

Mobilitat sostenible: Electrike

Sprint 1

[GitHub](#) - [Taiga](#) - [Project Record](#) - [Trello](#)



Cognoms	Nom	Roles	Correu electrònic		
			Google Drive	Taiga	GitHub
Asenjo Carvajal	Victor	Dev. team Member	victor.asenjo@estudiantat.upc.edu		victor.asenj@gmail.com
Balaer Morales	Eloi	Dev. team Member	eloi.balaer@estudiantat.upc.edu		eloi.x2@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		ferran.delavargaantoja@gmail.com
Dumitru Maroz	Alexandru	Scrum Master	alexandru.dumitru@estudiantat.upc.edu		alexandru666@outlook.es
Coll Ribas	Xavier	Dev. team Member	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

Introducció	1
Què s'ha fet?	1
Sprint master report	2
Descripció individual de treball	4
Alexandru Dumitru Maroz	4
Álvaro Rodríguez Rubio	4
Eloi Balaer Morales	5
Ferran de la Varga Antoja	5
Peilin Ni	6
Víctor Asenjo Carvajal	6
Xavier Coll Ribas	7
Avaluació de l'equip	7
Requisits	8
“Not” List - Actualitzada	8
Product Backlog	10
Requisits no funcionals - Plantilla Volere	12
10a. Requisits d'Aparença	12
11a. Requisits de Facilitat d'Utilització	12
11b. Requisits de Personalització i Internacionalització	13
11e. Requisits d'Accessibilitat	13
12a. Requisits de Velocitat i Latència	13
12c. Requisits de Precisió o Exactitud	14
12d. Requisits de Confiabilitat i Disponibilitat	14
12e. Requisits de Fortalesa o Tolerància a Fallades	14
13b. Requisits d'Interfície amb Sistemes Adjacents	15
14c. Requisits d'Adaptabilitat	15
15a. Requisits d'Accés	15
15c. Requisits de Privacitat	16

Gestió Taiga	16
Funcionalitats transversals	17
Serveis de tercers	20
Comunicació entre equips	20
Funcionalitats pactades	21
Cerimònia Agile	22
Report on the sprint planning, review and retrospective meetings	22
Updated release and iteration burndown charts and velocity chart	24
Metodologia	25
Visió global	25
Gestió de l'equip	25
Gestió projecte	26
Gestió repositori	26
Comunicació dins l'equip	27
Gestió de la qualitat	27
Estratègia de proves	29
Gestió de configuracions	29
Interacció amb companys	30
Gestió dels bugs	31
Tractament RNFs	31
Descripció tecnològica	31
Concepció global de l'arquitectura	31
Altres diagrames d'arquitectura	32
Patrons de disseny aplicats	33
MVC	33
Servicelocator	33
Singleton Pattern	34
Models de dades (UML)	35
Altres aspectes tecnològics	37
Instrumentació	37

Nombre de servidors	37
Configuració servidors	37
Nombre de BDs	38
Versionat de BDs	38
Nombre de llenguatges	38
APIs	39
API pròpia	39
Nombre d'APIs externes	39
Consum servei	39
Subministrament servei	40
Consum de dades obertes	40
Utilització d'eines de desenvolupament	40
Ús de Frameworks	40
Integració Continua	41
Desplegament (deployment)	41
Configuració del servidor	41
Firewall	42
Emmagatzematge	43
Configuració dels programes del servidor	43
GIT	43
MongoDB	43
Express .js	46
Referències	48

1. Introducció

1.1. Què s'ha fet?

Aquesta serà la primera iteració en la qual l'equip començarà a desenvolupar l'aplicació.

Per tal de poder implementar les tasques correctament, primer hem configurat l'espai de treball a Android Studio, després d'haver decidit el llenguatge i el framework amb el que treballarem. A més, hem decidit gestionar el repositori mitjançant CI i CD, el qual s'ha configurat també.

Un cop configurat tot l'entorn de treball de l'equip, hem començat a desenvolupar l'aplicació. 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:

- Navegació per l'aplicació
- Consultar punt de càrrega
- Consultar estació de Bicing
- Visualitzar mapa
- Log in
- Log out
- Sing up
- Imatge perfil
- Vista alternada
- Multiplataforma

1.2. Sprint master report

Scrum Master: Alexandru Dumitru Maroz

En aquest primer Sprint vam haver de canviar la manera en la qual treballem, ja que era el moment de desenvolupar el codi. Per a això ens dividim en tres grups bastant diferenciats: Interfície, Domini i Base de dades. Cadascun d'aquests grups representa cadascuna de les capes d'un programa amb disseny de 3 capes. Quant a la distribució va ser de 2 persones per a la interfície, 3 per al domini i 2 per al servidor. Aquesta última decisió hauria estat més bona si hi hagués tan sols 2 persones per a domini i 3 per a la interfície, ja que hem vist que hi ha molta més càrrega de treball, almenys per a aquest primer Sprint.

No obstant això, abans de posar-nos a desenvolupar el codi, vam haver de dur a terme dues tasques prèviament; triar les històries d'usuari definint les seves respectives tasques i preparar l'entorn de treball. Vam aprofitar el primer dia de classe presencial per a instal·lar totes les eines en cada ordinador dels integrants. A continuació ens vam posar a triar les històries d'usuari.

Per a aquest primer Sprint, vam voler ser conservadors, ja que no sabíem que tan bé ens desenvoluparíem com a equip i, a més, havíem d'aprendre un nou llenguatge de programació que no havíem utilitzat mai. En quant a les tasques de les mateixes històries, no vam tenir temps per a fer-les en aquell instant, per tant, per a la següent sessió, vam acordar en anar-les creant individualment. Durant aquesta sessió vam definir les hores que ens portaria crear cadascuna d'elles.

Les primeres setmanes de desenvolupament vam mantenir bastant bé els grups, però, al voltant de l'última, ens vam anar canviant segons les nostres necessitats. Per exemple, quan ens vam posar a ajuntar capes, era necessària una comunicació constant entre grups per a realitzar-ho i, per a aquesta tasca, 1 persona de cada subgrup va ser requerida.

