Mobilitat sostenible: Electrike

Sprint 2

GitHub - *Taiga* - *Project Record* - *Trello*

Cognoms	Nom	Roles	Correu electrònic		
			Google Drive	Taiga	GitHub
Asenjo Carvajal	Víctor	Public relations	victor.asenjo@estudiantat.upc.edu	victorasenj@gmail.com	
Balaer Morales	Eloi	Dev. team Member	eloi.balaer@estudiantat.upc.edu	eloix2@gmail.com	
Ni	Peilin	Dev. team Member	peilin.ni@estudiantat.upc.edu		
De La Varga Antoja	Ferran	Dev. team Member	ferran.de.la.varga@estudiantat.upc.edu		ferrandelavargaantoja@gmail.com
Dumitru Maroz	Alexandru	GitHub Master	alexandru.dumitru@estudiantat.upc.edu	alexandru666@outlook.es	
Coll Ribas	Xavier	Scrum Master	xavier.coll.ribas@estudiantat.upc.edu		xaviercollr@gmail.com
Rodriguez Rubio	Alvaro	Dev. team Member	alvaro.rodriguez.rubio@estudiantat.upc.edu		

Grau en Enginyeria Informàtica

Projecte d'enginyeria del software - Grup 21

Curs 2021-22 Quadrimestre de Primavera

Pàgina en blanc de cortesia

Índex

1. Introducció

1.1. Què s'ha fet?

Aquesta serà la segona iteració en la qual l'equip desenvolupa la part de l'aplicació més dirigida a l'usuari i el nou sistema de rutes amb punts de càrrega.

Primer hem escollit les tasques a fer per aquest sprint, les hem distribuit entre els integrants de l'equip per tal d'equilibrar el Project Record Track. Hem canviat una mica aquest per tal de facilitar el tracking de les hores de cada persona i per cada tasca, ja que al primer ho teníem separat i feia més complicat el seguiment en ambdós casos.

A continuació, hem començat a desenvolupar les tasques escollides. Per a aquesta entrega prevista ens hem compromès a fer una sèrie d'històries d'usuari i tasques, llistat que exposarem en l'apartat de Cerimònia Àgil i de les quals hem aconseguit enllestir les següents:

- Afegir un vehicle
- Eliminar un vehicle
- Modificar dades de vehicle
- Introduir bateria inicial
- Escollir vehicle de viatge
- Multi idiomes
- Bicings Propers (per la API que necessiten els companys)
- Chargers Propers (per la API que necessiten els companys)
- Visualitzar punts favorits al menú
- Visualitzar només punts favorits al mapa
- Rutes amb punts de càrrega
- Recerca de direccions
- Ocupació

En canvi aquestes, tot i haver-les definit com a objectiu per aquest sprint, no hem pogut finalitzar-les per falta de temps i dificultats al implementar-les:

- Activar notificacions
- Desactivar notificacions

1.2. Sprint master report

Scrum Master: Xavier Coll Ribas

Ens trobem al segon sprint del nostre projecte, on ja tenim una aplicació funcional i útil. Del primer sprint de tots, vam deixar la nostra aplicació ben encaminada, cosa que ens ha facilitat molt a tot l'equip de cara el treball d'aquest. La distribució de l'equip ha estat més o menys la mateixa, 2 persones al front-end, 3 a domini i 1 al back-end. L'Alexandru, entraria dins de tots els equips, ja que és un company polivalent i ha estat tocant una mica de tot.

Només començar, vam fer el sprint planning. Entre tots, vam decidir les històries d'usuari que fariem per aquest sprint i sobretot, com volíem que quedés l'aplicació al final d'aquest. Amb un consens total, vam decidir que l'aplicació la deixaríem enllestida al 80%. Entrem en més detall a l'apartat de *Sprint Planning*.

Per tal de poder fer un bon treball, vam intentar millorar tot allò que creiem que va fallar al darrer sprint. Una de les primeres coses que vam fer, va ser la repartició de tasques. Aquestes es van repartir segons l'àmbit (front-end, back-end...) i segons la càrrega de treball que va haver-hi en l'anterior sprint, és a dir, si una persona va treballar més hores, se li han assignat tasques menys costoses i en menor quantitat . A més, hem canviat el Project Record Track, per tal de poder fer un millor seguiment de les tasques, el temps estimat i el temps real. Aquests canvis han estat molt positius, ja que realment creiem que hi ha hagut una millor organització i no ens hem "trepitjat" feina com va passar anteriorment.

Pel que fa a les sessions presencials, abans de cada classe, posàvem en comú el Taiga. Miràvem com anaven les històries d'usuari, les tasques i en què estàvem treballant en aquell moment. Això feia que sabéssim en tot moment com anàvem, si ens faltava elaborar més o ens podem relaxar una mica. També, aprofitàvem aquella hora per fer les coses més "col·laboratives", com ara fer els merges, acabar de definir algunes tasques...

En relació amb el treball individual, cadascú tenia assignades les seves tasques amb les hores estimades. D'aquesta manera, hem reduït bastant la interacció entre nosaltres, cosa que ens ha permès avançar sense haver d'esperar coses dels altres.

Per comunicar-nos entre l'equip, continuem utilitzant els mateixos mitjans, que indicarem al punt 4.4.Comunicació dins de l'equip.

Per a acabar, aquest segons sprint, encara que no hem pogut acabar totes les tasques predefinides al principi, sí que hem completat algunes que eren molt necessàries. Tot i això, les històries d'usuari no acabades, els hi manquen un parell de tasques, per tant, el treball ha estat molt satisfactori.

1.3. Descripció individual de treball

Alexandru Dumitru Maroz

Al llarg d'aquest sprint tan sols vaig tenir tres tasques però que van prendre bastant temps ja que vaig haver d'aprendre com funcionaven dues llibreries i després implementar funcions pròpies. Aquestes dues tasques anaven relacionades amb el mapa, on vaig haver d'implementar que poguéssim afegir més de dues coordenades al generador de rutes, retornar informació sobre la mateixa i agrupar els markers de coordenades perquè es vegi de manera més estètica.

D'altra banda, també vaig anar ajudant als altres resolent els seus dubtes o problemes.

A més d'això, també vaig anar resolent uns certs bugs que teníem arracades com per exemple arreglar que un giny no funcionava en web. El que va portar a crear un servidor proxy propi per a solucionar-lo.

Álvaro Rodríguez Rubio

Durant aquest sprint he finalitzat les funcions que faltaven respecte als favorits dels bicings i he canviat les funcions principals del controlador de domini ja que carregavem tota la informació a l'inici en comptes d'anar carregant informació a mesura que es demanava.

He col·laborat a fer la part de multidiomes per l'usuari per guardar la selecció del usuari i al canviar els textos de l'aplicació juntament amb el Víctor, la Peilin i el Xavi.

També he fet que l'aplicació guardi un vehicle nou d'un usuari i que aquest, l'editi i l'elimini, mantenint actualitzat sempre el domini i la base de dades gràcies a les funcions de l'api del Xavi i a la part de la interfície feta pel Victor .

Eloi Balaer Morales

En aquest sprint m'he dedicat al desenvolupament de la capa de presentació i la part visual de l'aplicació. Al començament de l'sprint vaig començar afegint un canvi de vista de favorits a les seves pròpies coordenades al mapa, més endavant vaig afegir una vista per a la visualització de les gràfiques d'ocupació dels punts de càrrega favorits. A la vegada vaig arreglar diversos bugs que havien sorgit, com per exemple el del delay dels botons que havíem de polsar 2 cops per a què es realitzés una acció.

A la meitat de l'sprint, vaig afegir una barra de cerca de direccions a l'aplicació, per a poder fer rutes d'un punt a un altre, i vaig modificar el comportament del codi per a abastir aquest canvi.

Al final de l'sprint, he hagut de fer una vista més, on apareixen 3 funcionalitats noves, escollir vehicle on he utilitzat una llista horitzontal per mostrar els vehicles de l'usuari, escollir el tipus de ruta que farà servir l'usuari, que ha estat connectada amb l'algorisme de les rutes amb carregadors fet per Pei Lin per mostrar el nou tipus de ruta, i finalment escollir el percentatge de bateria del cotxe de l'usuari, que en principi no era una tasca meva, però m'ha estat assignada al final de l'sprint per deixar per acabada la vista.

En general estic content amb aquest sprint, ja que hem treballat bé i han sortit la gran majoria de tasques sense massa complicació, a més ens hem coordinat molt bé en moments de treball en grup, com podria ser en fer merge i ajuntar funcionalitats amb la gent de domini.

Ferran de la Varga Antoja

En aquesta Sprint 2 m'he dedicat a certes tasques relacionades amb la capa de domini. Primer de tot em vaig encarregar de petites tasques com la d'implementar que l'usuari pugui tenir un cotxe seleccionat de molts que en pot tenir. Amb aquest cotxe seleccionat es pot obtenir informació o fer rutes amb aquest vehicle i en qualsevol moment es pot canviar.

Després, em vaig encarregar de la implementació de les notificacions de l'aplicació. Com que no tenia res de coneixement del seu desenvolupament, al principi hi va haver un procés de recerca de com fer-ho, triant les llibreries i cercant diferents mètodes. Juntament amb la Peilin vam configurar el sistema de notificacions en el Service Locator.

He implementat que les notificacions surtin instantàniament i notificacions programades (que surtin en una hora concreta). També que l'usuari pugui rebre notificacions que es repeteixin cada setmana; aquestes informen sobre l'estat d'ocupació dels carregadors d'un punt de càrrega.

Em vaig informar de quina llibreria havia de fer servir per tal que les notificacions sortissin en segon pla (background) (quan l'aplicació no estigui en funcionament), cosa que he acabat trobant que de vegades, encara que es faci servir la llibreria depèn de la marca del mòbil.

Voldria destacar que em va costar bastant la implementació de les notificacions i hi vaig dedicar moltes més hores de les que estaven previstes. Degut això, hi ha petits detalls que no s'han pogut acabar i és per això que certes tasques no s'han pogut tancar del tot. Encara que la classe de notificacions que he construït tingui vàries funcionalitats, per aquest Sprint, des de la percepció de l'usuari, només s'ha fet una vista de mostra amb un botó per a cada punt favorit, que quan el cliques, apareix una notificació instantània informant el seu estat.

Peilin Ni

En aquest Sprint 2 he tingut la tasca principal de completar la US de rutes amb punts de Càrrega. Abans de començar aquesta tasca, tenia altres assignades amb menys pes que consisteixen simplement en petits canvis en el Domini per guardar les dades que requereix el sistema. A més, també m'he dedicat a fer les traduccions per al multiidiomes i fer algunes configuracions per als camps que es veuran afectats en el canvi d'idioma, juntament amb el Victor Asenjo.

Un cop acabada aquests, com les rutes amb càrrega depenien d'una tasca que estava finalitzant l'Alexandru, vaig ajudar a connectar amb el servei de notificacions en el service locator d'en Ferran.

Seguidament, m'he dedicat a desenvolupar la US amb un algorisme que busca la millor ruta passant per un carregador quan el cotxe arribi al 10%, que és quan considerem que és el límit i s'ha de carregar. A més d'assignar la ruta més eficient a l'Eloi per a que es pugui mostrar la ruta en el mapa, amb els waypoints.

Víctor Asenjo Carvajal

En aquest segon Sprint m'he encarregat de solventar determinats bugs i errors de càrrega. Hem completat la vista dels favorits i com l'usuari els pot afegir: des del mapa, eliminar-los, consultar aquests... Així com diversos filtres al mapa per facilitar la visualització del punts d'interés o rellevants per a l'usuari. Hem acabat la gestió dels cotxes d'un usuari: ara és completament funcional des de l'addició d'un nou cotxe, fins l'edició d'aquest i el seu esborrat. Tot complint amb els requisits de qualitat i usabilitat establerts: intentar que l'usuari hagi d'introduir o buscar manualment el mínim volum d'informació.

Hem implementat els mecanismes necessaris per fer la plataforma multi-idioma.

D'altra banda, he automatitzat la assignació de rols al Servidor de Discord interdepartamental.

Xavier Coll Ribas

En aquest segons Sprint, juntament amb ser el Scrum Master, he seguit estant al capdavant del Backend de la nostra aplicació.

Pel que fa al rol de backend, he seguit desenvolupant la nostra API, agregant noves funcionalitat a mesura que l'equip de domini i de presentació en necessitaven. També he hagut de fer alguns ajuntaments a les taules d'usuaris per tal de guardar nova informació que abans no havíem tingut en compte.

A més a més, al que he dedicat més temps és al tema de la ocupació dels punts de recàrrega. He fet un script de python on aquest recopila la informació a temps real dels diversos punts de recàrrega per tal de poder veure la seva ocupació i poder mostrar-la a l'usuari. Un cop fet el script, amb ajuda de l'Alexandru, vam poder fer que les dades es guardessin a la BD i vaig acabar de fer l'API per tal de que es poguessin consultar aquestes dades.

Pel que fa al rol de Scrum Master, des del primer moment vaig prioritzar que tinguéssim una bona organització entre tots, per això, només començar amb el sprint planning, vam definir les tasques i assignar-les a cadascú. També, durant les sessions presencials, ens centravem 10 minuts a mirar com la feina i què faltava.

1.4. Avaluació de l'equip

La puntuació s'ha obtingut fent la mitjana de la nota que cada membre ha donat en un formulari de forma anònima.

	Alexandru Dumitru	Álvaro Rodríguez	Eloi Balaer	Ferran de la Varga	Peilin Ni	Xavier Coll	Víctor Asenjo
Puntuació Sprint 1 (0-5)	5	5	5	5	5	5	5
Puntuació Sprint 2 (0-5)	5	5	5	3,8	5	5	5

2. Requisits

2.1. “Not” List - Actualitzada

Només ha hagut un únic canvi: els gràfics de concurrència també es mostraran tant en els punt de càrrega com en les estacions Bicing, punt que inicialment vam considerar no necessària per al segon cas.

