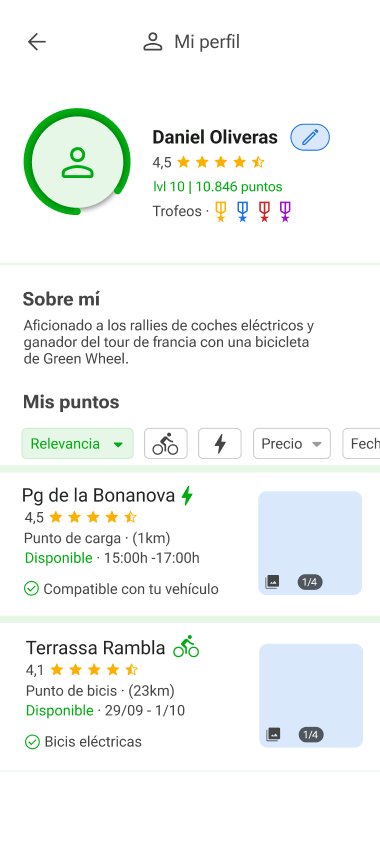
En aquesta pantalla mostrem el perfil de l’usuari, on prèviament s’ha d’haver clicat al botó pertinent al cercador de la pantalla inicial.

A la part superior trobem la imatge del perfil de l’usuari amb el seu nom i el nivell respectiu en l’aplicació. Per saber en quin estat es troba per passar de nivell veiem que al voltant de la imatge tenim un indicador.

A sota trobem 8 botons que permeten certes funcionalitats a l'usuari. El primer de tots, el de la part superior, permet accedir a la visualització del perfil de l'usuari per tal de fer les gestions pertinents. En el segon permet registrar els cotxes que desitgi per tal de compatibilitzar-ho amb l’aplicació i oferir una millor experiència. En el tercer tenim l’historial dels punts que hagi cercat recentment. En el quart mostrem les reserves que tingui vigents l’usuari. En cinquè lloc, tenim el botó de xat per iniciar-ne un amb un altre usuari de l’aplicació. El sisè permet accedir als punts que l'usuari hagi donat d'alta a l'app.

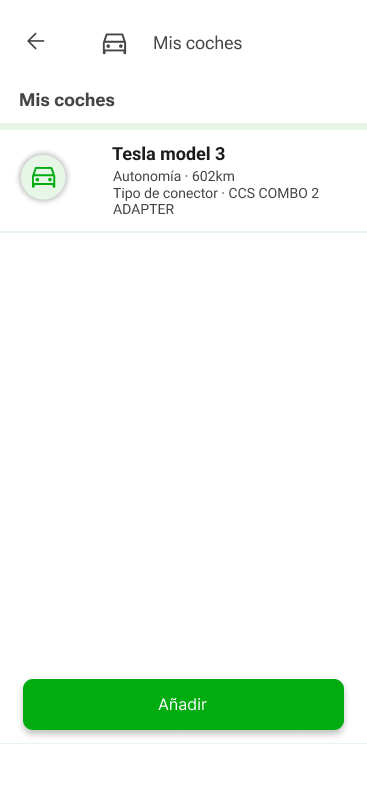
El botó blau ofereix ajuda en l’aplicació a través d’una guia per facilitar el funcionament d’aquesta a l'usuari i si fos el cas deixar algun comentari al respecte. També trobem el botó de tancar sessió.

## 5.4.7. El meu perfil

Aquesta és la pantalla del perfil de l’usuari. En ella es pot veure tota la informació respecte a l’usuari: el nom, la valoració, que es basa en la puntuació mitjana que altres usuaris li han posat, el nivell que té (el nivell s’aconsegueix realitzant objectius com podria ser reservar un punt o oferir un punt), una petita descripció sobre qui és i a continuació en el cas que l’usuari sigui propietari d’algun punt apareixeran aquests amb tota la seva informació.

## 

## 5.4.8. Els meus cotxes



## 

Aquesta és la pantalla on es poden veure els cotxes que l’usuari té registrats al seu compte. Pot tenir-ne diversos de registrats i es poden afegir seleccionant el botó d’Afegir que es troba a la part inferior de la pantalla.



## 5.4.9. Historial

En aquesta pantalla es mostrarà l’historial de l’usuari, és a dir, aquells punts pels quals ha passat recentment i de cada un podrà veure tota la seva informació. També té un botó per iniciar un xat, un per veure com hi pot arribar, un altre per valorar el punt i un altre per reportar-lo, en el cas que l’usuari no hagi tingut una bona experiència amb el punt. 

