**t**

**INCEPTION: DOCUMENTATION, 2nd PHASE**

[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**

[**Sprint master report**](#_qaggx69xr3nl) **3**

[**Abast del projecte**](#_wskxy55ll79d) **4**

[**Stakeholders**](#_2t2ap6w5flt2) **6**

[**Èpiques i històries d’usuari**](#_gga323156vpi) **8**

[**Mockups**](#_m12oecygd7ra) **1**0

[**Diagrama arquitectura física**](#_f6me26wgsaj9) **18**

[6.1 Hosting](#_76safm4yudln) 18

[6.2 Desplegament](#_vpb51bnne8wr) 20

[6.3 Server](#_2ogyp8t26gz) 21

[6.4 External services](#_i31n9ll8fu2e) 22

[6.5 Client](#_c33j6fv4de67) 23

[6.6 Llibreries externes importants](#_b5bnwxtrrn0) 24

[**Diagrama de classes UML**](#_ftdl1hm2fime) **25**

[**Metodologia**](#_4hvukngrq3em) **27**

[8.1. Visió general](#_8opmvv6sn3mv) 27

[8.2. Gestió projecte](#_erkaocdkhkbr) 28

[8.3. Gestió versions](#_lezr5ia1vc41) 29

[8.4. Comunicació equip](#_66nben96m52y) 31

[8.5. Gestió qualitat](#_f9z1eu9ri364) 31

[8.6. Gestió proves](#_ifmtpbyuqlr2) 31

[8.7. Gestió configuracions](#_mq56jow0047v) 32

[8.8. Interacció amb altres grups](#_3ygy22a12fm9) 33

[8.9. Gestió bugs](#_vfvwzlm0h7l8) 33

[8.10. IDEs](#_ojeatj1o7xex) 33

[8.11. Llenguatges](#_dlzgkzh8n9q0) 33

[8.12. Gestió bases de dades](#_3r8853i25ozq) 36

[**Serveis**](#_3mny5knhbuso) **37**

[**Referències**](#_prupja6a9c1f) **38**

# Sprint master report

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*.

# Abast del projecte

## 2.1. 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 | |

# 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*:

## 3.1. 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.

## 3.2. Usuaris que lloguen bicis

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

## 3.3. 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.

## 3.4. 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.

## 3.5. Administrador/moderador del sistema

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

## 3.6. Equip de desenvolupament

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

## 3.7. Competència

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

## 3.8. Generalitat

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

## 3.9. Altres equips de PES

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

## 3.10. *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.

## **3.11. *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.

## 3.12. *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.

## 3.13. El Racó

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

## 3.14. *Github*

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

**3.15. *AWS***

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

# Èpiques i històries d’usuari

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 |

# 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**.1.** Inici de sessióOk



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.

## 5.2. Registre d’usuari

Figura 1: 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.