SÍ ÉS	NO ÉS
Tenir perfils d'usuari	Xarxa social que permet sistema de followers
Enregistrar els vehicles elèctrics que l'usuari dóna d'alta	Registrar vehicles no elèctrics per part de l'usuari
Introduir les característiques del cotxe o moto elèctric d'un usuari	No té xat de cap mena
Geolocalització	Verificació del perfil d'usuari mitjançant dades privades
Generar rutes en el mapa pels usuaris entre la localització d'origen de l'usuari i el destí desitjat	Valorar perfils
Generar rutes en el mapa pels usuaris amb punts de recàrrega entre la localització d'origen de l'usuari i el destí desitjat	No efectua transaccions econòmiques entre l'usuari i punts de càrrega
Generar rutes ecològiques en el mapa pels usuaris amb punts de recàrrega entre la localització d'origen de l'usuari i el destí desitjat	Valorar punts del mapa
Comparativa entre el consum del cotxe o moto de l'usuari i un model semblant que utilitzi combustibles fòssils	No es permeten reserves de bicicletes a les estacions Bicing
Sistema de gamificació que dona premis als usuaris segons *n* emissions estalviades, quilòmetres recorreguts, etc.	
Visualització de punts de recàrrega ocupats en el moment	
Mostrar els tipus d'endoll per carregar el vehicle en el punt de càrrega seleccionat	

Calcular el temps de recàrrega del cotxe, arribat ja al punt de càrrega indicat	
Fitxa d'informació de cada punt de càrrega: potència, endoll, preu, etc.	
Fitxa d'informació dels punts d'estació Bicing	
Mostrar en el mapa els punts amb accés a bicicletes Bicing	
Oferta de diferents tipus de rutes com l'estàndard, amb càrrega, l'ecològica i l'ecològica amb càrrega.	
Compartir ubicació una vegada l'usuari arriba a un punt de càrrega mitjançant un enllaç	
Notificacions i avisos d'un punt de càrrega als usuaris que tinguin marcats aquell punt	
Mostra informació concreta d'establiments o llocs concrets	
Mostrar gràfics de concurrència de les estacions de càrrega al mapa	
POT SER	
Instal·lació de punts de recàrrega propis pels usuaris de la nostra aplicació	
Opció de reserva de punts de recàrrega limitat per una certa categoria d'usuaris.	
No mostra gràfics de concurrència a les estacions bicing	
Ampliació d'una nova categoria d'usuaris Premium per tenir funcionalitats extres a l'aplicació	
Aplicació col·laborativa on els usuaris poden afegir nous punts de càrrega i les seves característiques no existents en l'aplicació	
Recomanador de rutes genèric per diferents tipus de transport sostenible: bus, bicicletes, patinets elèctrics, etc	
Reportar estacions de càrrega del mapa: anomalies de funcionament	
Mostrar 3 estacions més properes a la ubicació de l'usuari en el cas de quedar-se sense connexió	
Filtrar els estacions de recàrrega per tipus de empresa	

2.2. Product Backlog

A la taula següent hi trobem les històries d'usuari de Electrike.

Tal com indica, la columna "Estat" indica l'estat actual de la US: en el cas que ja estigui finalitzada, es marca l'Sprint en el que s'ha tancat; en el cas d'haver-se començat , però, no acabat, el trobem *In progress*; per últim, si la US encara no s'ha obert per aquest Sprint, el marquem com a N/A.

[*Nota:*

- *Gran part de les US assignades a l'Sprint 1 han estat finalitzades. Aquelles que s'han començat són, en gran part, per altres Sprints.*
- Les caselles marcades en verd són US que hem afegit; groc, les que hem modificat el nom.]

Èpics	Històries d'usuari	Estat
Gestió Usuari	Login	Sprint 1
	Log out	Sprint 1
	Sign up	Sprint 1
	Eliminar perfil	Sprint 1
	Imatge perfil	Sprint 1
	Afegir punt a preferits	Sprint 2
	Eliminar punt de preferits	Sprint 1
	Activar Notificacions	In progress
	Desactivar Notificacions	In progress
	Multi idiomes	Sprint 2
	Compartir ubicació	N/A
Gestió Vehicles	Afegir un vehicle	Sprint 2
	Eliminar un vehicle	Sprint 2
	Modificar dades vehicle	Sprint 2
	Introduir bateria inicial	Sprint 2
	Escollir vehicle de viatge	Sprint 2

Gamificació	Consultar trofeu	N/A
	Guanyar trofeu	N/A
Mapes i rutes	Ruta més ràpida	Sprint 2
	Ruta amb menys contaminació	N/A
	Ruta amb punts de recàrrega	Sprint 2
	Bicing propers	Sprint 2
	Recerca de direccions	Sprint 2
	Vista alternada	Sprint 1
	Consultar Estació de Bicing	Sprint 1
	Consultar punt de càrrega	Sprint 1
	Gràfiques d'ocupació dels punts	Sprint 2
	Visualitzar punts favorits	Sprint 2
	Visualitzar només punts favorits en el mapa	Sprint 2
Requisits no funcionals	Multiplataforma	Sprint 1
	Mostrar ràpidament les rutes	In progress
	Ràpid aprenentatge	In progress
	Aplicació fluida	In progress
	Facilitat d'ús	In progress
	Navegació per l'aplicació	Sprint 1

2.3. Requisits no funcionals - Plantilla Volere

10a. Requisits d'Aparença

- El producte ha de ser atractiu per a tota mena de públic que tingui l'edat per poder conduir un vehicle.
- El producte ha de complir amb uns estàndards decidits pels participants del grup, per tal que l'estètica sigui simple i minimalista alhora que uniforme a tot el sistema.

Criteri de satisfacció

Una mostra representativa del públic major de divuit anys amb un vehicle elèctric, sense incentius ni entrenament o formació prèvia, ha de començar a utilitzar les funcionalitats principals del producte en 5 minuts després de la seva primera trobada amb aquest.

L'equip de desenvolupament ha de certificar que compleix els estàndards i patrons estètics acordats per tal que es vegi com a una marca unificada.

11a. Requisits de Facilitat d'Utilització

- El producte ajudarà al fet que l'usuari no cometi errors a l'introduir dades.
- El producte serà fàcil d'emprar per a qualsevol adult.
- El producte farà que els usuaris vulguin utilitzar-lo.
- El producte ha de ser emprat sense entrenament i amb un coneixement mínim sobre el vehicle elèctric que fa servir l'usuari.

Criteri de satisfacció

El 75% del panel de prova d'adults podrà navegar pel mapa, visualitzar i afegir a favorits punts de càrrega i Bicing, afegir un vehicle i cercar una ruta amb èxit després d'un dia fent servir l'aplicació.

El 60% d'usuaris farà ús regularment de l'aplicació després d'un mes d'ús i familiarització amb les seves funcionalitats.

11b. Requisits de Personalització i Internacionalització

- L'usuari podrà canviar el llenguatge de l'aplicació en qualsevol de les opcions que oferim.
- Convencions decimals i sistema internacional d'unitats.

Criteris de satisfacció

L'usuari podrà canviar entre els 3 idiomes disponibles en tot moment anant a l'opció, aquest són el castellà, l'anglès i el català. L'idioma seleccionat serà guardat per la propera sessió.

Les unitats de mesura seran marcades a cada camp.

11e. Requisits d'Accessibilitat

- El producte pot ser utilitzat per daltònics
- El producte serà utilitzable per persones amb sordesa.

Criteris de satisfacció

Les persones daltòniques no tenen cap impediment per utilitzar les funcionalitats principals de l'aplicació, ja que cap informació rellevant es mostra amb només diferenciació de colors.

Les persones amb sordesa no tenen cap dificultat al fer servir l'aplicació, perquè cap informació es notifica per sorolls.

12a. Requisits de Velocitat i Latència

- L'aplicació ha d'iniciar-se en menys de 7 segons
- El producte ha d'oferir una ruta en menys de 3 segons
- L'usuari ha de poder canviar de vista i aquesta s'ha de carregar en menys de 2 segons, en el cas del mapa en menys de 5 segons.
- La base de dades s'actualitzarà cada 5 minuts les dades que no són estàtiques.

Criteris de satisfacció

Els criteris anteriors es consideraran satisfets quan la connexió a internet de l'usuari és òptima i el dispositiu suporta amb normalitat la capacitat de l'aplicació.

12c. Requisits de Precisió o Exactitud

- Totes les quantitats s'arrodoneixen a 2 decimals.
- Les quantitats de temps s'expressen en minuts, hores o dies.

Criteris de satisfacció

Les mesures que no siguin de temps han de complir el primer criteri mencionat, ja que la resta de decimals no són importants per l'usuari.

Les unitats de temps l'actitud en segons no és necessària, perquè és massa exacta en una aplicació per crear rutes i aquest segons poden variar bastant per la velocitat de l'usuari, inclòs els minuts poden.

12d. Requisits de Confiabilitat i Disponibilitat

- El producte estarà disponible 24 hores al dia, 365 dies per any.
- El producte treballa el 97 % del temps sense fallades.

Criteris de satisfacció

El producte ha d'estar disponible i en funcionament el temps mencionat i no ha de fallar tret que la connexió de l'usuari no sigui bona.

12e. Requisits de Fortalesa o Tolerància a Fallades

- El producte ha de guardar l'última ruta consultada per l'usuari si es queda sense connexió
- El producte mostrarà les dades de l'usuari, els seus vehicles i marcarà els punts de càrrega i Bicing si l'usuari ha consultat aquestes dades mentre està enregistrat.

Criteris de satisfacció

El producte ha de poder mostrar la informació mencionada anteriorment sense connexió una vegada s'ha iniciat un comte i s'ha consultat aquesta informació mínim un cop quan aquest tenia connexió a internet.

13b. Requisits d'Interfície amb Sistemes Adjacents

- L'aplicació es podrà utilitzar en els navegadors webs més populars (Chrome, Edge...)

Criteris de satisfacció

L'aplicació ha de poder ser compilada i executada per a web. A més de ser testejada per l'equip i que les funcionalitats principals funcionin en els navegadors més populars.

14c. Requisits d'Adaptabilitat

- Android Jelly Bean, v16, 4.1.x o posterior, i iOS 8 o més nous.
- Dispositius iOS (iPhone 4S o posterior) i dispositius Android ARM.
- Window, MAC i Linux

Criteris de satisfacció

L'aplicació ha de poder ser compilada i executada per a web, Android i IOS a través de l'opció que dona el *framework* Flutter. A més de poder descarregar-la i utilitzar-la correctament indistintament del tipus de dispositiu.

15a. Requisits d'Accés

- L'usuari haurà d'iniciar sessió per tal de disposar de totes les funcionalitats importants del producte.

Criteris de satisfacció

Els usuaris enregistrats disposaran de les funcionalitats al complet del producte, en canvi, un usuari que no ha iniciat sessió només podrà visualitzar el mapa, els punts de càrrega i Bicing.

15c. Requisits de Privacitat

- El producte requerirà el consentiment per part de l'usuari per usar la seva ubicació.
- Es notificarà a l'usuari per tal que sàpiga quines dades seran emmagatzemades en la base de dades
- Es notificarà a l'usuari per tal que sigui conscient de quines dades seran utilitzades d'altres aplicacions.

Criteris de satisfacció

Els usuaris hauran d'acceptar aquest requisit per fer servir per complet l'aplicació i ha de ser conscient de l'ús que li donem i el que comporta.

Mantenim encriptades aquestes dades dintre de la base de dades del producte, a les quals només podrà accedir ell mateix i l'equip de desenvolupament, per tal de mantenir la privacitat de l'usuari.

2.4. Gestió Taiga

La informació relacionada al progrés de les tasques d'aquest Sprint i les US es poden consultar al nostre taulell de Taiga.

Dins de cada història d'usuari, hi ha el conjunt de tasques necessàries per tal de poder-la completar. Aquestes tasques estan ben indicades, és a dir, han seguit el procés de "Not done" → "In progress" → "Test" → "Closed". A més, està apuntat el temps previst per cada una i el temps real que hem tardat, que prèviament havíem apuntat en el PRT, per tal de fer un millor seguiment.

A més, a petició de Silverio, hem definit també els respectius "Criteris d'acceptació" de cada US d'aquest Sprint. Cal remarcar que aquests criteris poden anar actualitzant-se en futurs Sprints en el cas de que la US no s'hagi finalitzat. També pot ser que la US encara no assoleix tots aquests criteris, pel mateix motiu de que encara no està tancada.

Un punt a ressaltar és la gestió de les NFRs: no tots aquests requisits han passat a ser US, n'hi ha que formen part d'altres històries d'usuari i, per tant, es distingueixen com a tasques.

Aquelles que han passat a ser US directament i aquesta entrega són les següents:

- Requisits d'Aparença i facilitat d'utilització → Navegació aplicació: les funcionalitats de l'aplicació es troben fàcilment al navegar per aquesta. Les opcions ofertades es distingeixen molt bé en els apartats del menú lateral.
- Requisits d'Adaptabilitat → Multiplataforma: aplicació desenvolupada tant per web com aplicació mòbil.

Altres que han estat assignades dins d'US:

- Requisits d'accessibilitat → Consultar punt de càrrega: els punts de càrrega mostren la informació amb icones diferenciadores que ajuden a l'enteniment dels estats d'aquests punts, és a dir, no es distingeixen únicament per color.
- Requisits d'accés → *log in* i *sign up*: aquestes funcionalitats es duen a terme amb els respectius comptes de Google de l'usuari.

2.5. Funcionalitats transversals

Aspecte transversal	Descripció	Estat
Geolocalització	L'aplicació consisteix en un mapa on es mostra diferents marcadors i la localització de l'usuari, a més dels marcadors, també proporciona a l'usuari diferents camins per arribar a localitzacions (rutes)	Acabat
Xarxes socials	L'usuari es dona d'alta al sistema mitjançant les seves credencials de Google. La seva sessió dins l'aplicació és manté oberta.	Acabat
	Pot compartir una ubicació generant una URL.	N/A
Gamificació	L'aplicació assigna premis als usuaris segons els requisits predefinits dins del sistema.	N/A
Refutació	L'usuari rep notificacions d'aquells punts que ell ha activat. Aquests avisos informen de l'ocupació dels punts.	En progrés
Calendari	L'usuari pot activar notificacions personalitzades, defineix un dia i una hora. Aquesta segueix l'horari definit al calendari estàndard de la zona.	En progrés
Multiidioma	Proporciona a l'usuari diferents idiomes per als textos de l'aplicació, en concret: català, castellà i anglès	Acabat
Multiplataforma*	L'aplicació és compatible amb Android i web.	Acabat (versió 2)

Gràfics d'ocupació	Es podran veure gràfiques de les ocupacions dels punts Bicing i dels carregadors de Barcelona.	Acabat

- **Geolocalització**

Aquest aspecte s'ha desenvolupat amb la API de Google Maps, el qual ens ofereix tota la informació que he necessitat fins ara.

La nostra integració del concepte transversal ha estat la implementació d'un mapa com a finestra principal, en la que localitzem a l'usuari, diferents punts de càrrega i Bicing, que aconseguim de la base de dades i de les APIs obertes. També hem afegit rutes per trobar el mètode més ràpid per arribar a un punt, a més de la ruta alternativa en el cas que el vehicle elèctric necessiti un punt de càrrega intermig. Per obtenir les localitzacions al mapa de manera senzilla, hem implementat una barra de cerca de direccions, que serveix per afegir una ruta al mapa.