Entre els problemes que hem tingut al llarg d'aquest Sprint es poden destacar els següents. Vam poder observar que el servei de Mapa OpenStreetMap era massa complex per a la nostra aplicació pel que vam decidir canviar-lo a Google Maps. D'altra banda, també vam tenir dificultats amb el servidor dels quals s'han de destacar; la poca seguretat que assignem a tot el sistema d'AWS, el que va provocar

que qualsevol persona pogués entrar en la nostra base de dades. També vam tenir un possible *bug* amb el ssh que no ens permetia editar fitxers fora de la terminal, la qual cosa va provocar que iniciarem el nostre propi servidor.

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, per a aquest primer sprint, encara que no vam poder acabar totes les tavernes predefinides al principi, sí que vam poder afegir altres funcionalitats que ens permetran agilitar el treball en els pròxims lliuraments.

1.3. Descripció individual de treball

Alexandru Dumitru Maroz

Durant aquest Sprint he pogut executar 3 tasques principals a destacar.

Inicialment, vaig formar part de l'equip Back-end on ens vam posar a configurar tot el que és servidor, base de dades, API. Comptant tots els errors que vam tenir, vam haver de configurar des de zero dos servidors d'AWS i un propi que està allotjat en un portàtil. Ja que, com explicarem més endavant, vam tenir uns certs problemes.

La següent tasca en la llista seria “Debugger”, ja que al llarg de tot l'esprint vaig estar ajudant a solucionar els problemes que poguéss arribar a tenir qualsevol integrant; tant d'execució de codi, problemes amb les eines, Github, instal·lant llibreries, etc.

Per a acabar, sobretot en l'última setmana, vaig estar ajudant a programar unes certes funcionalitats de l'aplicació i solucionant problemes de compatibilitat de llibreries.

Álvaro Rodríguez Rubio

He format part del subgrup de domini, el qual ens vam dedicar a fer les classes del representades al UML, específicament, m'he dedicat a fer les dels punts de càrrega, de bicin i coordenades.

Després d'haver creat les classes assignades, em vaig encarregar de programar el controlador de domini per interconnectar les tres capes del nostre sistema. Durant la programació del controlador vaig haver de fer petits canvis en diverses classes segons la demanda. També em vaig comunicar amb el subgrup de *back-end* per tal d'acordar les dades i les funcions necessàries.

A mesura que acabava de programar les funcionalitats del controlador de domini, vaig treballar conjuntament amb el Víctor Asenjo per unir les capes a més de rebre ajuda de l'Alexandru Dumitru en alguns problemes que vam trobar.

Eloi Balaer Morales

He format part del subgrup de *front-end*, més enfocat a la capa de presentació on hem implementat diferents funcionalitats generalment visuals i de navegabilitat per l'aplicació, tot i que al començament del projecte vam implementar aspectes generals que al final han passat a la capa de Domini.

Centrant-nos més en el que he tractat, m'he encarregat d'implementar funcionalitats com podrien ser la visualització del mapa, que ha canviat d'Open Street Maps a Google Maps, ja que diversos requisits no es podien complir fent servir el nostre disseny anterior. Vaig implementar una versió antiquada per a la visualització de *markers*. Juntament amb Víctor, vaig ajudar-lo a dur a terme algunes tasques en pair programming relacionades amb *markers* i menús, a més, m'he encarregat de realitzar el *log in* i la seva configuració, juntament amb Peilin i Alexandru. Finalment, a l'última etapa de l'*sprint* m'he encarregat de la part de rutes i direccions que podem mostrar en el mapa, per anar d'un punt a un altre.

Ferran de la Varga Antoja

Jo he format part de l'equip de desenvolupament de la capa de domini juntament amb l'Álvaro i la Peilin. Som l'equip que vam dissenyar el model UML i que ara hem implementat les classes. Concretament, m'he encarregat de les classes Usuari, VehicleUsuari, Trofeu i TipusEndoll.

D'altra banda, vaig estar treballant amb la Peilin traient els apunts d'assignatures anteriors per poder implementar alguns patrons, que explicarem en el punt 5.3.

També vaig treballar en la configuració de l'API de Google Log in. Vaig estar buscant com utilitzar les funcions i llibreries de Flutter per a l'API de Google Log in i vaig començar a implementar el *Log in* i *Sign Up* de Google juntament amb el seu botó, però vaig haver de necessitar ajuda d'altres integrants. Finalment, vaig ajudar a l'Álvaro fent crides a través de Postman a la nostra base de dades i analitzant les respostes corresponents.

Peilin Ni

Jo formo part del subequip que s'encarrega de la capa de domini del sistema, implementant el diagrama UML que vam definir amb el mateix grup que s'encarrega de Domini en l'entrega anterior, concretament, la part de vehicles (tant elèctrics com de combustible).

Un cop acabada la primera versió de la implementació de les classes, m'he dedicat a desenvolupar la comunicació de la capa de domini amb altres serveis, entenent i iniciant així la implementació del patró de Service Locator amb els serveis de Google Login i Google Maps. Abans de començar, vam haver de configurar les credencials de Google Developer, que vaig fer amb l' Eloi, per tal d'habilitar les funcionalitats de l'API.

A meitat del desenvolupament de la connexió amb els altres serveis, aquesta feina s'ha delegat a l'Alexandru. Des d'aquell moment m'he dedicat, de forma individual, en el desenvolupament de gran part de la documentació per l'entrega i la gestió de Taiga i Trello.

Víctor Asenjo Carvajal

Des de *front-end* ens hem encarregat principalment de tota la part que interacciona amb l'usuari i facilita les gestions de l'aplicació amb una bona usabilitat: Google Maps, visionat d'informació de punts rellevants (carregadors i bicings), fàcil logatge... A part de la implementació de vistes, he integrat també diferents elements d'interacció de l'usuari per obtenir informació de forma més eficient d'altres capes i dels serveis. A més, hem connectat aquesta capa amb domini per rebre informació actualitzada.

A mesura que ha anat avançant l'*sprint* he treballat de manera més propera amb l'Álvaro: l'encarregat de certes funcions de domini necessàries per sol·licitar informació a la nostra base de dades.

Xavier Coll Ribas

Jo, juntament amb l'Alexandru, ens hem encarregat de la capa de persistència de l'aplicació. Primer de tot, vam haver de preparar el servidor en el qual hi hauria la base de dades, la API i altres fitxers relacionats amb aquests descrits anteriorment.