Figura 2: Pantalla de registre d’usuari

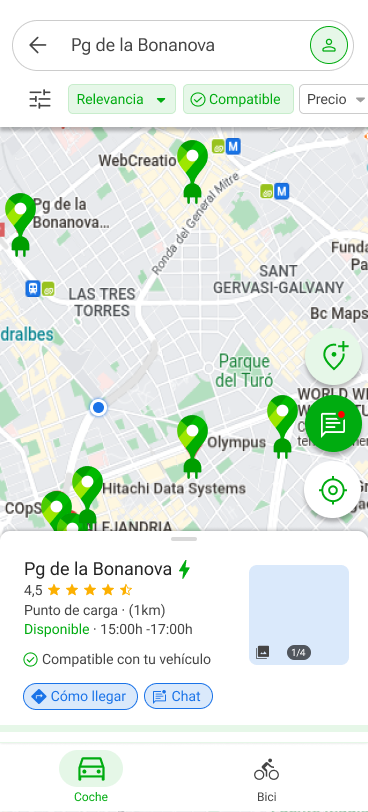
## 5.3. Pantalla mapa

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. Punts

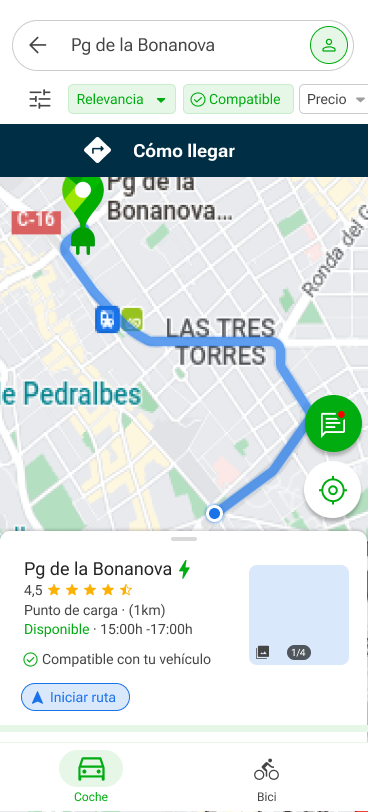


Aquesta és la pantalla que veurà l’usuari en posar una direcció a la barra de cerca. Quan un usuari busca una direcció es mostren tots els punts de càrrega propers a la direcció buscada. Es podrà filtrar els punts que es mostren per rellevància ja sigui de la distància, del preu o, en cas dels punts 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.

## 

## 5.5. 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

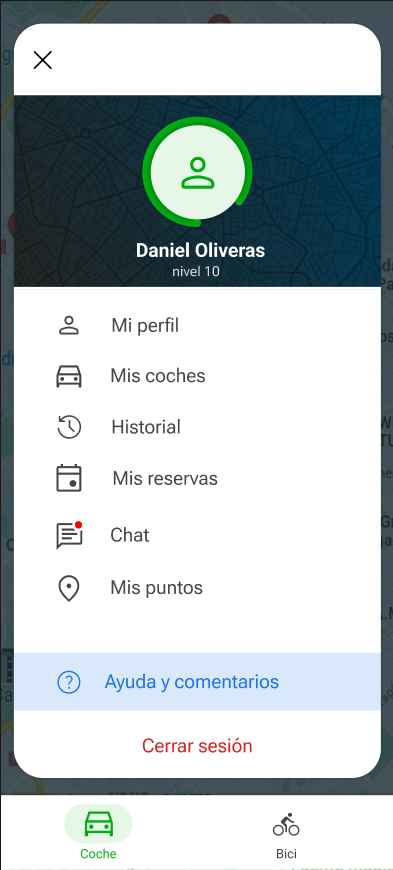
## 

## 

## 

## 

## 5.6. Pestanya perfil



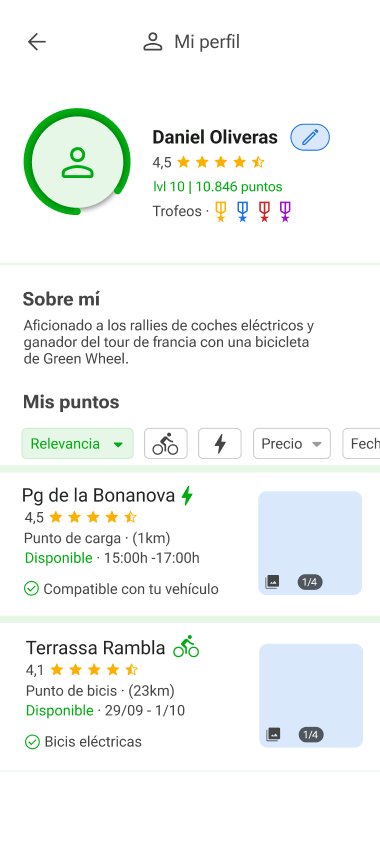
En aquesta pantalla mostrem el perfil de l’usuari, on prèviament s’ha d’haver clicat al botó pertinent al cercador de la pantalla inicial.

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.