- **Xarxes socials**

Per poder guardar la informació relacionada a un usuari dins del nostre sistema, hem fet servir la llibreria i l'API de Google Sign In, i hem utilitzat la web de Google console, on s'ha configurat aquesta gestió d'usuaris. Inicialment, tenim una pàgina per fer el login, però ara ha canviat a un apartat del menú, necessari per utilitzar diferents funcionalitats ja esmentades. Aquest login demana a l'usuari el correu electrònic de Google i fins ara només podem registrar-nos amb els nostres comptes, però fer-lo de manera pública serà canviar un valor.

A més, per al pròxim i últim Sprint, tenim plantejat que l'usuari pugui compartir la seva localització, o qualsevol altre, mitjançant una URL que es generaria quan ell decideixi fer ús d'aquesta funcionalitat.

- **Refutació**

L'usuari pot rebre notificacions sobre un punt de càrrega de la seva llista de favorits quan així ho desitgi. Tot i no estar la US acabada per l'Sprint 2, què és per quan la volíem tenir, tenim una primera versió d'aquesta en la que l'usuari pot rebre una notificació polsant un botó del punt. Per poder desenvolupar aquesta funcionalitat hem utilitzar dues llibreries de flutter que és *local_notifications* i *provider*, que es guarda les notificacions i s'encarrega d'enviar-les un cop arriba el temps de fer-ho.

- **Calendari**

Aquest aspecte està directament relacionat a l'anterior, és a dir, un usuari pot definir una notificació donat un dia concret o un dia de la setmana, una hora, etc. Aquestes característiques segueixen el fus horari de Catalunya, que és el territori sobre on està desenvolupat l'aplicació.

- **Multi idioma**

Per poder oferir aquest aspecte, nosaltres hem considerat les 3 llengües més parlades dins del territori en què està pensada l'aplicació: català, castellà i anglès. Fent ús d'un plugin que ens ofereix el mateix entorn, nosaltres només hem hagut de fer les traduccions dels camps que canviarien d'idioma i fer crida de la funció pertinent dins del codi de la UI per carregar el text que pertoca.

- **Multiplataforma**

Gràcies al framework amb el que treballem, la implementació de la pàgina web va enllaçada al desenvolupament de l'aplicació mòbil. D'aquesta forma, no necessitem implementar un nou codi per fer el desplegament en web. Al compilar el codi que tenim actualment, podem escollir la opció de desplegar-lo en format web, cosa que mostrara la versió desenvolupada, fins al moment.

- **Gràfics d'ocupació**

Donat un punt de càrrega de la nostra base de dades, l'usuari té la opció d'obtenir la concurrència d'aquests en format de gràfic lineal. Aquesta informació pot decidir obtenirla tant visitant la informació detallada d'un punt de càrrega com configurant una notificació per avisar-li de la ocupació del punt.

Per desenvolupar aquesta funcionalitat, han hagut de cooperar les 3 capes de l'arquitectura per tal d'obtenir les dades necessàries per dibuixar el gràfic que es mostra a l'usuari.

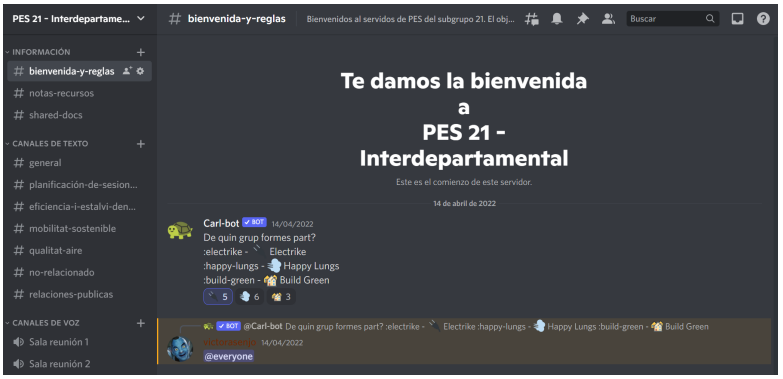
2.6. Serveis de tercers

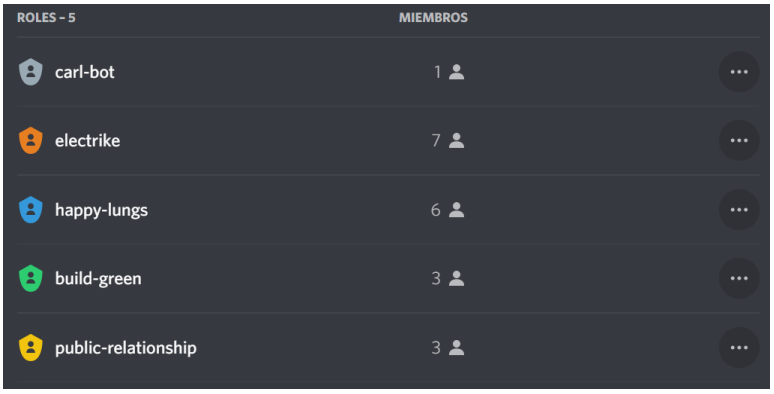
Comunicació entre equips

Tenim 1 via principal de comunicació i una en reserva per si falla la primera.

La principal és Discord per a reunions, consultes i discussions que engloben tot tipus de detall. S'ha eliminat el grup de WhatsApp perquè no s'utilitzava i s'ha incorporat a Discord un nou canal de comunicació de text que substitueix el de WhatsApp. A Discord estem tots els integrants del grup 21 de PES 2021-22 Primavera. S'han assignat rols a cada integrant del grup de Discord per diferenciar els equips de desenvolupament. Per dur la gestió d'assignació de rols s'ha emprat el bot Carl-bot: amb aquest bot es crea un missatge al canal que es vulgui ("Benvinguda i regles" en el nostre cas) i els usuaris de la sala poden interaccionar amb les reaccions; les quals tenen un rol vinculat que assigna automàticament. D'aquesta manera també facilitem la menció als diferents grups per comunicats entre equips.

Com a mètode de comunicació en cas de fallada de serveis de Discord és Gmail: el correu oficial de la Universitat.



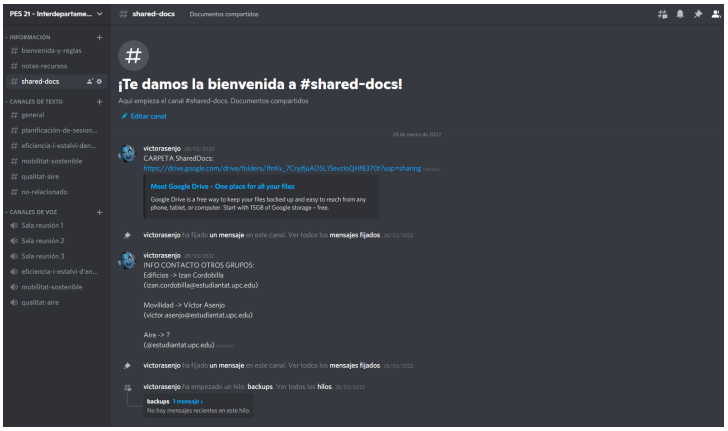


Funcionalitats pactades

La repartició de serveis acordada seria la següent:

Grup	Oferim a	Rebem de
Qualitat de l'Aire	*Electrike*	*BuildGreen*
Happy Lungs	els punts de contaminació i els seus nivells segons una determinada zona. Falta determinar en quin format.	geolocalització d'aquells edificis que pertanyen a un determinat rang d'eficiència
Cases Sostenibles	*Happy Lungs*	*Electrike*
BuildGreen	geolocalització d'aquells edificis que pertanyen a un determinat rang d'eficiència	els punts de càrrega (i si volem, parades de bici) més propers donada una ubicació concreta (coordenades).
Mobilitat sostenible	*BuildGreen*	*Happy Lungs*
Electrike	els punts de càrrega (i si volen, parades de bici) més propers donada una ubicació concreta (coordenades).	els punts de contaminació i els seus nivells segons una determinada zona. Falta determinar en quin format.

Per treballar conjuntament sobre un mateix fitxer fem servir Drive. El link a la carpeta d'aquest es troba com a missatge fixat al xat "Shared-Docs" de Discord i hi ha un fil, en aquest mateix canal, anomenat "backups" on es van pujant còpies de seguretat d'aquests documents periòdicament.



3. Cerimònia Agile

3.1. Report on the sprint planning, review and retrospective meetings

Sprint Planning

Aprofitant els costos de les històries d'usuaris definits anteriorment, vam decidir durant el sprint planning les històries d'usuari que volem treballar per aquest segon sprint per tal de igualar la càrrega del primer, a més d'evitar tenir molta càrrega de treball al tercer per poder dedicarnos a arreglar bugs, millorar les NFR i ser previsors amb les entregues i els exàmens finals d'altres assignatures sprint de final de quadrimestre. Com a principals vam escollir les referents a l'ús de l'usuari com els seus vehicles, els idiomes... També les que eren per les aplicacions dels altres companys de la asignatura:

L'objectiu principal d'aquest segon sprint és tenir acabada la nostra aplicació al 80% com a mínim, on l'usuari ja podrà fer un ús casi complert d'aquesta. Tot i que el nostre objectiu pot ser ambiciós, hem cregut que podem fer-ho veient el desenvolupament obtingut del sprint 1. D'aquesta manera, de cara a l'últim sprint només ens quedarà fer retocs i completar els bugs si n'hi ha. Per tal de deixar l'app al 80%, hem decidit escollit les seguents històries d'usuari.

Història d'Usuari	Puntuació
Afegir un vehicle	8
Eliminar un vehicle	5
Modificar dades de vehicle	3
Introduir bateria inicial	8
Escollir vehicle de viatge	2
Multi Idiomes	3
Ocupació	13
Activar notificacions	13
Desactivar notificacions	8
Rutes amb punts de càrrega	13
Bicing Propers (API)	5
Chargers Propers (API)	5
Recerca de direccions	3
Visualitzar punts favorits al menú	5
Visualitzar només punts favorits al mapa	3

Primer vam canviar de scrum master com es normal a l'inici d'aquest sprint en el project, sent el Xavi el nou scrum master.

A continuació vam pensar que aquestes fossin les candidates el segon sprint per finalitzar ja que eren les relacionades principalment amb l'ús del usuari i les que creiem que feien més utilitzable la nostra aplicació.

Vem continuar amb la mateixa estructura d'equips que al anterior sprint exceptuan per l'Alexandru que ha deixat de formar part del equip de l'API i la base de dades com a tal i s'ha dedicat a la llibreria de google maps principalment. També ha passat a oferir ajuda a tots els companys, sent una figura més de full-stack.

Sprint Review

Sprint 1 review

Sprint review feta amb Maria José a la sessió de classe següent a la de la entrega.

La gran majoria de US s'ha completat de forma satisfactòria, a excepció de la "Visualització de Mapa".

En aquesta, la tasca de Geolocalitzar l'usuari a través del dispositiu amb el que està consultant l'aplicació per prendre la coordenada d'origen, ha estat completada amb èxit, al contrari del que creiem. La consideràvem més complicada que introduir manualment la direcció d'origen. Per aquest motiu, s'ha deixat per el Sprint 2 la tasca de poder introduir un punt d'origen manualment, a més de prendre per defecte la localització de l'usuari.

Una altra US que no s'ha tancat ha sigut la de "Rutes ràpides" degut a que l'equip ha considerat més òptim deixar la tasca que falta, que és mostrar gràficament la distància i duració del recorregut, per quan tinguem els tres tipus de rutes que acabarem oferint. A part d'aquestes US, no hi ha hagut cap més inconvenient relacionat al progrés de les funcionalitats.

Com a punts a millorar: la Product Owner ha considerat convenient afegir a l'inici de l'aplicació una petita llegenda per facilitar la interpretació de les icones que fem servir per indicar els estats i compatibilitats dels endolls de les estacions de càrrega.

També recomana mostrar les localitzacions, tals com ciutats, províncies, barris, etc, en el mapa per facilitar i situar l'usuari al mapa. A més, coincideix amb l'equip amb la idea de clusteritzar els punts de càrrega i Bicing per tenir un mapa més "net" i millorar la UX.

Els punts forts han estat que un gran percentatge de les US assignades com objectiu en aquesta entrega s'han completat amb satisfacció i bona qualitat de resultats. És més, s'han obert altres US que no estaven previstes per aquest Sprint, cosa que ens fa pensar que podem ampliar l'abast de cara als pròxims lliuraments. A la Mª José li ha agradat la idea de que gravem videos amb petites demostracions entre nosaltres per tal de tenir un back-up en el cas de que falli el servidor, o qualsevol altre element, al dia de la demo oficial.

Sprint Retrospective

Sprint 1 Retrospective

Tenint en compte que aquest és el primer Sprint de desenvolupament dins d'un entorn completament nou per a tots els membres de l'equip, el caos que ens hem trobat a l'inici és bastant comprensible.

Primer de tot, les eines de gestió de projecte no s'han utilitzat de la forma més eficient: el Taiga no s'ha anat actualitzant com cal i les tasques no s'han anat assignant a l'inici, que és quan s'ha de fer per mantenir una càrrega de treball compensada entre tots els membres; en quan a PRT, no hem indicat clarament sobre quina tasca i US anava treballant cadascú i, per tant, a l'hora de calcular les hores totals reals dedicades a aquelles tasques, només hem pogut obtenir valors estimats. De cara al pròxim Sprint, es planteja tenir una bona definició de tasques tant al PRT com al Taiga, d'aquesta forma tothom tindrà el seu propi objectiu al Sprint.

Seguidament, relacionat al punt anterior, podem observar com en el PRT hi han membres que doblen les hores dedicades a uns altres. Per tal de disminuir aquesta desigualtat, a més de millorar les eines de gestió, hem vist convenient disminuir la quantitat de tasques d'aquest membres per la següent iteració i, augmentar les tasques en aquells que tenen menys hores dedicades. D'aquesta forma, de cara a l'últim Sprint, tothom tindrà unes hores similars per repartir les tasques equitativament.

Com a punts positius, hem de mencionar que l'equip té una molt bona comunicació entre sí i hi ha un molt bon ambient de treball. Això ajuda a la hora de demanar canvis o informació entre les diferents capes de l'arquitectura.

Com a últim, la qualitat del codi és força bona ja que tothom segueix un estàndard definit en el nostre entorn de Github i, per tal de poder fer commit d'una tasca, aquesta ha de superar els testos del CD.

Sprint 2 Retrospective

Per començar, analitzarem els punts febles per poder millorar de cara el pròxim Sprint.