Després de escollir el servidor i configurar-lo, vam muntar allà la BBDD.

Una setmana més tard, vam trobar problemes que ens estava donant aquella opció de BD (que explicarem més detalladament més endavant), els quals ens va fer haver de fer canvis i tornar a configurar el nou servidor.

Un cop tot configurat, vaig començar a desenvolupar la API. Les funcions de la API les vaig definir juntament amb l'Alvaro, que treballa sobre la capa de domini, on m'indicaven què necessitaven exactament. Posteriorment, vaig fer l'esquema relacional de la base de dades.

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
<i>Puntuació Sprint 1 (0-5)</i>	5	5	5	5	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 dona 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 tipus
Geolocalizació	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.	No mostra informació concreta d'establiments o llocs concrets
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	
No mostra gràfics de concurrència a les estacions bicing	
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.	
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/NA.



[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	In progress
	Eliminar punt de preferits	Sprint 1
	Activar Notificacions	N/NA
	Desactivar Notificacions	N/NA
	Multi idiomes	In progress
	Compartir ubicació	N/NA
Gestió Vehicles	Afegir un vehicle	In progress
	Eliminar un vehicle	In progress
	Modificar dades vehicle	In progress
	Introduir bateria inicial	In progress
	Escollir vehicle de viatge	In progress
Gamificació	Consultar trofeu	N/NA

	Guanyar trofeu	N/NA
Mapes i rutes	Ruta més ràpida	In progress
	Ruta amb menys contaminació	N/NA
	Ruta amb punts de recàrrega	N/NA
	Bicing propers	N/NA
	Recerca de direccions	In progress
	Vista alternada	Sprint 1
	Consultar Estació de Bicing	Sprint 1
	Consultar punt de càrrega	Sprint 1
	Gràfiques d'ocupació dels punts	N/NA
	Visualitzar punts favorits	In progress
	Visualitzar només punts favorits en el mapa	In progress
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](#).

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 distingeixen 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
Gamificació	L'aplicació assigna premis als usuaris segons els requisits predefinits dins del sistema.	N/NA
Refutació	L'usuari rep notificacions d'aquells punts que ell ha activat. Aquests avisos informen de l'ocupació dels punts.	N/NA
Calendari	L'usuari pot activar notificacions personalitzades, defineix un dia i una hora. Aquesta segueix l'horari definit al calendari estàndard de la zona.	N/NA
Multiidioma	Proporciona a l'usuari diferents idiomes per als textos de l'aplicació, en concret: català, castellà i anglès	Començat

- Geolocalització


Al principi volíem utilitzar les llibreries i APIs d'Open Street Maps, però al final vam canviar tot el que teníem (que no era poc, mapa, marcadors, clusters i pràcticament rutes) a Google Maps per la falta d'informació i problemes que ens van sorgir amb l'anterior llibreria.

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, que més endavant es tindran en compte els punts de càrrega per fer aquestes. 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

Al nostre projecte hem considerat que era necessari una gestió d'usuaris per a guardar els punts de càrrega favorits, els trofeus que conté un usuari, i els cotxes que posseeix aquest. Per a la realització d'aquest requisit, 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.

Com podem observar, el nostre projecte no cobreix en totalitat totes aquelles que es proposen en la taula del Tema 1, “Aspectes transversals típics”. Després de parlar amb el Silverio, hem acordat afegir altres dos aspectes transversals alternatius (que han sigut acceptats pel professor), que són els que es mostren en la taula següent.

Aspecte transversal	Descripció	Estat
Multiplataforma*	L'aplicació és compatible amb Android i web.	Acabat (versió 1)
Gràfics d'ocupació	Es podran veure gràfiques de les ocupacions dels punts Bicing i dels carregadors de Barcelona.	N/NA 

*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.

2.6. Serveis de tercers

Comunicació entre equips

Tenim 2 vies principals de comunicació i una en reserva per si falla alguna de les 2 primeres.

Les principals són WhatsApp pel que fa a consultes ràpides i plantegem reunions on-line mitjançant Discord per a discussions que requereixen més detall i que WhatsApp no pot abarcar. Al grup de WhatsApp només hi ha els "relacions públiques" (els encarregats de comunicar-se amb els altres subgrups) mentre que a Discord estem tots els integrants del grup 21 de PES 2021-22 Primavera.

Els comunicats rellevants es fan pel grup de WhatsApp. Així assegurem que informació rellevant no passi per alt i quedi soscavada sota altres missatges a Discord ja que els responsables de cada subgrup (els "relacions públiques") s'encarreguen d'informar al seu corresponent subgrup. Els implicats en el tema s'encarreguen d'acordar per Discord el dia de reunió per atacar el problema o el dubte...

Com a mètode de comunicació en cas de fallada de serveis de WhatsApp i/o 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	<i>Electrike</i>	<i>BuildGreen</i>
<i>Happy Lungs</i>	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	<i>Happy Lungs</i>	<i>Electrike</i>
<i>BuildGreen</i>	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	<i>BuildGreen</i>	<i>Happy Lungs</i>
<i>Electrike</i>	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

Un cop havíem establert el cost de cada història d'usuari i per tant, ja havíem fet el nostre product backlog, vam decidir durant el sprint planning les històries d'usuari que volíem treballar per aquest primer sprint. Vam pensar que les principals podien ser aquelles que fossin imprescindibles per la nostra app, com ara poder iniciar sessió, mostrar rutes o veure els carregadors. Les històries seleccionades van ser les següents:

Història d'Usuari	Puntuació
Login	2
Logout	2
Sign Up	3
Eliminar perfil	5
Imatge de perfil	5
Afegir punt a preferits	5
Eliminar punt de preferits	3
Vista alternada	3

Ruta més ràpida	13
Consultar estació Bicing	5
Consultar punt de recàrrega	8
Visualitzar mapa	13
Multiplataforma	13
Navegació de l'aplicació	5

Vam pensar que aquestes fossin les candidates per marcar l'inici del nostre projecte, ja que podien fer les funcionalitats més bàsiques de la nostra aplicació.