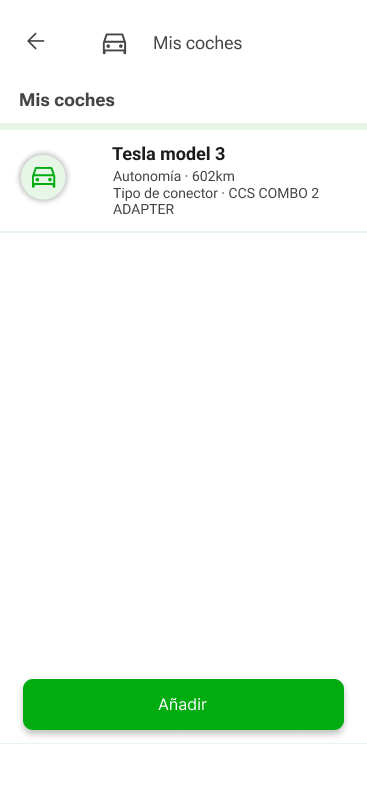
El botó blau, ofereix ajuda en l’aplicació a través d’una guia per facilitar el funcionament d’aquesta a l'usuari i si fos el cas deixar algun comentari al respecte. També trobem el botó de tancar sessió.

## 5.7. 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.

## 

## 5.8. Els meus cotxes



## 

Aquesta és la pantalla on e s 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.



## 5.9. 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. 

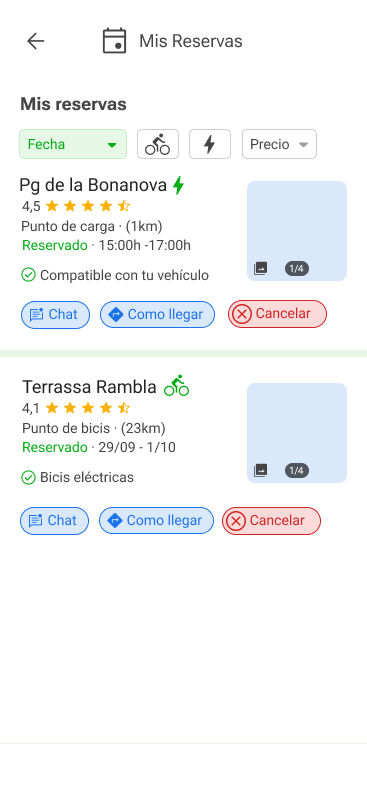
## 

## 

## 

## 

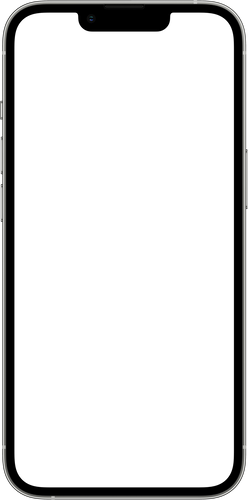
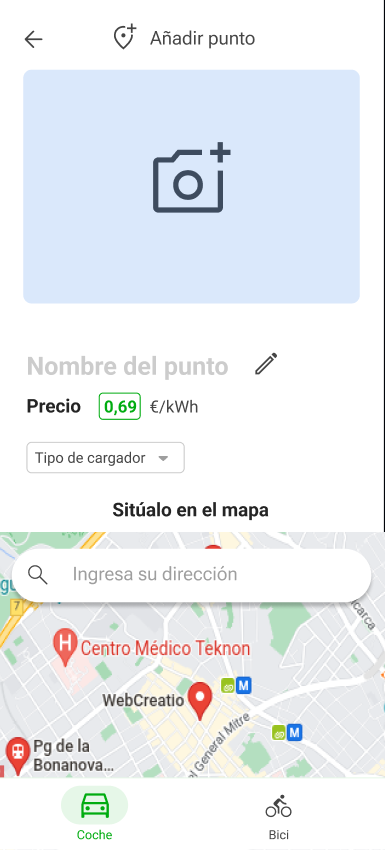
## 5.10. Les meves reserves



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.