Tot i haver fet una segmentació i assignació de tasques més detallada que al Sprint 1, s'ha notat que moltes d'aquestes generen dependències amb altres assignades a diferents membres.

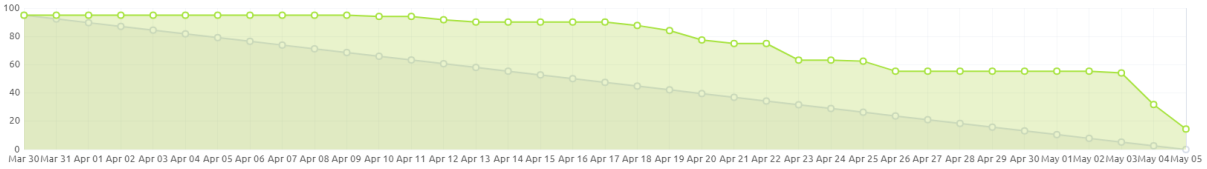
És a dir, algunes tasques depenien d'altres, ja que es necessitava informació d'una altra capa de l'arquitectura i, per tant, no es podia avançar fins que l'anterior no finalitzi. Per aquest problema, l'equip veu convenient dues solucions: assignem les tasques de forma més agrupada de manera que una persona s'encarregui de les tasques que formen una mateixa US o, establir deadlines a la persona encarregada de la tasca de la que es depèn, de forma que evitem que es deixin tasques vitals per a la última setmana del Sprint o s'assignin tasques noves a l'últim moment. Per aquest mateix motiu, hi ha hagut dues US que no s'han pogut tancar per aquesta entrega les quals són totes les de la família de "Notificacions", ja que no vam calcular molt bé el temps i no es va començar amb la suficient antelació.

A part d'això, no hi ha res més dolent a destacar ja que, gràcies a la bona assignació i definició de tasques al principi d'aquest Sprint, ens hem evitat problemes que ens hem trobat en el Sprint 1 tals com: càrrega de treball molt descompensada on algunes persones doblen les hores de treball d'altres, "trepitjar-se" la feina entre membres per la mala assignació de tasques, etc. L'adient organització de l'entorn de treball com el Taiga i el PRT, ha millorat les dinàmiques de treball i, tanmateix, a compensar aquell desbalanç de hores dedicades al Sprint anterior.

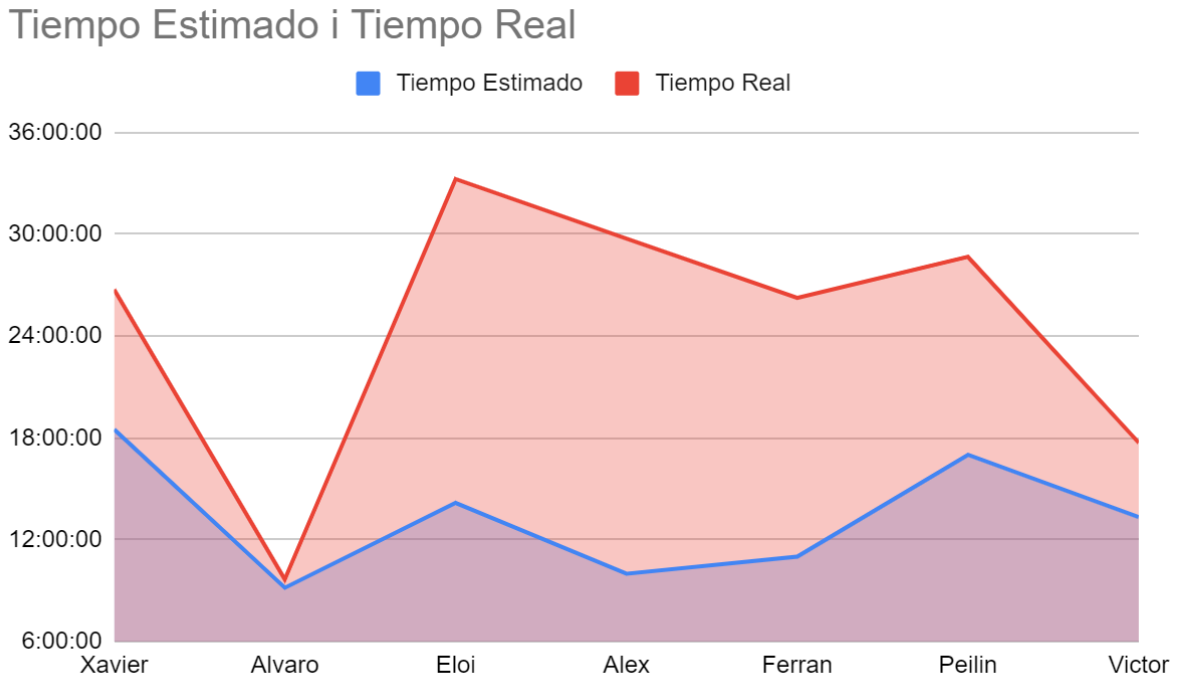
Per últim, en quant a qualitat de codi, al seguir el mateix estàndard establert a l'inici del procés de desenvolupament, no hi ha cap problema en quant a diferència de

format del codi. Cal fer especial èmfasi en que l'equip ha considerar suprimir la necessitat dels tests unitaris ja que no n'hem fet ús d'aquests ni en la creació de rutes amb càrrega, que és considerada com una de les tasques més complexes del sistema.

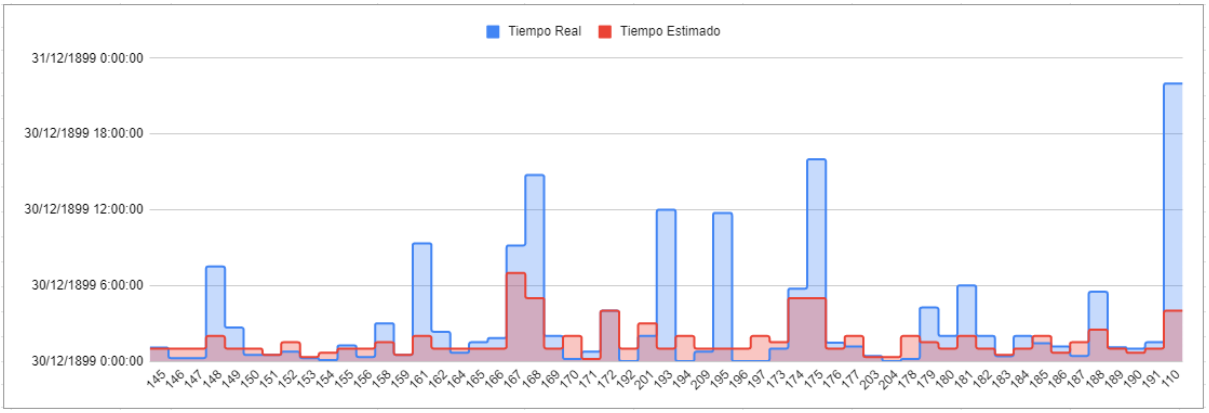
3.2. Updated release and iteration burndown charts and velocity chart



Quant al Burndown chart podem observar com la majoria d'històries d'usuari s'han anat completant cap al final, la qual cosa no és molt ideal per a evitar els bugs d'última hora.



En aquest altre gràfic podem veure com, a excepció d'Álvaro, tots hem tingut imprevist en el desenvolupament de les nostres tasques. Pel que no hem arribat a estimar correctament els temps.



En aquest últim gràfic podem veure el temps individual per cada taverna. Aquí es pot observar com la majoria de tavernes les vam poder aproximar correctament a excepció d'unes certes tasques que la majoria van ser resoldre problemes matemàtics (càlcul de rutes), creació de clústers, recopilació d'ocupació.

Així i tot destacar que aquest tipus de tasques les vam anar fent esporàdicament entre més persones, per la qual cosa el temps real no arriba a ser de tan sols una persona.

4. Metodologia

4.1. Visió global

Nosaltres hem adoptat una metodologia de treball SCRUM. D'aquesta metodologia podem destacar 5 events: la del Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective i Refinement.

Pel que fa al Sprint Planning a les fases d'Inception: primer decidim el Scrum Master d'aquell sprint, mirem quines són les tasques a elaborar i ens distribuïm la feina entre els integrants.

En el Daily Scrum, cada integrant de l'equip, o els subgrups en què s'ha creat per fer les tasques, posen en comú el treball que ha avançat i si necessita ajuda o implicació d'algun altre integrant més. També es posa en comú informació interessant trobada de cara al procés de desenvolupament. A destacar, a cada daily scrum, miràvem com anava el desenvolupament de les històries d'usuari i posàvem al dia el Taiga. Llavors, de decidir en quines tasques ens posàvem a continuació

Per aquest Sprint, si podrem fer una sprint review juntament amb la Mª José. Tot i no estar per aquesta mateixa entrega, la idea és fer una presentació, amb els stakeholders (els professors) dels resultats obtinguts durant aquest Sprint fent una presentació del producte actual. D'aquesta forma veurem si s'han completat els objectius definits en el Sprint Planning.

Per últim, la fase de refinament ens l'hem trobat quan algú de nosaltres descobria alguna informació o aportava quelcom interessant i la compartia amb el grup. Llavors, es feien les modificacions pertinents. Tot i seguir aquesta planificació, és molt habitual el contacte entre els membres de l'equip, ja que molts treballem de manera conjunta i necessitem avançar al mateix ritme.

Gestió de l'equip

Per treballar en els Sprints hem segmentat l'equip en tres sub equips principals segons la capa en la qual s'havia de treballar: capa de presentació, capa de domini i capa de dades.

Segons les preferències de cada membre, s'han assignat 2 persones per capa de Presentació: Víctor Asenjo i Eloi Balaer; 3 persones per Domini: Álvaro Rodríguez, Peilin Ni i Ferran de la Varga; i 1 per la BD: Xavier Coll. L'Alexandru Dumitru, que abans formava part del grup de BD, ara segueix sent el github master com a l'anterior sprint i participa en tasques de tot el projecte (fullStack).

Un cop les tasques assignades a cada membre han anat finalitzant, els equips s'han anat redistribuint segons la demanda d'altres capes o tasques encara pendents. És a dir, si en Domini necessitàven més ajuda, el que havia acabat la seva feina en BD els ajudava.

4.2. Gestió projecte

La gestió del projecte la fem mitjançant diversos softwares.

Per mantenir en ordre les nostres tasques, fem servir el Trello, on apuntem què hem de fer, per a quan ho hem de fer i qui ho ha de fer. Tenim columnes per marcar aquelles tasques a fer, fetes, en procés o en revisió. També tenim definit les hores estimades per cada tasca i el temps real que hem tardat.

Al Taiga tenim les històries d'usuari, amb els seus propis *Criteris d'acceptació* i amb els seus respectius costos (calculats prèviament amb una sessió de planning poker que vam fer online). Allà, tenim el backbone, on ens organitzem per sprints. Dins de cada sprint, tenim diverses històries d'usuari dividides per tasques, on cada cop que treballem, ens assignem una i la intentem realitzar. En cas que l'acabem, la movem a la casella de "Ready for Test" i, un cop integrat, es classifica com Closed.

Per tenir constància del nostre temps de treball, tenim un Google SpreadSheet on anotem les hores que dediquem individualment per cada tasca relacionada amb el projecte. Per finalitzar, les tasques les repartim entre membres del grup, ja que hem vist que no és eficient treballar tots junts en una cosa concreta i, a més, ens és difícil coincidir a tots.

4.3. Gestió repositori

De cara a la gestió de repositoris fem servir principalment GitHub i tenim un encarregat, Alexandru Dumitru, que és el Github Master, qui fa les branches i els merges a petició dels membres del grup. Quan una persona creu que el que ha desenvolupat, té els estàndards establerts, no dóna error i passa els corresponents tests, aquest ho puja a la seva branch. Un cop allà, es demana a l'encarregat del Github que faci merge cap a master o cap a una altra branca que necessita el codi pertinent, per exemple, un company del mateix subgroup.

Respecte al workflow del mateix repositori, hem dividit aquest en dos: en Backend i Frontend, per tal de facilitar la seva implementació i tenir-lo diferenciat. Així doncs, el Backend es troba a un repositori i d'altra banda el Frontend. Pel que fa a l'estructura interna de cada repositori, seguim un model GitFlow on hi haurà una branca Màster on estarà el producte final i múltiples branques secundàries de desenvolupament per cada membre que estigui treballant al repositori. Podem veure tota aquesta estructura als següents enllaços: Backend; Frontend.
[*Notes:*

- *Com podem observar, actualment el nostre repositori ja té gràfics de workflow segons els commits que cada membre ha anat fent. Tot i així, aquesta informació no és precisa: hi ha membres que han treballat en un altre entorn pel desenvolupament de la BD o en la dedicació de la descripció del document.*]

4.4. Comunicació dins l'equip

Pel que fa a la comunicació entre membres del grup, tenim un grup de Whatsapp on decidim horaris de reunions i per on enviem petites demos de l'evolució de l'aplicació amb vídeos, imatges... i parlem sobre assumptes poc rellevants i volàtils.

D'altra banda, tenim un Discord on tenim un petit "achieve" de tecnologies o documents d'interès per al projecte i per a on ens comuniquem de manera més habitual i per on fem les reunions online.

A part de les reunions que es duen a terme durant les dues sessions presencials de classe a la setmana, també s'ha organitzat reunions online quasi diàries per acabar de polir el treball fet o per definir les tasques a fer.

Per a aquest Sprint, ens hem arribat a connectar diàriament per treballar en sub equips i, especialment, en la integració del projecte i resolucions de problemes, ja que la comunicació és més inmediata.