Un cop vam decidir **què** havíem de fer, vam planificar **com** ho hauríem de dur a terme. Com ha comentat anteriorment el nostre Scrum Master, l'Alexandru, vam separar-nos en tres equips diferents per tal de desenvolupar-les. Vam assignar-nos cadascú a un equip, i a partir d'aquell moment, vam començar amb a treballar. Cada equip va decidir en aquell moment per quines parts començar i un cop consensuat, ho vam explicar als altres equips del nostre grup. Això és degut a que certes històries d'usuari tenen relació directa amb altres grups, per exemple:

Guardar punt preferit (Interfície) → Crida per guardar el punt preferit (Domini) → Guardar el punt preferit a la BBDD (Persistència).

En aquell moment, tots vam decidir començar a treballar en les tasques que no tinguessin relació amb altres grups, ja que així al principi avançaríem més ràpid.

Sprint Review & Sprint Retrospective

Durant aquest primer *sprint*, encara no hem pogut fer ni la *review* ni la *retrospective*, això és degut a que tècnicament aquest no ha acabat. Tenim planejat fer-lo durant la pròxima sessió, i es parlarà de:

- Històries completades

Revisió de les històries completades i analitzar que ens ha mancat fer. Poder entendre per què no han estat complertes (mala planificació al *planning poker*, massa dificultat, no s'ha treballat prou...)

- Balanceig de temps pels components del grup

Intentar analitzar les hores dedicades pels components del grup, per exemple, veure si algú no ha treballat seguint la mateixa dedicació que els altres, o també si algú ha treballat massa.

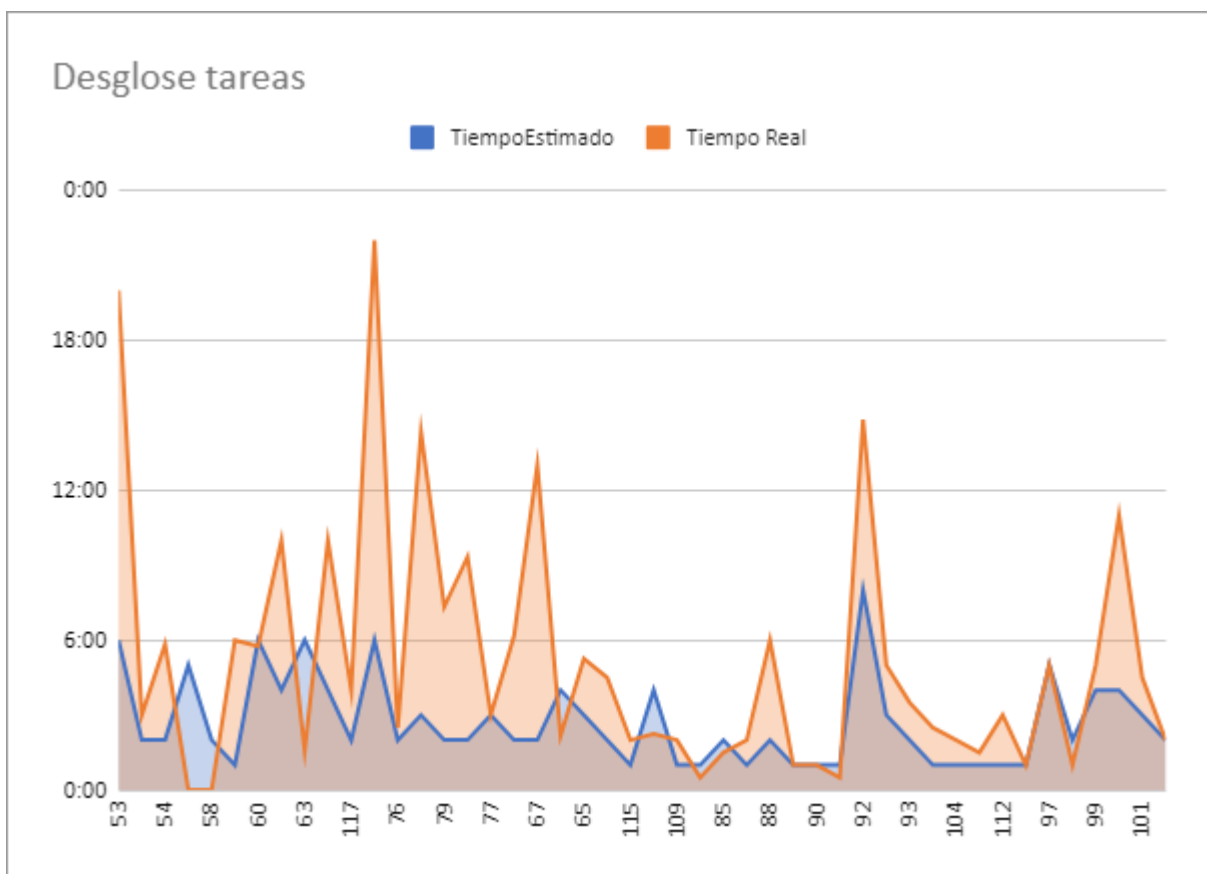
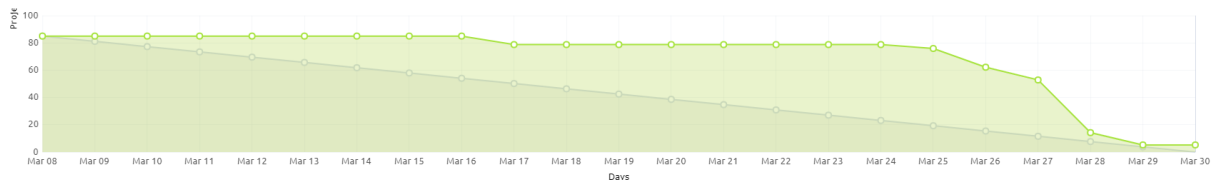
- Analitzar qualitat del codi