## 

## 5.11. 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.

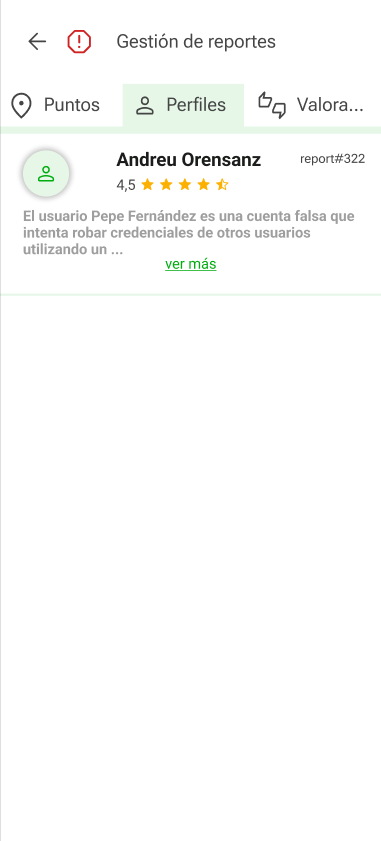
## 

## 

## 

## 

## 5.12. Pàgina d’administració de reports



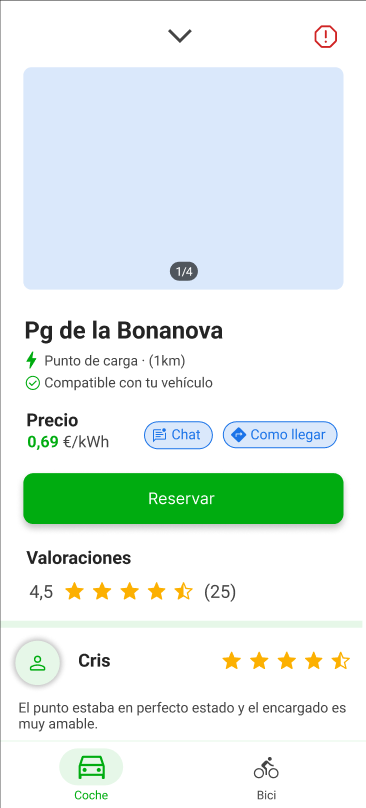
Els administradors podran gestionar els reports des de la pantalla que es mostra a la figura 12. Hi trobem tres seccions on es troben els diferents tipus de reports classificats.

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.

## 

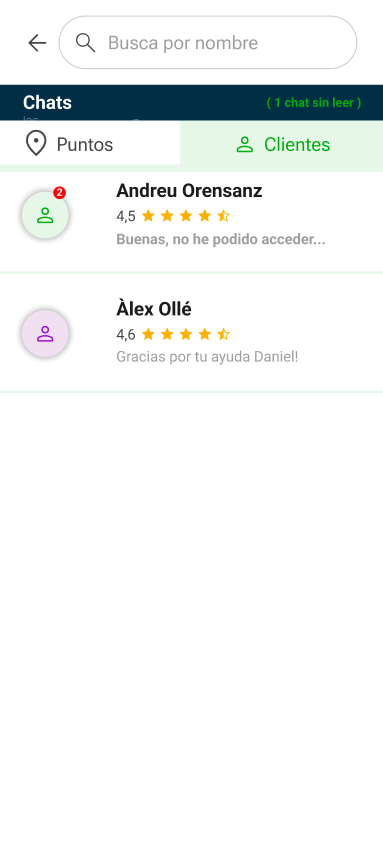
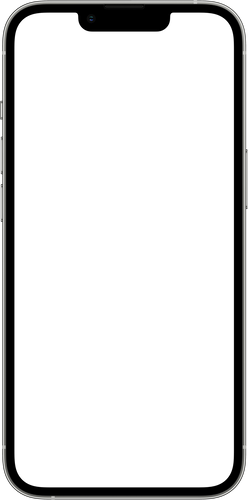
## 5.13. Visualitzar punt



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.



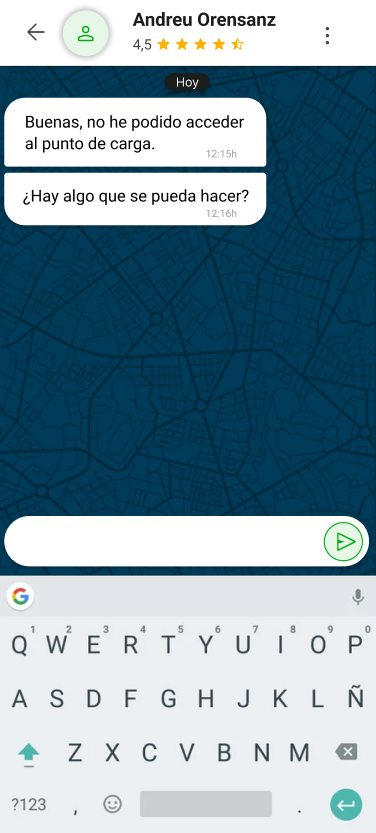
## 5.14. Xats



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.