4.5. Gestió de la qualitat

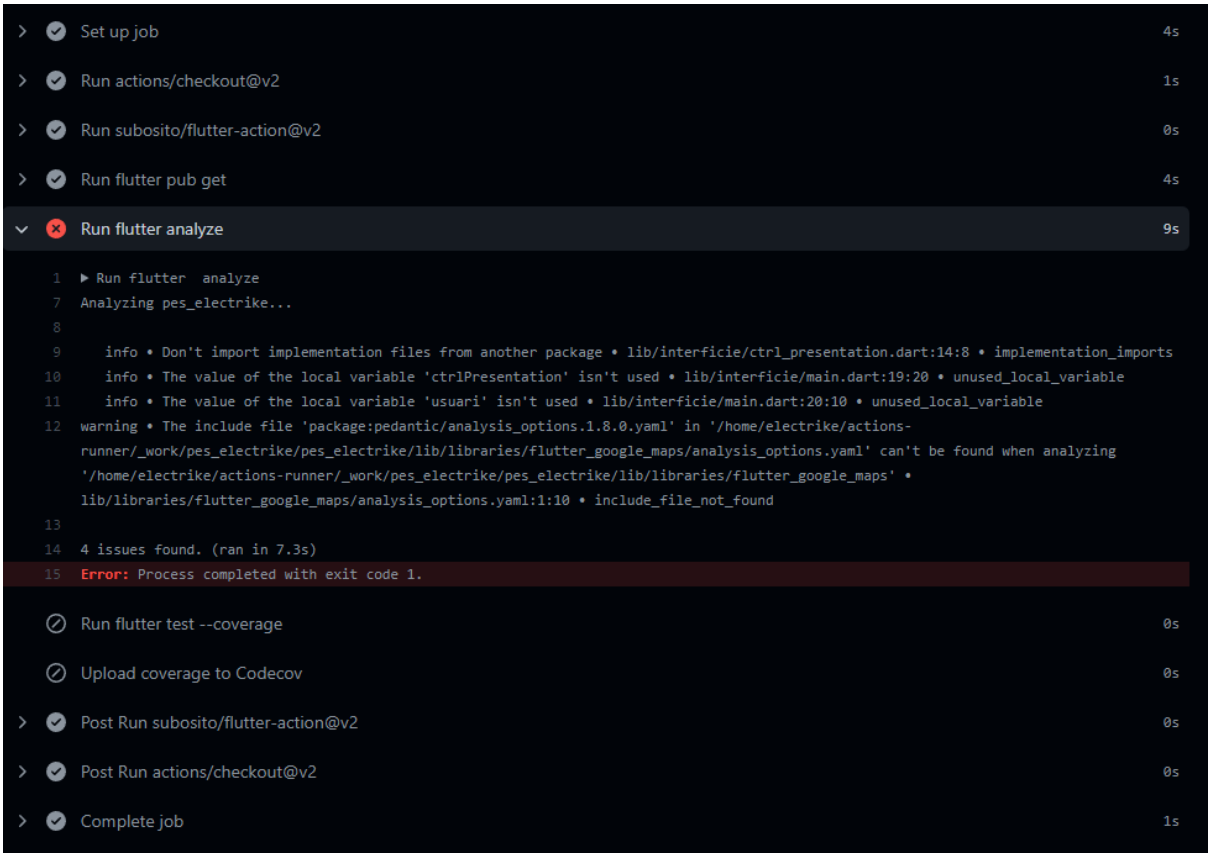
Després d'haver estat desenvolupant amb flutter hem pogut observar que implementa nativament un analitzador de codi estàtic, d'aquesta manera hem evitat utilitzar eines com SonarQube.

A més, aquesta anàlisi es realitza automàticament tant en les màquines individuals de cada integrant com per al CI del nostre repositori Github. Tenir-ho localment aporta gran avantatge sobre el desenvolupament, ja que no teníem la necessitat d'eines de tercers i sobretot ho fa en directe estalviant-nos temps. D'aquesta manera ens assegurem de pujar un codi net.

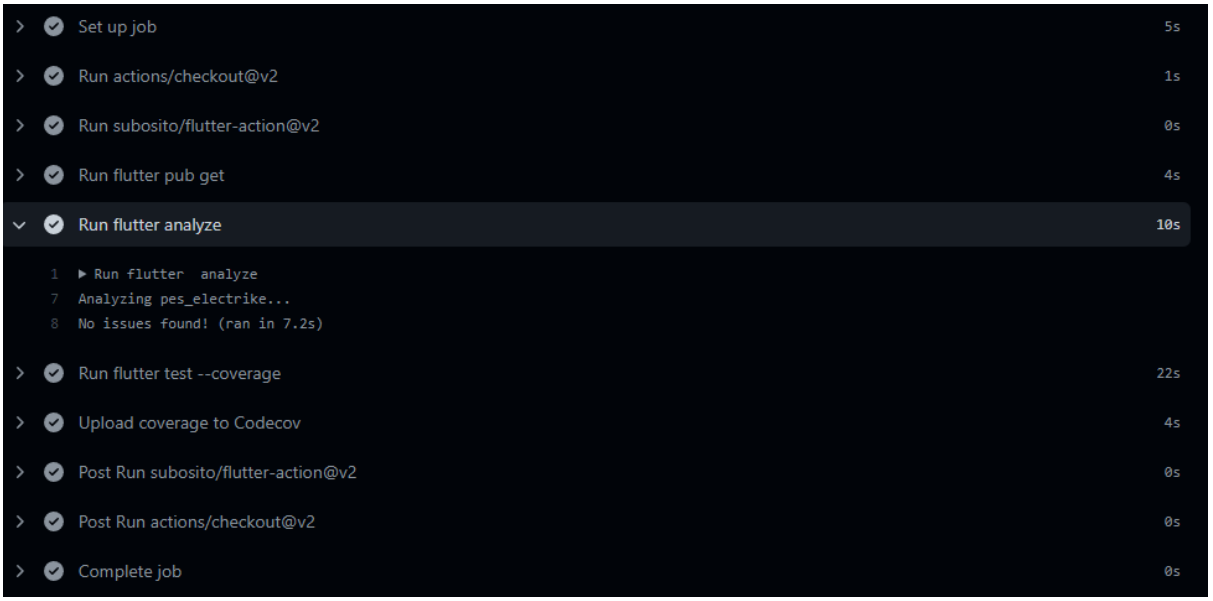
Tenir aquest sistema ens ha ajudat a tenir un codi més net i organitzat perquè, sense dependre de qui estigués programant, es pot llegir i interpretar sense problemes.

En el següent exemple podem veure com el CI de Github no va acabar correctament a causa de problemes amb l'anàlisi. L'explicació dels errors és bastant intuïtiva, fins i tot arribant a dirigir-te al punt exacte emprant el IDE localment. A més, si algun d'aquests no és comprensible, flutter té una documentació on explica cadascun d'ells, donant fins i tot exemples de com evitar-ho.

El canvi més notori i que ens és molt útil va ser tenir el codi net de prints, llevar variables i importància en desús i mantenir uns noms homogenis.



En la següent imatge es pot observar com s'ha realitzat el workflow degudament sense errors.



Ja tenim el sistema muntat per a realitzar els test automàticament quan afegim algun.

4.6. Estratègia de proves

Definirem un conjunt de joc de proves limitat per tal d'anar fent testing automatitzat cada cop que hi ha un *commit* al GitHub, aquestes proves encara no estan implementades a GitHub.

Actualment, segons les tasques, fem *mains* per tal de testejar la nova funcionalitat. Si permet provar-la printant valors per consola, la persona que la ha implementat la funció pot veure el resultat i comprovar si funciona de manera correcta, aquestes són principalment les referents a domini i les de l'API. En el cas de que aquesta tasca es pugui provar directament en l'aplicació, com són principalment les de la interfície, s'implementen i es proven per diversos companys per tal de trobar errors i bugs o casos no contemplats que puguin dur al crash de l'aplicació. Ara per ara no hem trobat necessari implementar tests unitaris i no els preveiem desenvolupar en el futur.

L'últim mètode que utilitzem per provar tasques es el Postman. En aquest *workspace* comprovem que les funcions de l'API funcionan correctament i retornen la informació o valors esperats. Emprem sobretot aquest mètode a l'hora de fer els tests d'integració entre les capes per confirmar que la comunicació entre aquestes és correcta.

4.7. Gestió de configuracions

Per al moment actual de desenvolupament tenim diverses configuracions implementades.

La primera i més important seria el Continuous Integration, el qual executa proves de codi estàtic. Encara que tots els canvis es pugen correctament al github per a evitar perdre codi, al moment de realitzar merge entre branches, Github avisa que les corresponents branches poden tenir futurs problemes a causa d'aquests errors.

A continuació tindríem el Continuous Deployment, el qual ens permet fer un build automàtic tant de l'aplicació web com android perquè siguin fàcilment accessible des de Github.

Finalment, hem configurat dos ecosistemes de servidor.

El primer seria de testing que ho tenim allotjat en un servidor privat propi, encara que accedit a través de DDNS. Aquest mateix ho utilitzem per a desenvolupar les API, fer les primeres proves de noves implementacions, etc.

El segon seria AWS, el qual utilitzem més per a producció, és a dir, el que tindrà accés un usuari final.

D'altra banda, a causa d'un bug, vam tenir que implementar un servidor proxy per fer les peticions a una api de Google. Aquesta solució fa servir CORS-Anywhere conjuntament amb heroku perquè necessitem un domini amb certificat ssl.

4.8. Interacció amb companys

Aquesta secció fa referència a les APIs que hem d'usar d'un altre grup i la que hem de proporcionar nosaltres.

Pel que fa a les relacions d'oferta i consum de serveis a tercers ja les hem vist a l'apartat Funcionalitats pactades.

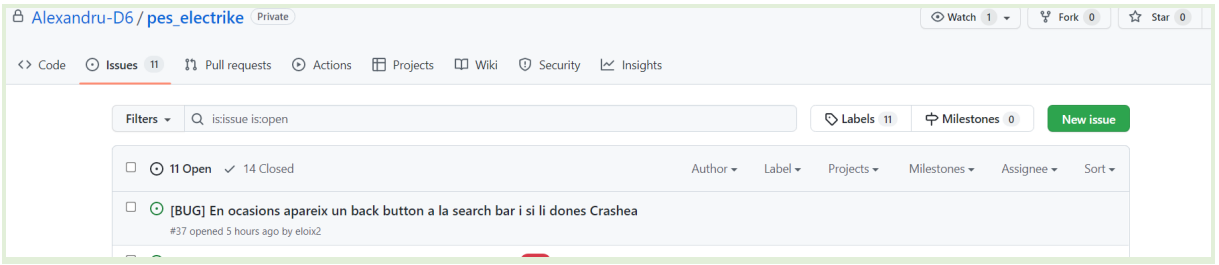
Pel nostre projecte usarem l'API de la qualitat de l'aire, ja que creiem que pot encaixar amb la nostra app. D'altra banda, nosaltres proporcionarem els punts de recàrrega (i possiblement punts d'estacionament Bicing) més propers donada una localització concreta. Aquesta API s'ofereix al grup de cases sostenibles.

En aquest sprint ja hem completat aquestes relacions i en el nostre cas falta integrar el servei ofertat per Happy Lungs dins el nostre sistema. La integració hauria de ser relativament senzilla en quant a que és adaptar una funció ja implementada (rutes per carregadors) per què optimitzi la ruta amb les dades de contaminació. La documentació d'aquesta API la podem trobar a l'annex i visualitzar addientment a través de Swagger.

Aquestes negociacions les fem a través d'un portaveu que és l'encarregat de comunicar-se amb els altres equips i comunicar les necessitats de la nostra aplicació.

4.9. Gestió dels bugs

La gestió dels bugs es fa mitjançant GitHub, on hi ha un log amb tots els bugs a la finestra d'issues i els anem solucionant de mica en mica, on a més fem una descripció d'aquests i què hem fet per solucionar-los a part d'assignar-los. D'aquesta manera tenim tots el problemes actuals agrupats en una única finestra i podrem fer un millor seguiment dels mateixos.



Aquests també es comenten en el grup per tal de que l'equip sigui conscient dels bugs. A més de que quan un integrant te temps i esta implicat en la part que hi ha un bug ja sigui perquè ell mateix o ha programat o perquè es dedica a programar coses semblants i relacionades, ho intenta arreglar.

4.10. Tractament RNFs

Els requisits no funcionals estan posats al Taiga, on allà anem modificant-los i adequant-nos a les necessitats. Alguns d'aquests RNFs les considerem com a User Stories senceres i, d'altres, els tenim apuntats en el llistat de requisits no funcionals i els tenim en consideració constantment com podrian ser:

-Requisits d'Aparença: on l'equip de desenvolupament ha de certificar que compleix els estàndards i patrons estètics acordats per tal que es vegi com a una marca unificada.

-Requisits de Personalització i Internacionalització: ens assegurem que l'usuari pugui canviar l'idioma de l'aplicació.

-Requisits d'Accessibilitat: les persones daltòniques no tenen cap impediment per utilitzar les funcionalitats principals de l'aplicació, ja que cap informació rellevant es mostra amb només diferenciació de colors. Les persones amb sordesa no tenen cap dificultat al fer servir l'aplicació, perquè cap informació es notifica per sorolls.

-Requisits d'Accés: els usuaris enregistrats disposaran de les funcionalitats al complet del producte, en canvi, un usuari que no ha iniciat sessió només podrà visualitzar el mapa, els punts de càrrega i Bicing.

- Requisits de Privacitat: els usuaris hauran d'acceptar aquest requisit per fer servir per complet l'aplicació i ha de ser conscient de l'ús que li donem i el que comporta.

Mantenim encriptades aquestes dades dintre de la base de dades del producte, a les quals només podrà accedir ell mateix i l'equip de desenvolupament, per tal de mantenir la privacitat de l'usuari.

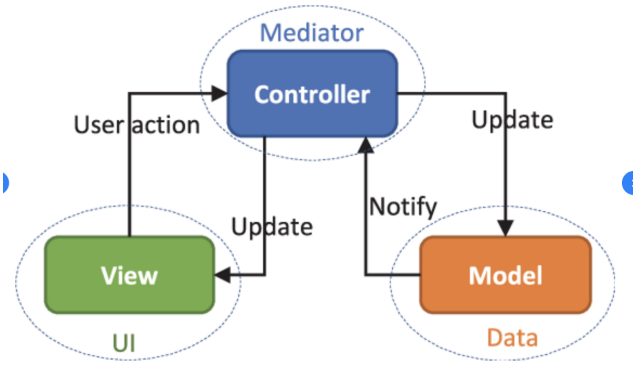
5. Descripció tecnològica

5.1. Concepció global de l'arquitectura

Quant a l'arquitectura del nostre programari podem destacar dos: MVC i disseny en 3 capes, on tots dos es complementen entre ells.