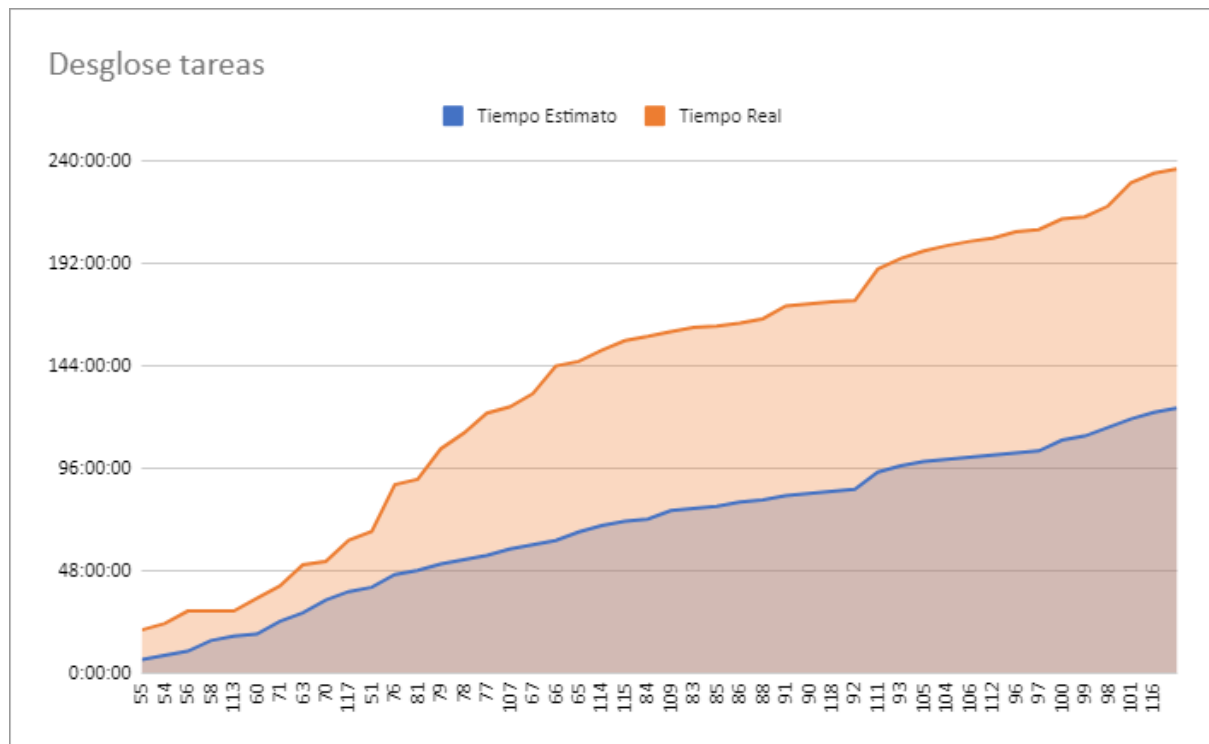
Veure si el codi segueix els estàndards establerts i intentar millorar la qualitat d'aquest.

- Organització de l'equip

Lligat amb un dels punts anteriors, parlar de com ha anat l'organització entre nosaltres, si hi ha hagut malentesos, què podem millorar o què ha fallat.

3.2. Updated release and iteration burndown charts and velocity chart





Pel que fa als gràfics hem pogut observar com les nostres expectatives de temps han sigut aproximadament la meitat del temps real dedicat. D'altra banda, també destacar del burndown chart com vam acabar quasi totes les tasques al final. Qual cosa no és la correcta, però cal mencionar que aquest projecte és la primera vegada programant en aquest llenguatge i framework, per tant, hem tingut un període d'adaptació que ha pogut provocar aquests temps.

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.

Per aquest Sprint, si podrem fer una sprint review juntament amb la M^a 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 aquest primer Sprint hem segmentat l'equip en tres subequips 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: Alvaro Rodríguez, Peilin Ni i Ferran de la Varga; i 2 per la BD: Xavier Coll i Alexandru Dumitru.


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 necessitaven 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.

Al Taiga tenim les històries d'usuari, 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  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. 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.

[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 o parlem sobre assumptes ràpids.

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 majoritàriament. 

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 subequips i, especialment, en la integració del projecte i resolucions de problemes, ja que la comunicació és més immediata. 

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. 

```
> ✔ Set up job 4s
> ✔ Run actions/checkout@v2 1s
> ✔ Run subosito/flutter-action@v2 0s
> ✔ Run flutter pub get 4s
▼ ✖ Run flutter analyze 9s
  1 ▶ Run flutter analyze
  7 Analyzing pes_electrike...
  8
  9 info • Don't import implementation files from another package • lib/interficie/ctrl_presentation.dart:14:8 • implementation_imports
 10 info • The value of the local variable 'ctrlPresentation' isn't used • lib/interficie/main.dart:19:20 • unused_local_variable
 11 info • The value of the local variable 'usuari' isn't used • lib/interficie/main.dart:20:10 • unused_local_variable
 12 warning • The include file 'package:pedantic/analysis_options.1.8.0.yaml' in '/home/electrike/actions-runner/_work/pes_electrike/pes_electrike/lib/libraries/flutter_google_maps/analysis_options.yaml' can't be found when analyzing
    '/home/electrike/actions-runner/_work/pes_electrike/pes_electrike/lib/libraries/flutter_google_maps' •
    lib/libraries/flutter_google_maps/analysis_options.yaml:1:10 • include_file_not_found
 13
 14 4 issues found. (ran in 7.3s)
 15 Error: Process completed with exit code 1.

  ✔ Run flutter test --coverage 0s
  ✔ Upload coverage to Codecov 0s
> ✔ Post Run subosito/flutter-action@v2 0s
> ✔ Post Run actions/checkout@v2 0s
> ✔ Complete job 1s
```

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

```
> ✓ Set up job 5s
> ✓ Run actions/checkout@v2 1s
> ✓ Run subosito/flutter-action@v2 0s
> ✓ Run flutter pub get 4s
> ✓ Run flutter analyze 10s
  1 ▶ Run flutter analyze
  7 Analyzing pes_electrike...
  8 No issues found! (ran in 7.2s)
> ✓ Run flutter test --coverage 22s
> ✓ Upload coverage to Codecov 4s
> ✓ Post Run subosito/flutter-action@v2 0s
> ✓ Post Run actions/checkout@v2 0s
> ✓ Complete job 0s
```

De moment, no tenim cap gràfic de *coverage* pel fet que no vam veure necessari fer cap test unitari important. Així i tot, 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. A més, segons les tasques, farem servir TDD com a estratègia de desenvolupament del codi que sigui necessari. No ho aplicarem per aquelles funcionalitats més trivials i simples com “Getters i Setters”, com es pot observar en el codi actual, sinó que ho farem més endavant, especialment en el procés de desenvolupar les rutes ecològiques o amb punt de càrrega.