## 

## 5.15. 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.

## 



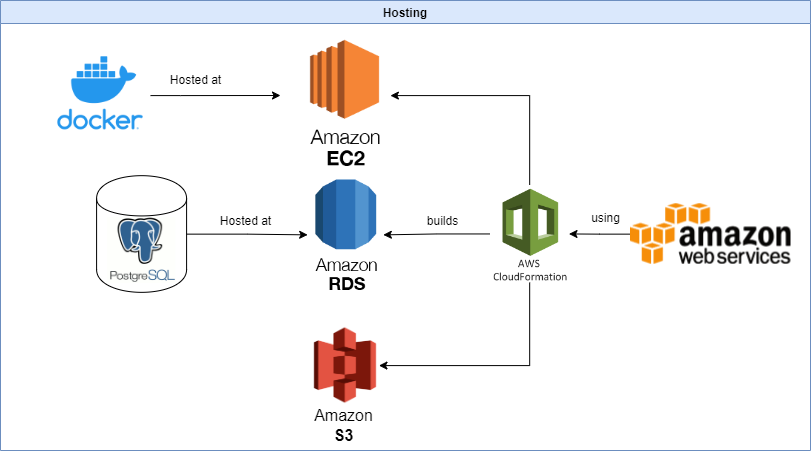
# 

# 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.

## 

## 6.1 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]***](#p9yra1ry95o5)o *docker swarm*[**[2]**](#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.

## 6.2 Desplegament

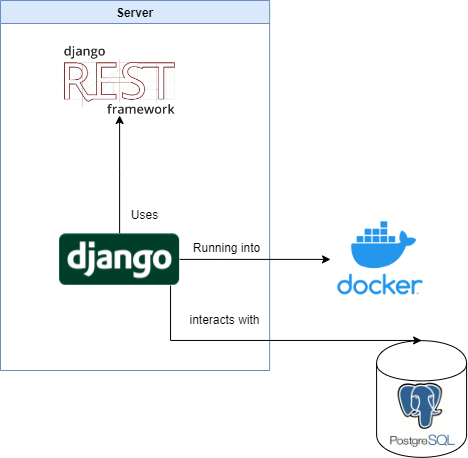
Per tal de desplegar el nostre codi a l’aplicació, teníem vàries opcions que ens eren vàlides i que ens podien servir per a fer la construcció, testejar i desplegar el servidor a *Amazon Web Services* en un *Docker*.

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.