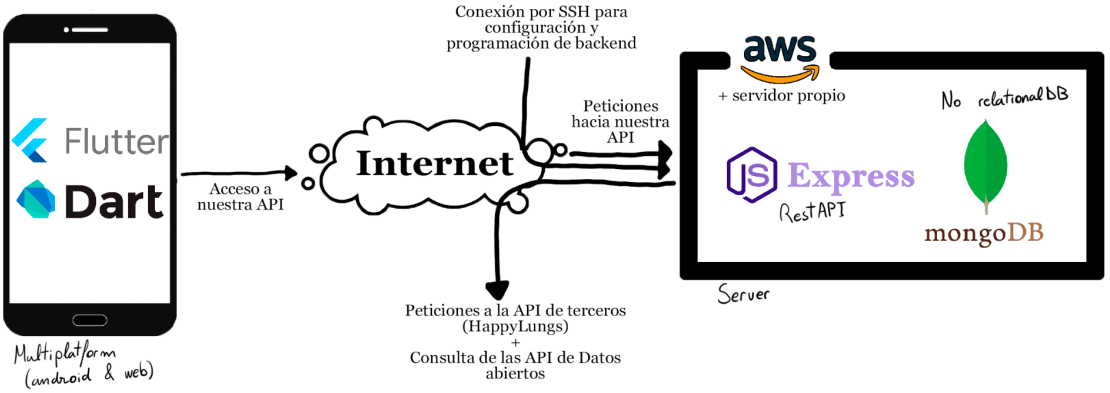
Per a començar explicaré el disseny en 3 capes: el nostre objectiu amb aquesta arquitectura era dividir el problema principal per a tenir de més petits, on podem diferenciar dràsticament el funcionament. Habitualment, i com l'hem utilitzat nosaltres, s'acostuma dividir en interfície, domini i base de dades. D'aquesta manera, les persones que estan desenvolupant la lògica de tota l'aplicació no s'han de preocupar de com es mostrarà la informació o com s'emmagatzema la informació. El que ens ha permès aquesta tècnica és poder dividir el treball de manera eficient i evitar solapaments de companys durant el desenvolupament.

Pel que fa el MVC, ens permet interconnectar totes les capes, ja que sense ell, hi hauria molt d'acoblament entre les capes i, en el cas de que s'hagin de fer canvis, seria molt complicat localitzar els canvis a fer. A més, fa que el codi quedi més net i, per consegüent, més mantenible.



Model-View-Control(MVC) design pattern.

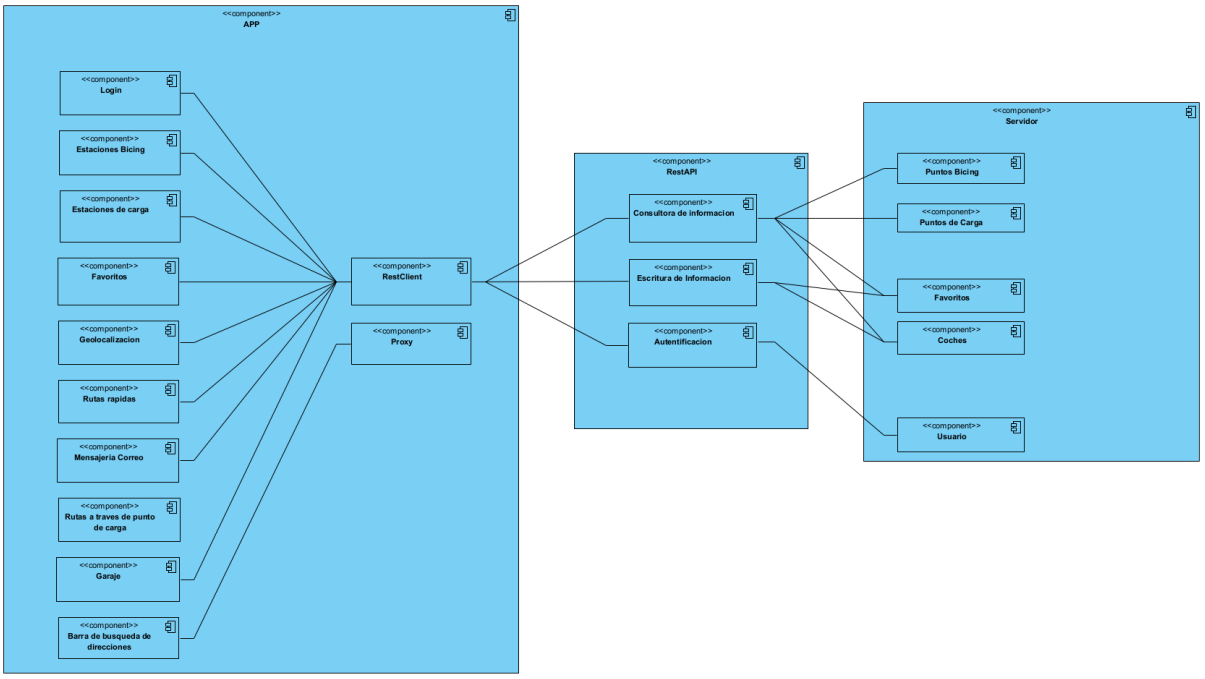
5.2. Altres diagrames d'arquitectura



En aquesta imatge es pot observar la nostra arquitectura física i tecnològica de l'aplicació. Començant amb la pròpia aplicació, la qual és multiplataforma tant en Android com en web.

Per a tota la persistència de dades, l'aplicació consulta el servidor a través d'una API pròpia. Actualment, el servidor està allotjat en un ordinador personal per a l'etapa de desenvolupament, encara que per al producte final es trasllada a AWS.

Quant al propi servidor, aquest implementa una API REST a través de ExpressJS connectada amb una base de dades no relacional MongoDB. A més d'oferir una API tant per a nosaltres com per a Build Green, també consulta tota informació varia d'altres API, entre elles la API de Happy Lungs.



En aquest diagrama de component es poden veure les principals funcionalitats de la nostra aplicació, API REST i de la base de dades.

5.3. Patrons de disseny aplicats

MVC

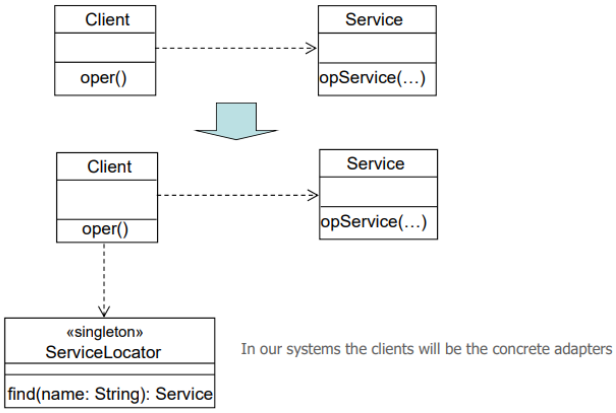
Com hem comentat anteriorment, hem dividit el sistema en 3 principals capes: Presentació, Domini i BD.

La idea és que la capa de presentació no "guardi" cap dada, sino que s'encarregui només de demanar aquelles que les capes inferiors contenen i, mostrar aquesta a l'usuari.

Això s'aconsegueix implementant un controlador de Domini qui, després de rebre l'ordre de la capa de Presentació, obté les dades de la BD, les procesa en el Domini i les envia a la capa de dades.

D'aquesta forma disminuim l'acoblament entre capes, fent el codi més mantenible i reusable.

Servicelocator



La necessitat de emprar aquest patró és semblant a l'anterior però, per disminuir l'acoblament amb els serveis. El nostre sistema hauria de comunicar-se amb un servei extern, el qual nosaltres no controlem i és fàcilment mutable. En el cas de que aquest servei decideixi canviar el tipus de dada que retorna o el nom de les seves funcions, seria molt complicat detectar els canvis que hauriem de fer en el nostre propi sistema.

Per tal de evitar això, concentrem les crides al servei en classes adaptadores i, implementant un Service Locator, tenim aquí la col·lecció de serveis que utilitza el sistema.

```dart
import 'package:flutter_project/domini/services/google_login_adpt.dart';
import 'package:flutter_project/domini/services/google_maps_adpt.dart';
import 'package:flutter_project/domini/services/google_places_adpt.dart';
import 'package:get_it/get_it.dart';

final serviceLocator = GetIt.instance;

void setUpLocator() {
  serviceLocator.registerLazySingleton<GoogleMapsAdpt>(() => GoogleMapsAdpt());
  serviceLocator.registerLazySingleton<GoogleLoginAdpt>(() => GoogleLoginAdpt());
  serviceLocator.registerLazySingleton<GooglePlaceAdpt>(() => GooglePlaceAdpt());
}


GoogleMapsAdpt get getMapsService {
  return serviceLocator<GoogleMapsAdpt>();
}

GoogleLoginAdpt get getLoginService {
  return serviceLocator<GoogleLoginAdpt>();
}

GooglePlaceAdpt get getPlaceService {
  return serviceLocator<GooglePlaceAdpt>();
}

```

Figura: Classe service_locator.dart del nostre projecte

Singleton Pattern

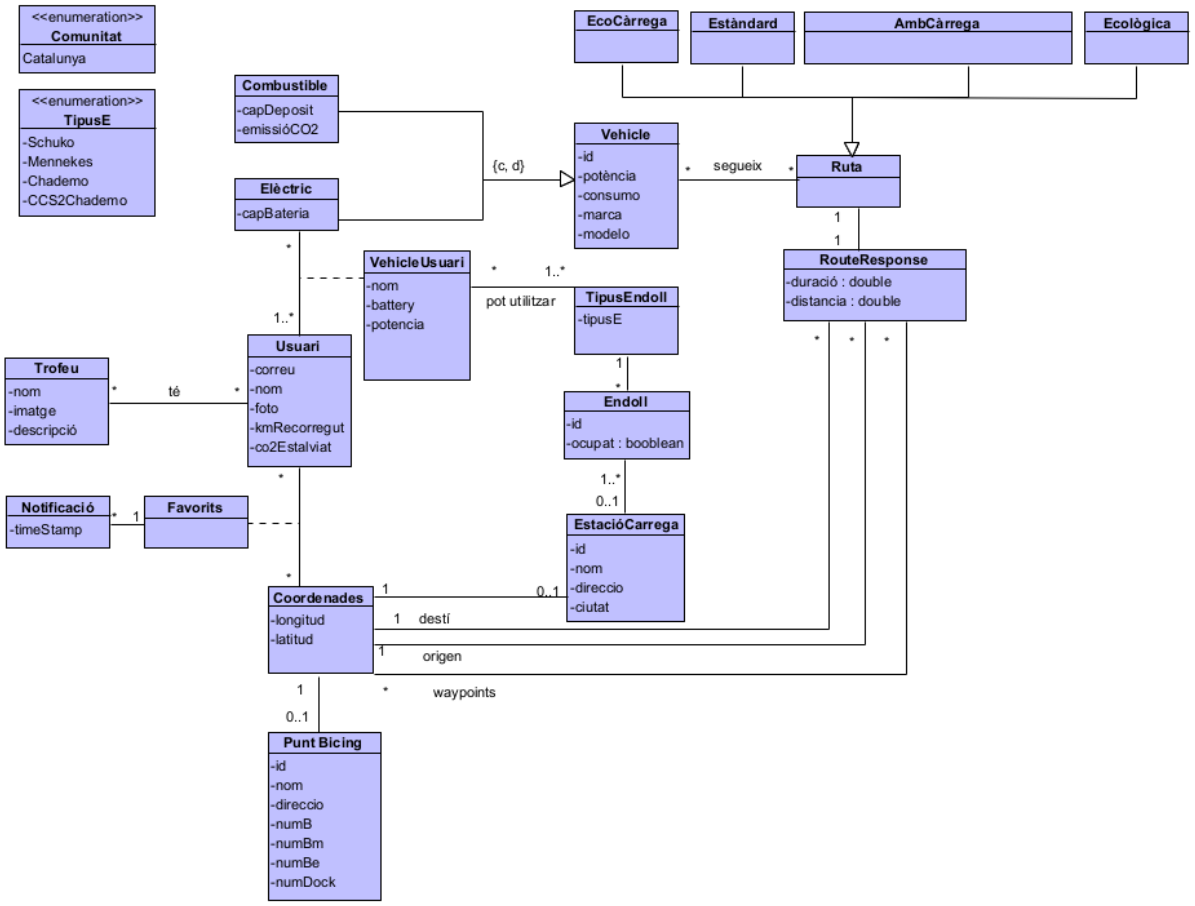
Un altre problema seria iniciar la sessió i assegurar-nos d'obtenir informació d'una mateixa instància de, per exemple, un usuari. En el cas de que volguessim fer actualitzacions en el conjunt sencer d'usuaris del sistema, voldriem només una sola copia d'aquesta. D'altra forma, podrien haver conflictes de sincronització.

Per aquesta raó, el Singleton Pattern s'utilitza per descriure que una classe pot tenir com a màxim una instància o cap (si no està inicialitzada). Tant el Servicelocator com els Controladors que hem vist anteriorment també compleix el patró Singleton perquè només tenen una sola instancia.

```
class CtrlDomain {
  static final CtrlDomain _singleton =  CtrlDomain._internal();
  factory CtrlDomain() {
    return _singleton;
  }
  CtrlDomain._internal();
}
```

5.4. Models de dades (UML)

El diagrama de classes UML de domini el podem trobar actualitzat a continuació. L'únic que hem canviat respecte al Sprint 2 es l'eliminació d'alguns atributs



- Claus primàries: (Vehicle, id), (Usuari, correu), (Coordenades, longitud+latitud), (PuntBicing, id), (EstacióCàrrega, id), (Trofeu, nom), (Endoll, id), (TipusEndoll, tipus)

- Una ruta no pot tenir el mateix punt d'origen que de destí.

- Un usuari no pot utilitzar un endoll a una estació de càrrega en el que el seu vehicle elèctric no hi sigui compatible.

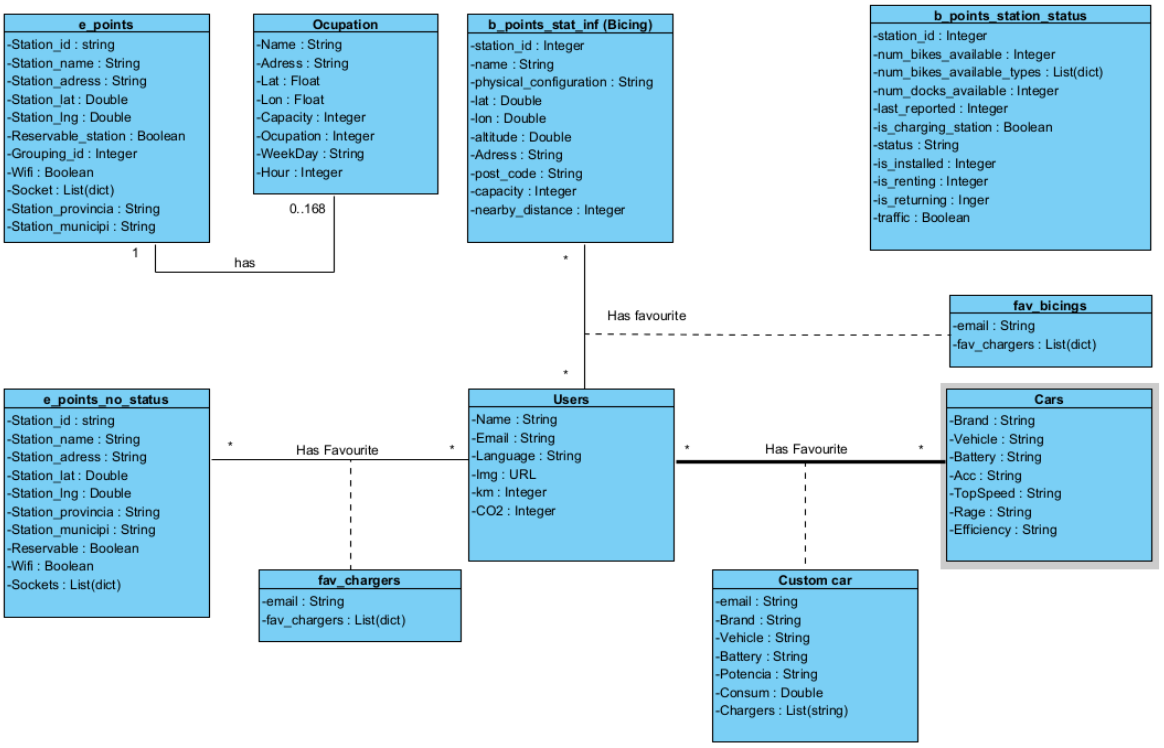
- L'usuari no pot tenir com a destinació una estació de càrrega amb el que el seu vehicle no tingui endoll compatible.