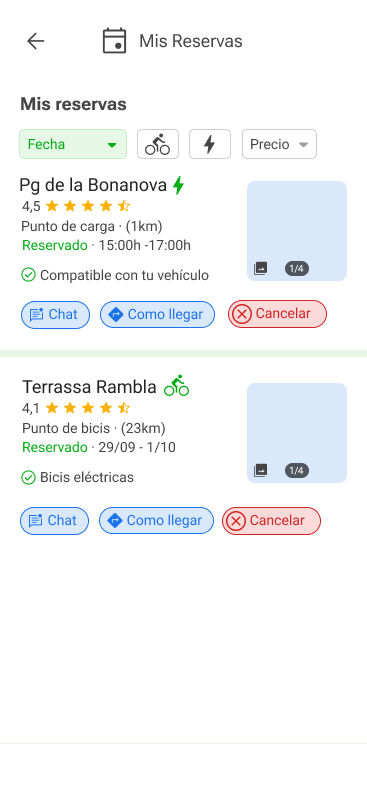
## 

## 

## 

## 

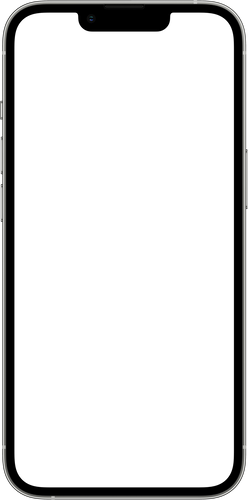
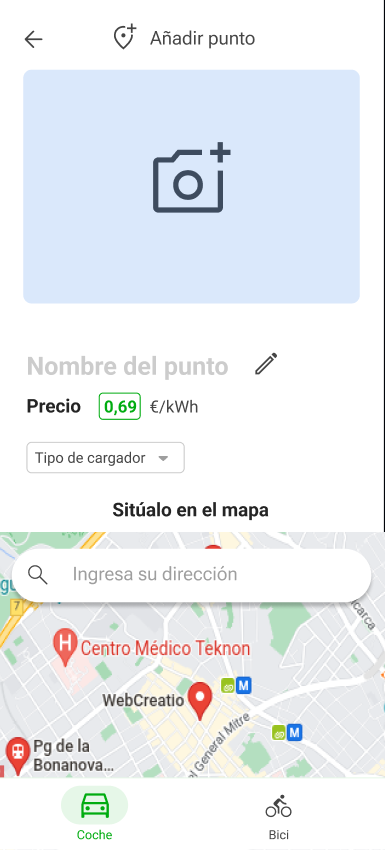
## 5.4.10. Les meves reserves



En aquesta pantalla es mostraran totes les reserves que l’usuari tingui d'ara endavant, en ella pot veure tota la informació del punt de càrrega o bici que hagi reservat. També té un botó per iniciar un xat, un per veure com hi pot arribar i un altre per cancel·lar la reserva.

## 

## 5.4.11. Afegir punt



En aquesta pantalla l'usuari pot afegir un punt nou proporcionant la informació necessària. El punt, a més d'un nom, preu, adreça i tipus, tindrà imatges que serviran a la resta d'usuaris per consultar-ne l'estat.

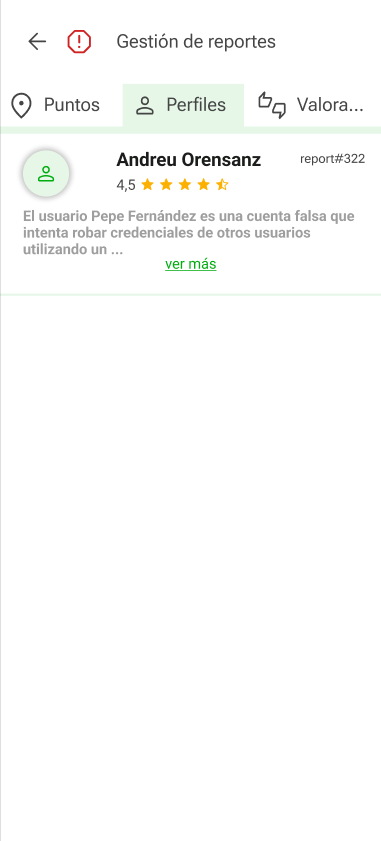
## 

## 

## 

## 

## 5.4.12. Pàgina d’administració de reports



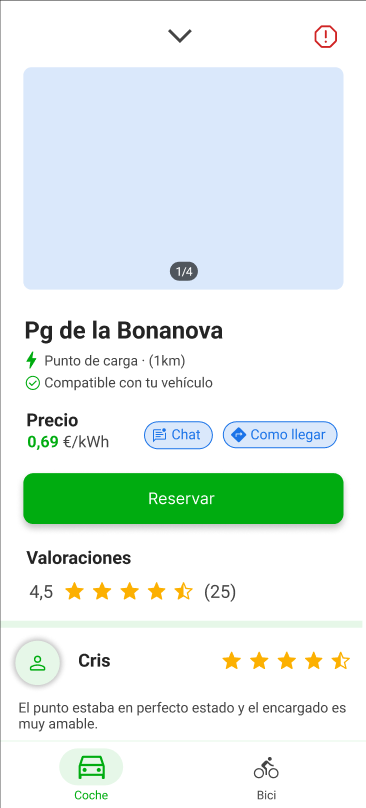
Els administradors podran gestionar els reports des de la pantalla que es mostra a la figura 12. Hi trobem tres seccions on es troben els diferents tipus de reports classificats.