Per al moment, no hem vist la necessitat de crear cap joc de proves ja que no hem implementat cap algorisme o funció complexa.

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 tant proves de codi estàtic com test unitaris. 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 tindriem 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.

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.



Encara no tenim les APIs ja que ens trobem en les primeres fases del nostre projecte. Tot i això, ja hem parlat amb els altres grups i hem arribat a un acord de quines coses necessitarem. Ens hem repartit qui oferirà el servei a la resta i qui l'utilitza.

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.

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 els problemes actuals agrupats en una única finestra y podrem fer un millor seguiment dels mateixos.

4.10. Tractament RNFs

Els requisits no funcionals estan posats al Taiga, on allà anirem modificant-los i adequant-nos a les necessitats. Alguns d'aquests NFR les considerem com a User Stories senceres i, d'altres, que estan incloses en altres US i, per tant, es representen com a tasques.

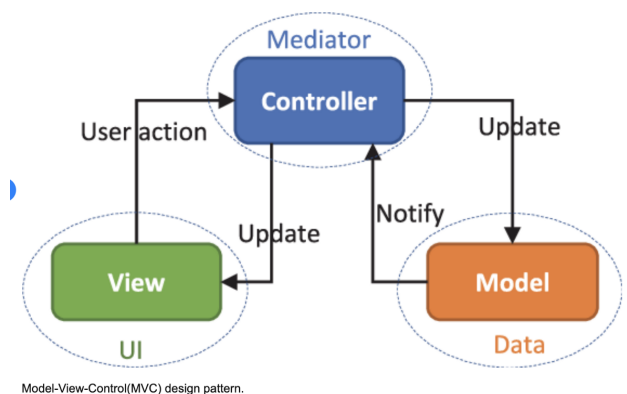
5. Descripció tecnològica

5.1. Concepció global de l'arquitectura

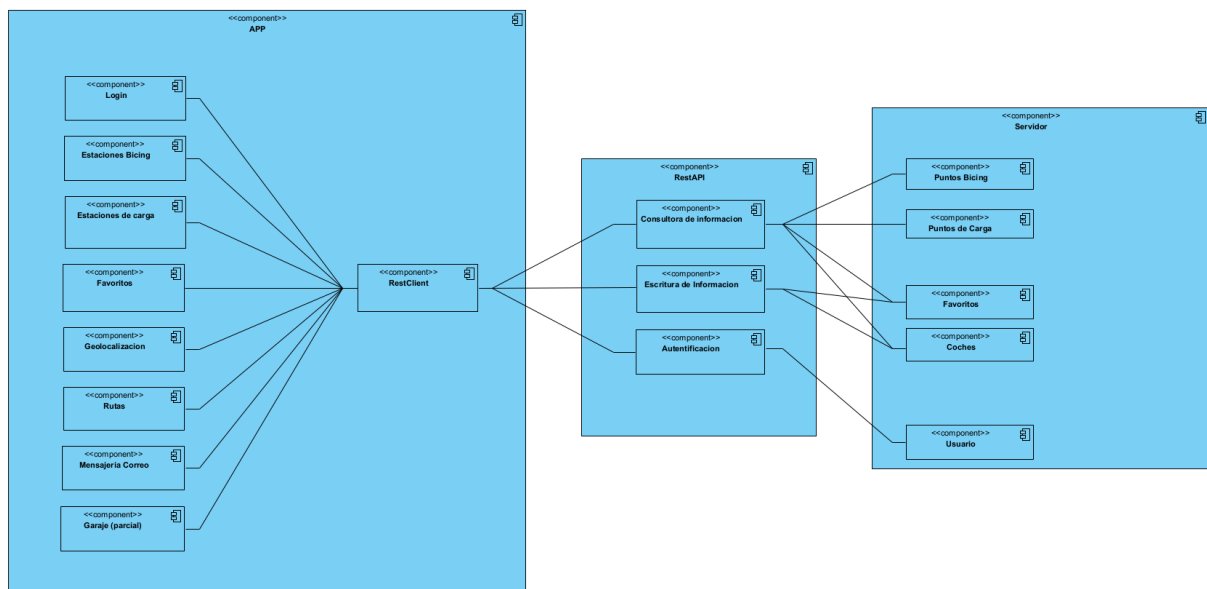
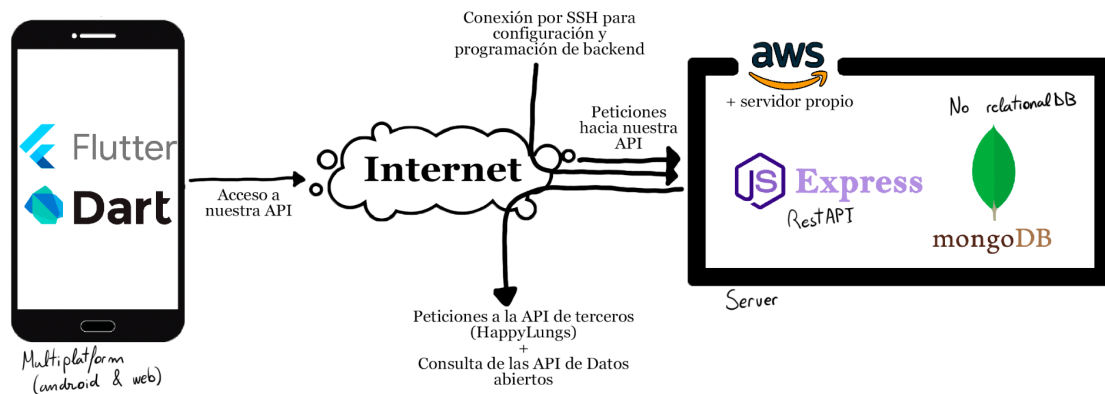
Quant a l'arquitectura del nostre programari podem destacar dos: MVC i disseny en 3 capes, on tots dos es complementen entre ells.