## 6.3 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]**](#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.

## 6.4 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.

## 6.5 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.



### 6.5.1 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, 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.

### 6.5.2 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.

## 6.6 Llibreries externes importants

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

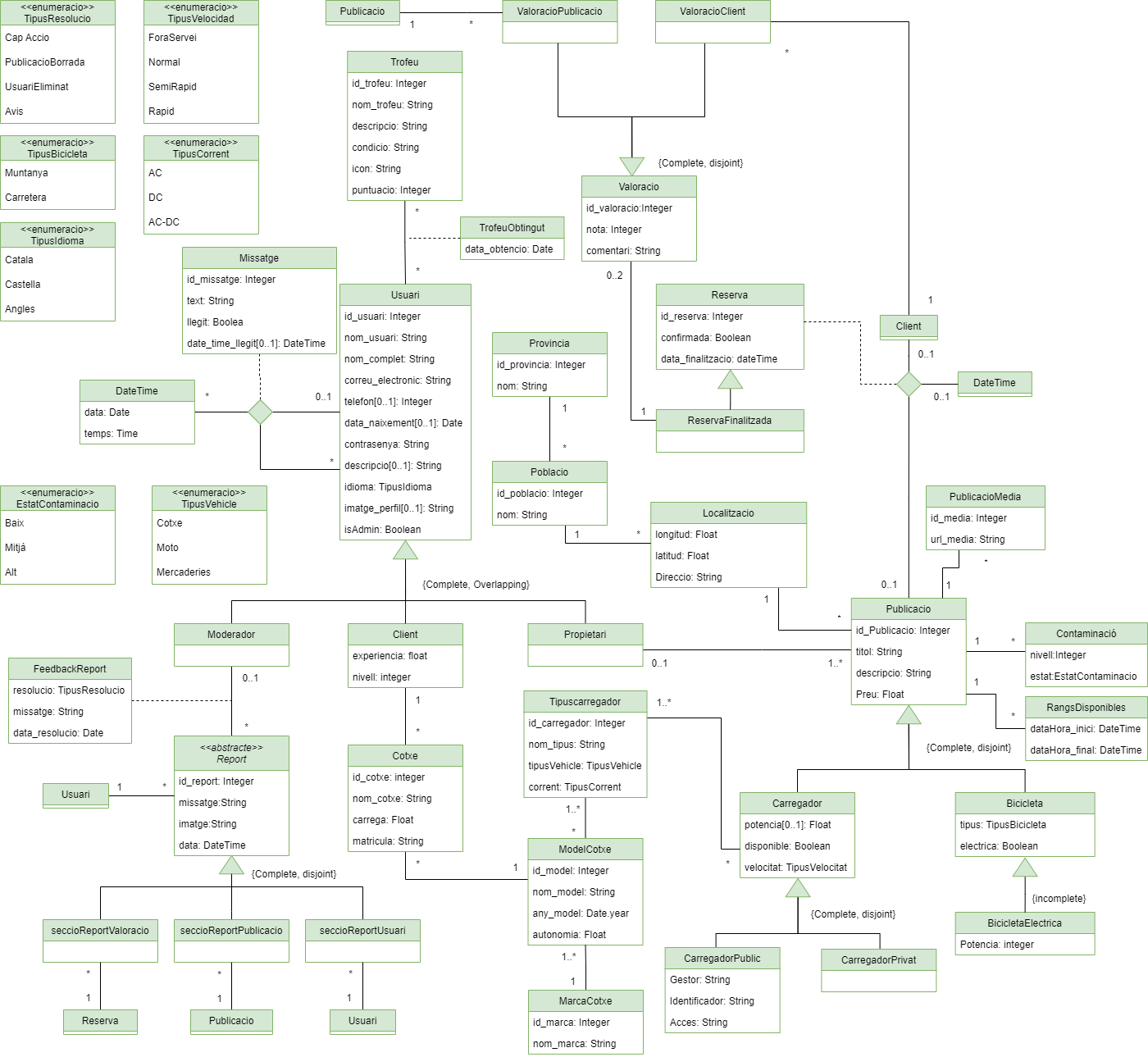
El nombre d’aplicacions que es fan servir, sobretot en *frameworks* de *JavaScript*, és molt gran, volem destacar dues llibreries i serveis que ens són de gran ajuda per a la realització del projecte.

* ***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.