- Donada una coordenada que es un punt de Càrrega no pot ser d'un punt de Bicing ni viceversa:

- L'usuari només pot marcar com a favorit estacions de càrrega o punts Bicing.

En el nostre cas, ens hem trobat un gran repte al plantejar el esquema de la BBDD, ja que estem fent servir una BBDD no relacional, on s'organitza per documents, és a dir, no hi ha claus primàries ni relacions amb altres taules. No obstant, hem decidit representar-ho amb les relacions que fem servir nosaltres (tot i no ser de PK).

El diagrama de classes UML que descriu la base de dades és la següent (actualitzat segons sprint 2):



Com hem comentat anteriorment, no hi ha claus primàries com a tal, però si que nosaltres usem diversos atributs per tal d'identificar les dades, aquest són:

- (e_points, Station_lat + Station_lng)
- (e_points_no_status, Station_lat + Station_lng)
- (b_points_stat_inf, lat +lon)
- (b_points_station_status, lat +lon)
- (Users, email)
- (Cars, Brand+Vehicle)

- (Fav_chargers, email)
- (Fav_Bicings, email)
- (Custom_car, email)
- (Ocupation, Lat+Lon)

5.5. Altres aspectes tecnològics

Instrumentació

Nombre de servidors

Després d'estar fent servir durant les dues primeres setmanes un servidor AWS EC2 per gestionar les dades de la nostra aplicació, vam decidir canviar, ja que no ens estava donant els resultats esperats en quant a posibilitat de desenvolupat codi remotament a traves de SSH. L'Alexandru va decidir muntar ell un servidor local a casa seva, fent servir un portàtil amb una distribució Ubuntu Server 20.04 LTS. Des que utilitzem aquest nou servidor, no ens ha donat cap error i les peticions a ell són molt ràpides. No obstant això, per al producte final (producció) tindrem tots el nostres serveix de API y DB en AWS EC2.

D'altra banda, a causa d'un bug, vam descobrir que necessitarem un servei de proxy per fer peticion a API desde web. Per això, vam desplegar un petit script a heroku que fa de servidor proxy personal. Això no es posible ferlo ni en els servidores Amazon ni el personal peque no tindrem un domini web amb certificat SSL.

Configuració servidors

Dins el servidor hi haurà una base de dades MongoDB la qual ofereix un sistema NoSQL per emmagatzemar dades. Per l'altra part, també tenim implementat una Rest API amb el framework Express JS per tal d'accedir a les nostres dades de forma molt fàcil sense cap protocol ssh.

En quant a les dades obertes que usarem, el servidor serà l'encarregat de fer peticions per actualitzar l'estat de tots els punts de càrrega i de Bicing. D'aquesta manera, el dispositiu mòbil o la web poden consultar les dades post processades directament al servidor.

Addicionalment, per evitar fer ús dels minuts de Actions que ofereix Github (workflows per realitzar CI/CD) ja que son bastant limitats. Vam traspassar tota la carga de treball al nostre servidor fent ús de Github com a intermediari. Això va ser posible perquè Github ofereix aquesta opció, que es pot consultar a través d'aquest enllaç: GitHub self hosted runners

Nombre de BDs

Tenim una sola base de dades amb diverses taules, aquestes són:

- Usuaris
- Cotxes d'usuaris
- Carregadors elèctrics (estàtics)
- Carregadors elèctrics (dinàmics)
- Punts Bicing (estàtics)
- Punts Bicing (dinàmics)
- Cotxes elèctrics
- Punts preferits de Bicing
- Punts preferits de carregadors
- Ocupació

Versionat de BDs

Pel motiu d'estar utilitzant una base de dades no relacional com és MongoDB ens permet poder actualitzar les taules sense tenir la necessitat de migrar les dades a una nova taula. Per tant, això ens dona bastanta llibertat a l'hora de decidir d'afegir un nou paràmetre. A més, fem servir una BD no relacional, ja que ens centrarem més en el rendiment que ens pot oferir per fer consultes i no tant en l'emmagatzematge de dades.

Nombre de llenguatges

Principalment utilitzarem tres llenguatges diferents:

- Dart: per al desenvolupament general de l'aplicació Android, per a dispositiu mòbil, i per la pàgina web.

- NoSQL: per administrar les peticions que li arriben a la base de dades.

- JavaScript: per a crear l'API de la nostra aplicació.

- Python: Per fer el script de recopilació de dades dels punts de recàrrega de Barcelona.

APIs

API oferida

La nostra API serà una integració per tota la base de dades, és a dir, tots els accessos al servidor, per part de l'aplicació, es farà a través d'aquesta amb l'objectiu d'evitar fallos de seguretat. D'aquesta manera evitem connectar-nos directament al servidor.

El nostre servei ofereix dues funcions. Aquestes funcions han estat acordades amb els membres de l'equip de *Build Green*. Oferim la possibilitat de consultar quines són les estacions de recàrrega de vehicles elèctrics i punts bicings propers. És a dir, donada una ubicació i un radi, dones aquest punts que estarien dins el cercle demanat.
Per obtenir més informació, dirigiu-vos a la documentació d'aquesta API. Aquesta documentació està en castellà per petició de l'equip *Build Green*.

API interna

Actualment, a nivell de gestió interna la app recull i fa la petició de dades per mitjà d'una API desenvolupada pel Xavier Coll. La documentació està en progrés i s'entregarà per complet al darrer sprint.

Nombre d'APIs externes

- Google Sign in
- Google Maps API
- Directions API
- Places API

Consum servei

La nostra idea serà un cop aconseguits els punts de contaminació de Catalunya, poder calcular la ruta més ecològica donat un punt A i un punt B. La ruta més ecològica serà aquella que passi pels punts menys contaminats.
En cas que això no sigui possible per restriccions indefinides, tenim pensat canviar el color de la ruta en funció dels paràmetres

Subministrament servei

Aquest servei ja ha estat proporcionat juntament amb la documentació. Estem a l'espera de que el provin i ens comentin si els falta alguna cosa o hi ha cap error.

Consum de dades obertes

Pel que fa a les dades obertes fem servir les següents:

- Estat estacions de Bicing
- Informació sobre les estacions de Bicing
- Informació sobre tots els vehicles elèctrics
- Estacions de recàrrega per a vehicles elèctrics
- Informació de totes les marques de vehicles
- Informació de les estacions de recàrrega de vehicles elèctrics

Modificació de llibreries lliures

Durant aquest sprint vam anar modificant dues llibreries ja que no disposaven de les funcionalitats necessàries per al nostre objectiu.

La primera és la de Google Map, encara que estigui bastant completa, li falten funcions com per exemple retornar informació sobre una ruta, poder crear rutes amb més d'un punt, poder crear clústers de marcadors de manera senzilla. Per aquesta raó vam importar aquesta llibreria i la vam afegir al nostre projecte per a poder-la modificar a les nostres necessitats.

D'altra banda, quant als clústers de marcadors, per a evitar haver de crear des de zero l'algorisme, utilitzem una llibreria que per a la plataforma d'Android funcionava com era d'esperar però en Web no. Per aquesta raó, la vam haver de modificar per a fer-la compatible amb la nostra solució.

Destacar que totes aquestes llibreries tenen llicències que permeten la seva modificació.

Utilització d'eines de desenvolupament

Ús de Frameworks

- Google Maps:

Aquest és usat per tal de poder tenir un mapa a l'app, consultar rutes, mirar punts de recàrrega...

- Flutter:

S'ha escollit aquest llenguatge, ja que per tal de fer una aplicació multiplataforma, és la millor opció.
És un framework dissenyat per Google pensat per desenvolupar programari multiplataforma sense haver de refer el codi.

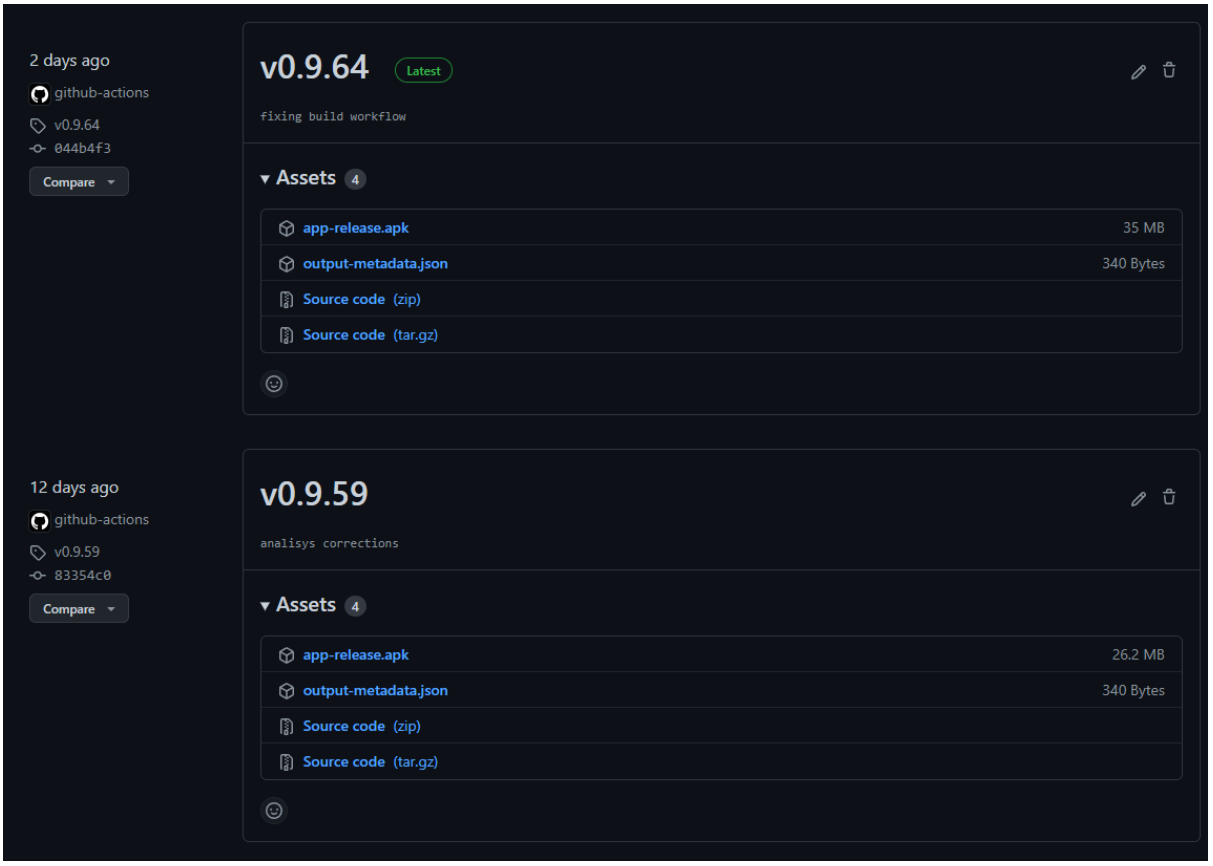
- Express JS: es un framework basat en node.js amb el objectiu de crear una API que pot accedir a diversos bases de dades, entre elles MongoDB.

Integració Continua

Pel que fa a la integració continua, tenim el Github automatitzat amb l'objectiu que amb cada push a una de les branques, tant principals com secundàries, estiguin sense errors i compleixin amb els requisits de qualitat. Per aconseguir tot això, farem servir les eines natives de flutter; flutter analyze. Tot això s'implementarà amb un workflow automatitzat de Github.

Desplegament (deployment)

En quant al Deployment continuat, farem servir un workflow de Github per tal de tenir automatitzat la tasca de fer builds del projecte y penjar-les directament al repositori de Github. D'aquesta manera podrem tenir un seguiment de totes les versions de l'aplicació segons vagi avançant durant els diferents Sprints.



Tan mateix com el CI, utilitzarem la eina native flutter build per fer això. Destacar que es farà build tant de l'aplicació web com la de android, amb la possibilitat de penjar la web per poder accedir públicament desde qualsevol lloc, encara que això està de moment en investigació per veure com funciona les Github Pages.

Configuració del servidor

Per a començar, la nostra base de dades va estar allotjada en Amazon Web Services (AWS) ja que és un dels proveïdors més grans de servidors i volíem aprendre com funciona el seu ecosistema. D'altra banda, volíem evitar l'ús de Virtech per a no haver de bregar amb configurar la VPN en els mòbils.

Dit això, vam triar utilitzar el servei Elastic Compute Cloud (EC2) ja que ens oferia una màquina virtual amb un sistema operatiu tipus Ubuntu. La primera elecció va ser usar una de les ISO que oferia Amazon. Però desafortunadament, després d'intentar instal·lar els programes necessaris i la seva respectiva configuració, vam decidir tornar enrere i utilitzar Ubuntu ja que estem més familiaritzats amb això. Al final va resultar molt més senzill.

AWS ofereix aquest servei EC2 de manera gratuïta al llarg d'1 any conjuntament amb 30GB d'emmagatzematge, més que suficient per al nostre propòsit.

La primera configuració que vam haver de fer és la del propi servidor, en quant a processador vam triar t2.micro (el pla gratuït), configurat amb dos discos d'emmagatzematge; un d'ells per al programari necessari i l'altre dedicat solament per a la base de dades amb 15GB cadascun. Finalment el firewall perquè només nosaltres puguem accedir; deixant accés sobretot als ports 22 (SSH), 27017(MongoDB) i 3000(RestAPI).