Comptem amb una secció per als reports dels punts, una altra per als usuaris i una darrera per a valoracions.

Cada report es mostra amb l’usuari que l’ha fet a més d’una descripició y una id per identificar-lo.

## 

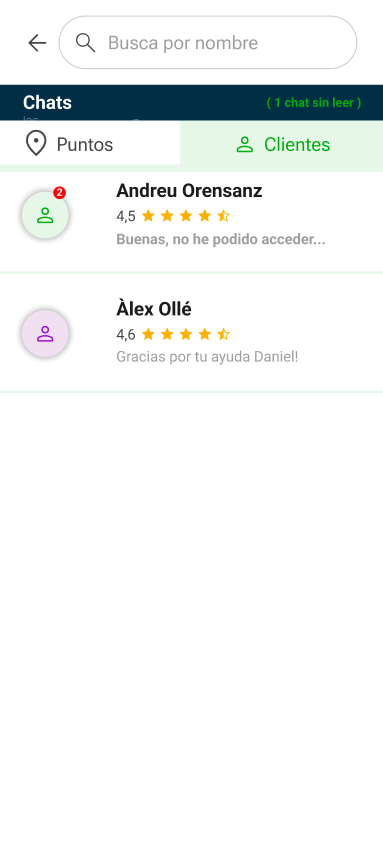
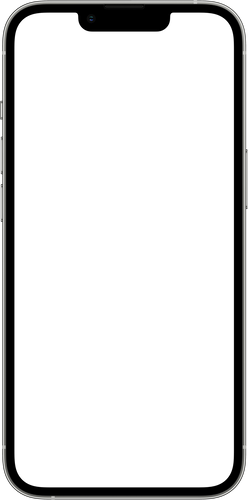
## 5.4.13. Visualitzar punt



Quan un usuari vol veure més informació d'un punt, accedeix a aquesta pantalla. Aquí s'hi mostra, a més de la informació general que apareix en altres pantalles, les valoracions, el preu i se li dóna accés a la funcionalitat de reserva. Si l’usuari hagués de reportar el punt té l’opció de fer-ho fent tap a la part dreta de dalt.



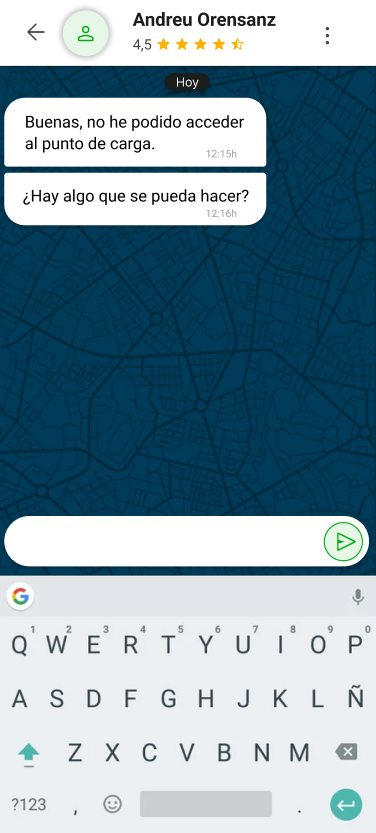
## 5.4.14. Xats



En aquesta pantalla l’usuari té accés als diferents xats oberts. Es mostren classificats en “Clientes” i “Puntos” per diferenciar els xats on l’usuari n’es el arrendador o arrendatari . En la part superior veiem una barra de cerca per a buscar un xat en específic.

## 

## 5.4.15. Xat

El xat integrat a l'aplicació permetrà enviar i rebre missatges de text entre usuaris per facilitar i agilitzar la comunicació.

Des d'aquesta pantalla, l'usuari també podrà reportar l'usuari accedint a les opcions dels tres punts.

## 



## 

## 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*:

## 5.1. 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.

## 5.2. Usuaris que lloguen bicis

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

## 5.3. 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.

## 5.4. 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.

## 5.5. Administrador/moderador del sistema

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

## 5.6. Equip de desenvolupament

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

## 5.7. Competència

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

## 5.8. Generalitat

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

## 5.9. Altres equips de PES

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

## 5.10. *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.

## 5.11. *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.

## 5.12. *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.

## 5.13. El Racó

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

## 5.14. *Github*

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

**5.15. *AWS***

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

# 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>