# 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.

## 8.1. 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.

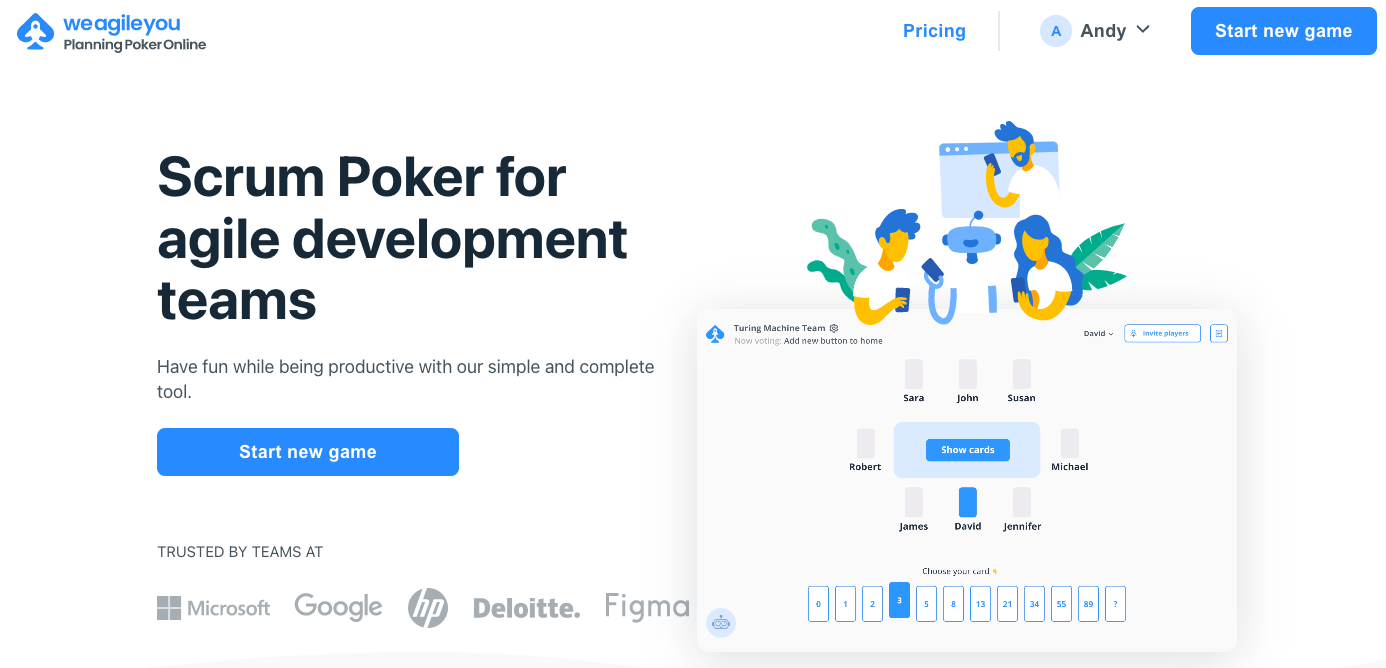
Pel que fa als *sprints*, ja hem decidit que setmanalment cadascun dels membres del grup tindrà diverses tasques a realitzar. Aquestes poden tenir relació a una història d’usuari i, per tant, aniran relacionades amb el desenvolupament de l’*App*, i d’altres poden ser tasques externes relacionades amb 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:

## 8.2. Gestió 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*.

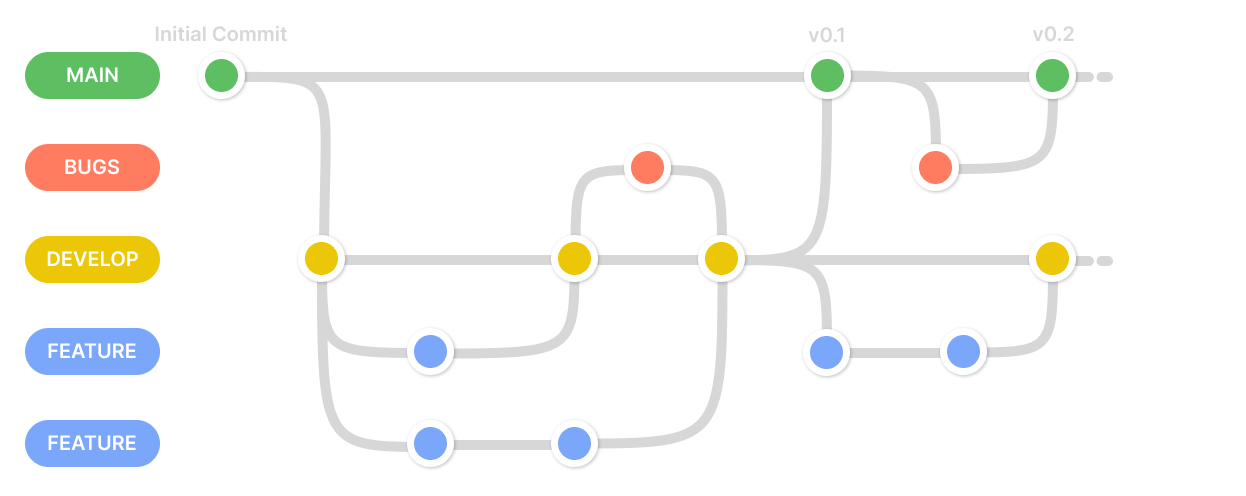
Pel que fa al cicle de vida d’una història d’usuari, aquesta seguirà les fases que ens determina la metodologia *Scrum* al *Taiga*: Quan una història creada s’assigna a un *sprint* determinat es defineix com a nova (*New*). Un cop s’assigna a un membre de l’equip i aquest comença a treballar en alguna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

## 8.3. Gestió 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-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.