Quant a l'últim punt de configuració del servidor, aquest va ser la clau ja que és el que permet connectar-se al servidor. La primera vegada el vam configurar perquè qualsevol IP pogués entrar al servidor tenint les corresponents credencials. La nostra sorpresa va ser que vam ser hackejats no només una, sinó que dues vegades, esborrant-nos les dades. Per sort encara no teníem res compromès ja que eren solament dades obertes, però ens va fer aprendre al fet que havíem de restringir l'accés. La part positiva d'aquesta experiència va ser que teníem a algú que comprovava les vulnerabilitats.

Per tant, la configuració de servidor quedaria així:

5.5.1.1. Firewall

▼ Reglas de entrada

Filtrar reglas

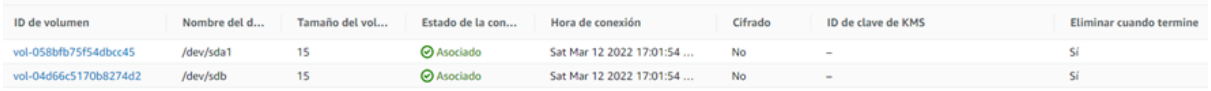
ID de la regla del grupo ...	Intervalo de p...	Protocolo	Origen	Grupos de seguridad
sgr-012d16ff088ed68ed	3784	TCP	0.0.0.0/0	launch-wizard-1
sgr-089321fed5ae455bf	27344	TCP	0.0.0.0/0	launch-wizard-1
sgr-070ffef25b1b1f0f3	22	TCP	0.0.0.0/0	launch-wizard-1

▼ Reglas de salida

Filtrar reglas

ID de la regla del grupo ...	Intervalo de p...	Protocolo	Destino	Grupos de seguridad
sgr-00da5bc1a701ce1d0	Todo	Todo	0.0.0.0/0	launch-wizard-1

Emmagatzematge



ID de volumen	Nombre del d...	Tamaño del vol...	Estado de la con...	Hora de conexión	Cifrado	ID de clave de KMS	Eliminar cuando termine
vol-058bfb75f54dbcc45	/dev/sda1	15	Asociado	Sat Mar 12 2022 17:01:54 ...	No	–	Sí
vol-04d66c5170b8274d2	/dev/sdb	15	Asociado	Sat Mar 12 2022 17:01:54 ...	No	–	Sí

Configuració dels programes del servidor

GIT

Per a tenir un còpia de seguretat de totes les dades, usem un repositori de git per a emmagatzemar tots els codis o script que usarem en el backend. Per començar també guardem les dades mongoDB com a redundància, la qual cosa ens va ajudar quan vam sofrir els atacs de seguretat.

En cas de no venir instal·lat el GIT es pot instal·lar de manera senzilla amb:

```
1. sudo apt-get install git

2.
```

I per a accedir al repositori remotament vam decidir utiliza una clau SSH, que en cas de no disposar ja d'una, per a generar-la s'utilitza les següents comandes:

```
1. ssh-keygen

2.
```

El següent pas ja seria en github.com, anant a la configuració general del compte, hi ha un apartat “SSH and GPG keys” on podràs afegir de manera senzilla la key prèviament copiada.

MongoDB

Per a l'emmagatzematge de dades vam decidir usar MongoDB pel fet d'usar una base de dades no Relacional. Per a configurar-la seguim les instruccions oficials de la documentació.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

```
wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -

touch /etc/apt/sources.list.d/mongodb-org-5.0.list

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list

sudo apt-get update

sudo apt-get install -y mongodb-org


#Para iniciar/parar/reiniciar/estado el servicio de MongoDB
sudo systemctl start/stop/restart/status mongod

#Para eliminar MongoDB
sudo service mongod stop

sudo apt-get purge mongodb-org*

#En caso de path default
sudo rm -r /var/log/mongodb
sudo rm -r /var/lib/mongodb
```

Cal destacar que segons en què servei cloud o PC local s'està utilitzant unes certes versions, les més noves per exemple, poden no funcionar. Per tant, s'hauria de fer un downgrade de versió.

Com hem esmentat anteriorment, volíem tenir una còpia de seguretat en GitHub per tant vam haver de canviar el directori default de Mongo. Per a això vam haver de fer el següent:

```
#En el directorio destino
mkdir database #nombre opcional
sudo systemctl stop mongod

#en caso de querer guardar las tablas actuales
mv /etc/lib/mongo/* ./database/

sudo chown mongodb:mongodb -R ./database


#hay que cambiar la configuración de mongoDB
sudo nano /etc/mongod.conf

#En el apartado “dbpath” hay que cambiarlo hacia el anteriormente creado
#p.e. ./database
```

Ara ja podem obrir de tornada el servidor i podrem crear les taules i usuaris necessaris. Per entrar al servidor i crear un usuari admin devem a través de terminal:

```
mongosh

use admin

db.createUser({user: “admin”, pwd: “admin”, roles:[“root”]})

```

Per a poder accedir de manera remota segura cal configurar de la següent manera l'arxivi mongod.conf:

```
#Volvemos a entrar en mongod.conf

Sudo nano /etc/mongod.conf



#En el apartado bindIp deberemos agregar la DNS ipv4 publica del AWS justo después de añadir una coma

#Ademas, para mas seguridad en el sistema, podemos activar que para cualquier acceso se requiera de credencial, por ejemplo, el admin creado anteriormente.


#Descomentamos la línea “#security:” y en la siguiente línea agregamos “authorization: enabled”

```

He de destacar que per a tenir una major seguretat s'hauria de modificar el port default assignat al MongoDB.

A partir d'ara, per a accedir localment haurem de fer-ho de la següent manera:

```
Mongosh -u admin -p admin
```

I si volem accedir remotament es farà de la següent manera:

```
Mongosh mongodb://admin:admin@ipv4:27017/admin
```

Per a crear les taules de manera més senzilla hem utilitzat MongoDB Compass ja que és un IDE que permet gestionar les teves bases de dades remotament.

Express .js

Per a la part de la API personal, hem decidit utilitzar una RESTAPI ja que la nostra base de dades serà bastant estàtica al llarg del desenvolupament pel aquest motiu les peticions no canviaran massa.

Per a començar, hem de crear una carpeta on estarà allotjat tota la configuració i instal·lar tots programes necessaris:

```
1.  mkdir API
2.  cd API
3.
4.  sudo apt update
5.  sudo apt -y install curl dirmngr apt-transport-https lsb-release ca-certificates
6.  curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash –
7.  sudo apt -y install nodejs
8.
9.  npm install -g n
10. npm install -g npm@8.5.4 #por si no se instalo la ultima versión
11. npm install express
12. npm install mongodb
13.npm install cors
14.
```

Una vegada fet això ja podem crear un fitxer .js i crear una API amb express.js amb mongoDB.

Proxy CORS

En cas de tenir problemes amb CORS o peticiones HTML en general, una solució rapida i sencilla es utilizar una proxy. Encara que hi ha proxy gratuites, com Electrike podria ser una aplicació comercial, necessitem una solució pròpia. Per tant, vam haver de configurar un petit script de JS que s'està executant en un servidor heroku. Per a això vam haver de seguir els següents passos.

El primer pas és crear un nou repositori de GitHub per a poder fer push al servidor de Github. Dins d'aquest, es genera un fitxer index.js que allotgen el script de proxy, el qual s'utilitza la llibreria CORS-*Anywhere. Fet això podem executar els següents comandos. (Partint de la base que ja tenim instal·lat Express .js de l'apartat anterior)

```
1. curl https://cli-assets.heroku.com/install-ubuntu.sh | sh
2. heroku login -i
3.  heroku create
4.
5. sudo npm install cors-anywhere
6. sudo npm init
7.
8. git add .
9. git commit -m “first commit to heroku”
10. git push heroku main(master) #dependiendo de tu repositorio
11.
```

Una vegada fets tots aquests passos, la proxy serà l'enllaç a la web, que es podrà trobar en la dashboard de Heroku.

6. Referències

[1]. Documentació Flutter. Flutter. Consultada el 29 de març del 2022.

https://esflutter.dev/docs/resources/faq

[2]. Plantilla de Especificación de Requisitos Volere (Febrer 2006). James & Suzanne Robertson rectores del Atlantic Systems Guild. Consultada el 27 de març del 2022.

https://www.volere.org/wp-content/uploads/2018/12/template_es.pdf

[3]. Java Script. Mozilla.org Individuales. Consultada el 28 de febrer del 2022.

https://developer.mozilla.org/es/docs/Web/JavaScript

[4]. MongoDB (2021). MongoDB. Consultada el 26 de febrer del 2022.

https://www.mongodb.com/docs/

[5]. Deloitte (2022). Deloitte Touche Tohmatsu Limited. Consultada el 27 de febrer del 2022.

https://www2.deloitte.com/es/es/pages/technology/articles/ceremonias-scrum.html

ANNEX

DOCUMENTACIÓN API

La API ofrece los puntos cercanos de Bicing y cargadores eléctricos de Barcelona. Dado una ubicación (latitud y longitud) y una distancia en kilómetros se devuelve un listado en formato JSON con la información de los puntos que se haya escogido (bicing o cargadores). A continuación, voy a hacer una explicación de como usarla. Esta documentación se ha desarrollado en castellano por petición del equipo Green Build.

BICINGS CERCANOS

Esta función te permite conseguir en un formato JSON todos los bicings cercanos dada una ubicación y una distancia. Dada la distancia, te va a dar todos los puntos Bicings que estén dentro del radio establecido.

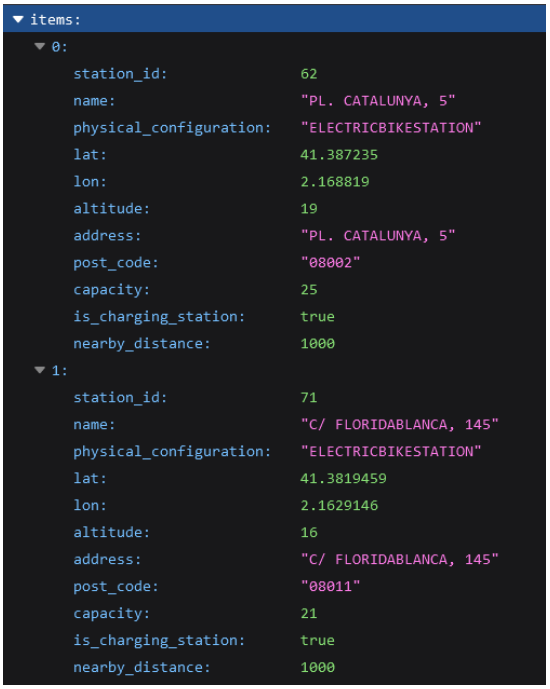
Para hacer una petición de esta función, mejor os pongo un ejemplo:
Se tiene que hacer una petición **GET** con este URI:

http://electrike.ddns.net:3784/near_bicings?lat=41.385503&lon=2.1634768&dist=0.5

Esta función devuelve todos los bicings cercanos dada la ubicación que vemos en lat y lon, que están dentro del radio de distancia 0.5 km (500 metros).

Es muy importante que los parámetros estén en minúscula. Probadlo en postman antes de ponerlo dentro del código.
Los parámetros que tiene que tener la URI es:

- http://electricke.ddns.net:3784/near_bicings
- lat : La latitud de la ubicación. Cuantos más decimales, más preciso. Es importante que no haya ninguna coma, siempre poner los decimales separados por puntos. Ej: lat=41.0234 lat=43.23456 lat=12.3434
- lon : La longitud de la ubicación. Cuantos más decimales, más preciso. Es importante que no haya ninguna coma, siempre poner los decimales separados por puntos. Ej: lon=31.0234 lon=23.23456 lon=12.3434
- dist : Distancia máxima de los bicings cercanos que se desea. La distancia es en kilometros. Si se quiere poner decimales, importante que sean separadors por puntos. Ej: dist=0.5 dist=1 dist=4



Esto es un ejemplo de una respuesta de la API.
Todo se engloba en un items, y a partir de allí, es un array.

Podéis ver lo que incluye una estación de bicing en la imagen.

CHARGERS CERCANOS

Esta función te permite conseguir en un formato JSON todos los cargadores cercanos dada una ubicación y una distancia. Dada la distancia, te va a dar todos los cargadores que estén dentro del radio establecido.

Para hacer una petición de esta función, mejor os pongo un ejemplo:
Se tiene que hacer una petición **GET** con este URI:

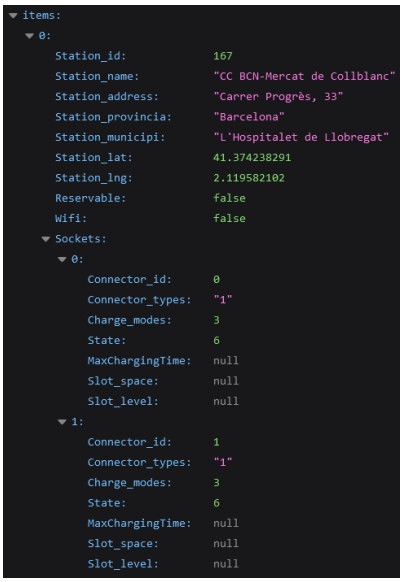
http://electrike.ddns.net:3784/near_chargers?lat=41.385503&lon=2.1634768&dist=5

Esta función devuelve todos los cargadores cercanos dada la ubicación que vemos en lat y lon, que están dentro del radio de distancia 5 km.

Es muy importante que los parámetros estén en minúscula. Probadlo en postman antes de ponerlo dentro del código.

Los parámetros que tiene que tener la URI es:

- http://electricke.ddns.net:3784/near_bicings
- lat : La latitud de la ubicación. Cuantos más decimales, más preciso. Es importante que no haya ninguna coma, siempre poner los decimales separados por puntos. Ej: lat=41.0234 lat=43.23456 lat=12.3434
- lon : La longitud de la ubicación. Cuantos más decimales, más preciso. Es importante que no haya ninguna coma, siempre poner los decimales separados por puntos. Ej: lon=31.0234 lon=23.23456 lon=12.3434
- dist : Distancia máxima de los cargadores cercanos que se desea. La distancia es en kilometros. Si se quiere poner decimales, importante que sean separados por puntos. Ej: dist=0.5 dist=1 dist=4



Esto es un ejemplo de una respuesta de la API.
Todo se engloba en un items, y a partir de allí, es un array. Dentro de Sockets, tenemos los diferentes cargadores que ofrece el punto de cargar.

Podéis ver lo que incluye una estación de carga en la imagen.

Documentació API Happy Lungs