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.



5.2. Altres diagrames d'arquitectura



5.3. Patrons de disseny aplicats

MVC

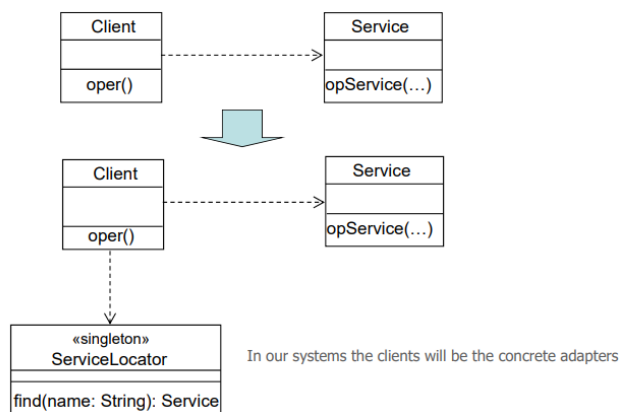
Com hem comentat anteriorment, hem dividit el sistema en 3 principals capes: Presentació, Domini i BD.

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.

```

1  import 'package:flutter_project/domini/services/google_login_adpt.dart';
2  import 'package:flutter_project/domini/services/google_maps_adpt.dart';
3  import 'package:flutter_project/domini/services/google_places_adpt.dart';
4  import 'package:get_it/get_it.dart';
5
6  final serviceLocator = GetIt.instance;
7
8  void setUpLocator() {
9    serviceLocator.registerLazySingleton<GoogleMapsAdpt>(() => GoogleMapsAdpt());
10   serviceLocator.registerLazySingleton<GoogleLoginAdpt>(() => GoogleLoginAdpt());
11   serviceLocator.registerLazySingleton<GooglePlaceAdpt>(() => GooglePlaceAdpt());
12 }
13
14
15 GoogleMapsAdpt get getMapsService {
16   return serviceLocator<GoogleMapsAdpt>();
17 }
18
19 GoogleLoginAdpt get getLoginService {
20   return serviceLocator<GoogleLoginAdpt>();
21 }
22
23 GooglePlaceAdpt get getPlaceService {
24   return serviceLocator<GooglePlaceAdpt>();
25 }
26

```

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 volguéssim 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 instància.