## 8.4. Comunicació equip

El nostre equip té dos canals de comunicació:

* 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.

## 8.5. 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*.

## 8.6. Gestió 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 React Native*. 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 hàgem 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.

## 8.7. Gestió configuracions

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.

Aquest script 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 a la conta 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.

## 8.8. Interacció amb altres grups

La interacció amb els altres equips de PES sobre oferir i rebre uns serveis es realitzarà mitjançant l’aplicació WhatsApp. En cas que aquesta fallés utilitzaríem el correu de la universitat per a poder-se comunicar.

Per tal d'aconseguir comunicar quines funcionalitats volem i quines funcionalitats volen els altres equips que els oferim, s’han afegit els números de telèfons de cada representant d'interacció de grups i s'ha realitzat una negociació privada per cada equip. A més a més, també s’han aprofitat les estones a classe per poder parlar físicament amb els altres grups i poder tancar les funcionalitats.

La funcionalitat d’utilitzar WhatsApp com a eina principal és la facilitat de fer consultes ràpides i donar informació de manera senzilla i ràpida. Una altra raó molt important és la importància de tenir guardat escrit en el xat les funcionalitats finals i negociacions que realitzarem per tal que en cas que es presenti cap incongruència en el que dos grups havien acordat es pot recórrer al text per resoldre l’assumpte.

## 8.9. Gestió 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.

## 8.10. 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 *WebStorm* per a *React Native*.

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.

La cosa positiva és que en 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.

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.

## 8.11. 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, l’aplicació base s’executarà sobre dispositius *Android*, encara que *React Native*, el *framework* que hem elegit, ens permet fer-la multiplataforma. *React Native* està desenvolupat per *Facebook* i està realitzat amb *javascript*. Gràcies a la gran comunitat que té, compta amb milers de llibreries per a fer la vida més senzilla als i les programadores.

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.

### 8.11.1 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

### 8.11.2 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);

}

### 8.11.3 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

### 8.11.4 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.

## 8.12. Gestió bases 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…).

# Serveis

Com bé havíem dit en el punt 8.8 (Interacció amb altres grups), s'ha realitzat una negociació mitjançant WhatsApp i hem arribat a un acord sobre quines funcionalitats rebrem i donarem.

Primer de tot, quan es va comentar que havíem de donar i rebre funcionalitats amb altres grups, vam analitzar quines funcionalitats *API* ens interessarien incorporar a la nostra aplicació i llavors, veient els tipus d'aplicacions que oferien els nostres companys, vam veure que l’aplicació de la qual ens aniria millor rebre un servei seria la de contaminació.

Aleshores, amb aquesta idea ens vam posar a treballar i els vam enviar la nostra proposta. Els hi vam dir que ens interessava una funcionalitat per a saber la contaminació en una coordenada qualsevol. Finalment, la negociació amb el grup de contaminació va arribar a la fi quan ens van dir que no podien saber la contaminació d'una coordenada qualsevol, però sí que ens podien proporcionar la contaminació en una zona. Tot i no tenir tanta precisió com volíem també ens anava bé.

En conclusió, la funcionalitat que rebem del grup que treballa en els nivells de contaminació, és la següent:

* El valor de contaminació de la zona on es troba una coordenada qualsevol en format API.



Sobre la negociació amb el grup de l'Ajuntament els vam indicar clarament que volíem donar-los una funcionalitat i ells van acceptar ràpidament. La primera proposta que ens van sol·licitar era que els donéssim la funcionalitat de dibuixar el trajecte entre dos punts d'un mapa, cosa que vam declinar perquè no és una funcionalitat de resposta d'una API.

Llavors la segona proposta que ens van enviar és que els retornéssim els punts de càrrega elèctrics i de lloguer de bicicletes propers en una zona, la qual cosa hem acceptat.

En conclusió, la funcionalitat que servim al grup de l'ajuntament és la següent:

* Donada una zona o coordenada, retornar el conjunt de punts de càrrega o llocs de lloguer de bicicletes propers de la zona.



# 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>