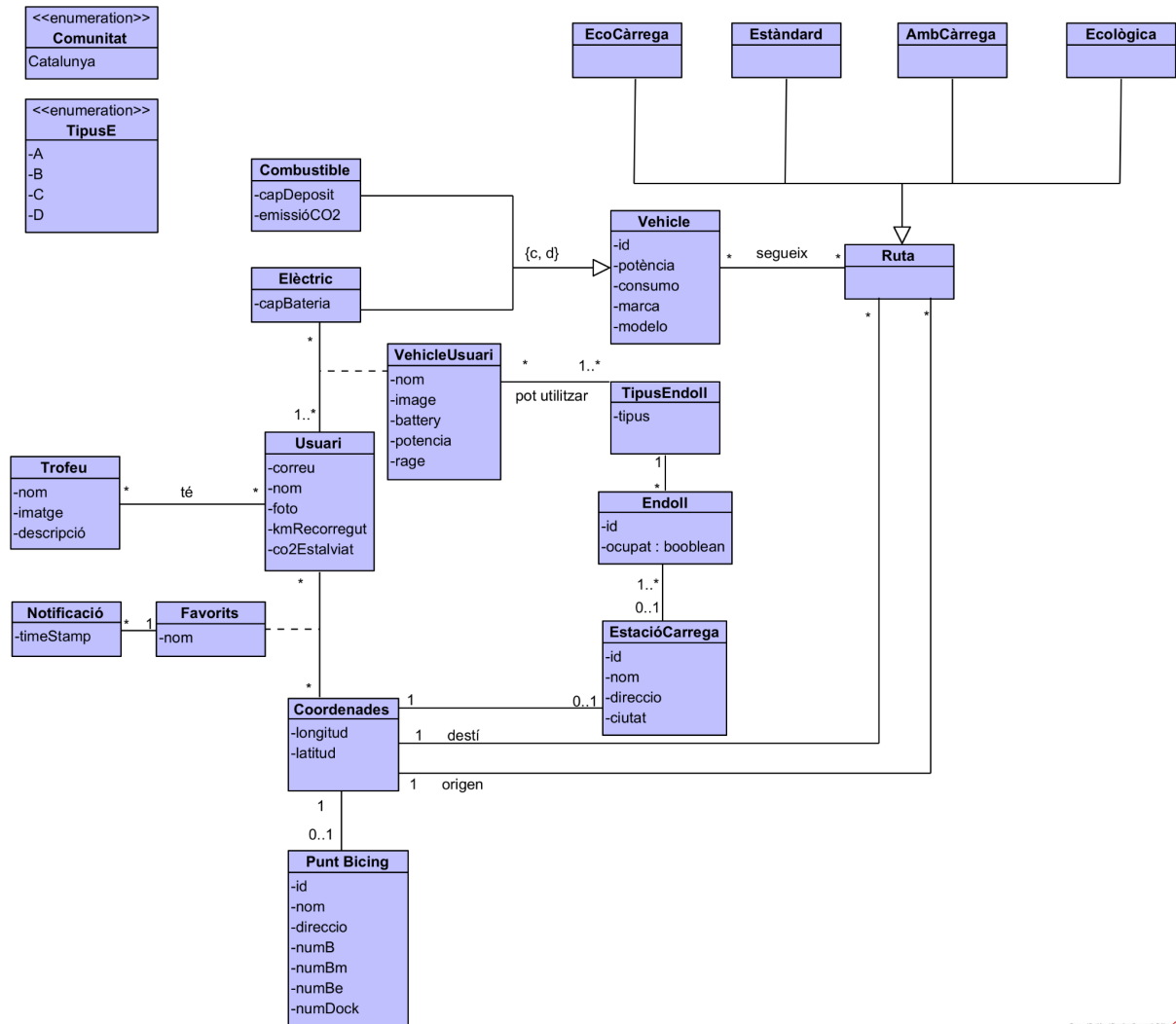
```

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 a inception 2 ha sigut que hem afegit alguns atributs necessaris a VehicleUsuari i Punt Bicing i hem tret la classe Mapa.



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

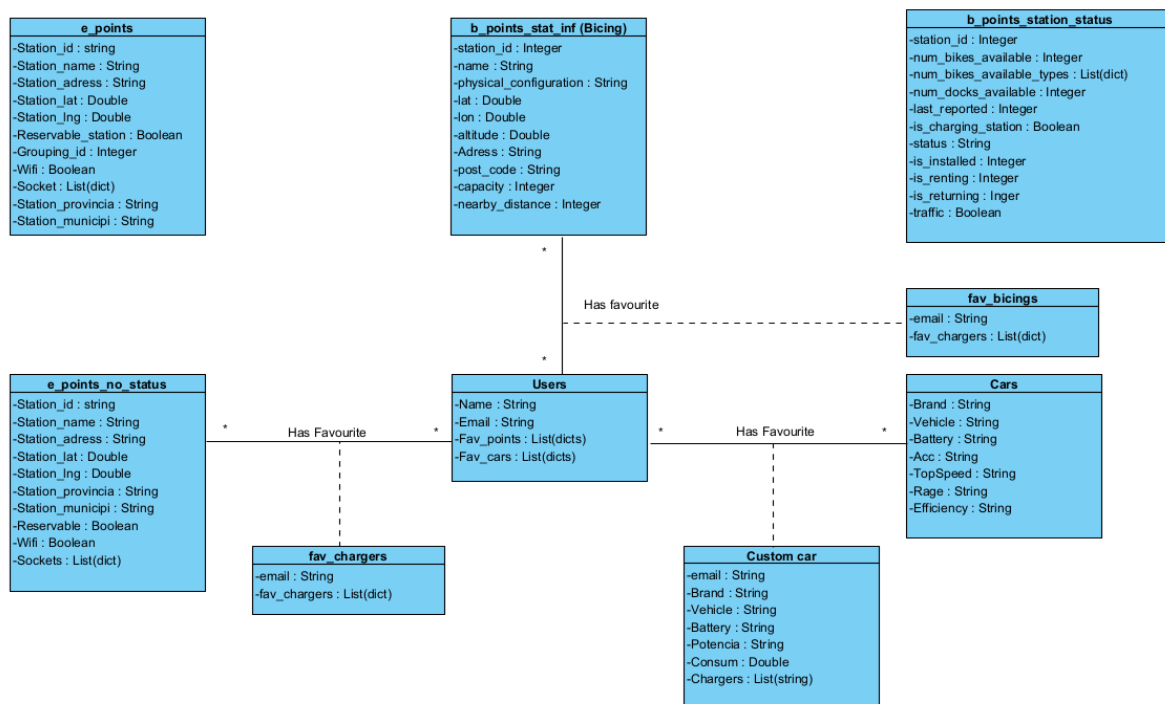
- Una ruta no pot tenir el mateix punt d'origen que de destí.

- 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 Carrega 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:



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)

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 possibilitat 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.

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 postprocessades directament al servidor.

Adicionalment, per evitar fer ús dels minuts de Actions que ofereix Github (workflows per realitzar CI/CD) ja que son bastant limitats. Vam traspasar tota la carga de treball al nostre servidor fent ús de Github com a intermediari. Això va ser possible 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

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.

APIs

API pròpia

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.

A més, el nostre servei a oferir serà: proporcionar els punts de recàrrega (i possiblement punts d'estacionament Bicing) més propers donada una localització concreta. Per tant, per temes de seguretat, ja que amb la API actual també es poden consultar les dades dels usuaris, desenvoluparem un API que només donara aquesta informació.

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

Després d'haver parlat amb l'equip de Build Green vam decidir que li donarem els punts de càrrega i, opcionalment, els de Bicing més propers a una coordenada concreta.

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](#)

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 tant, per a les funcions més importants com poden ser les de càlculs matemàtics, o semblants, hi haurà tests unitaris, entre altres proves automàtiques “necessàries”. Per aconseguir tot això, farem servir les eines natives de flutter; flutter analyze i flutter test. 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				
Q 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				
Q 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	-	Si
vol-04d66c5170b8274d2	/dev/sdb	15	Asociado	Sat Mar 12 2022 17:01:54 ...	No	-	Si

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 utilitzar 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/>

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

```
2.
3. touch /etc/apt/sources.list.d/mongodb-org-5.0.list
4.
5.     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
6.
7. sudo apt-get update
8.
9. sudo apt-get install -y mongodb-org
10.
11.
12. #Para iniciar/parar/reiniciar/estado el servicio de MongoDB
13. sudo systemctl start/stop/restart/status mongod
14.
15. #Para eliminar MongoDB
16. sudo service mongod stop
17.
18. sudo apt-get purge mongodb-org*
19.
20. #En caso de path default
21. sudo rm -r /var/log/mongodb
22. sudo rm -r /var/lib/mongodb
23.
```

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:

```

1. #En el directorio destino
2. mkdir database #nombre opcional
3. sudo systemctl stop mongod
4.
5. #en caso de querer guardar las tablas actuales
6. mv /etc/lib/mongo/* ./database/
7.
8. sudo chown mongod:mongod -R ./database
9.
10.
11. #hay que cambiar la configuración de mongoDB
12. sudo nano /etc/mongod.conf
13.
14. #En el apartado "dbpath" hay que cambiarlo hacia el anteriormente creado
15. #p.e. ./database

```

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

```

1. mongosh
2.
3. use admin
4.
5. db.createUser({user: "admin", pwd: "admin", roles:["root"]})
6.

```

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

```

1. #Volvemos a entrar en mongod.conf
2. Sudo nano /etc/mongod.conf
3.

```

4. #En el apartado bindIp deberemos agregar la DNS ipv4 publica del AWS justo después de añadir una coma
5. #Ademas, para mas seguridad en el sistema, podemos activar que para cualquier acceso se requiera de credencial, por ejemplo, el admin creado anteriormente.
- 6.
7. #Descomentamos la línea “#security:” y en la siguiente línea agregamos “authorization: enabled”
- 8.

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:

```
1. Mongosh -u admin -p admin
```

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

```
1. 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.

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>