Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Creació d'una aplicació de control de revisions de motocicletes.

Document: Memòria

Alumne: Lluc Pagès Pérez

Tutor: Dr. MARTIN CAMPOS, IGNACIO CLEMENTE
Departament: INFORMÀTICA, MATEMÀTICA APLICADA I ESTADÍSTICA
Àrea: LLENGUATGES I SISTEMES INFORMÀTICS

Convocatòria (mes/any): Juny/2021

1. Introducció i objectius

La tecnologia avança cada vegada més ràpid, i gran part del sector té com a objectiu facilitar la vida a les persones. Un dels invents tecnològics més presents en la nostra vida diària, són els telèfons intel·ligents (o smartphones).

De la mateixa manera, la tecnologia en el món del motociclisme augmenta exponencialment. Des d'invents que milloren la seguretat del conductor (p. ex. Control de tracció, Anti-lock Braking System (ABS)), fins ajudes per augmentar la velocitat de conducció i temps de resposta (p. ex. accelerador electrònic, aerodinàmica del vehicle).

1.1. Problemàtica

És conegut que el manteniment dels motors de combustió és extens i cal tenir en compte molts paràmetres, a més a més, la legalitat vigent ens obliga a passar un seguit de revisions. Podem trobar aquesta informació a diferents llocs:

- **Manual d'usuari:** Inclòs en el vehicle al moment de la compra. Destinat a usuaris sense coneixement elevat però generalment no proporciona una informació completa del funcionament del vehicle.
- **Manual de taller:** Destinat a tallers de reparació. És necessari uns coneixements avançats. Podem trobar-hi tota la informació necessària, com pressions dels cargols, quadre elèctric, etc.

Aquests manuals ens proporcionaran la informació necessària, però porta temps, uns coneixements mínims i generalment no els tenim a disposició. També, comporta consultar els manuals periòdicament.

Això implica unes possibilitats molt altes d'error. Ja sigui per la possibilitat d'oblit del nombre de kilòmetres que hem recorregut o de l'última data de revisió.

Per finalitzar, avui en dia existeixen motocicletes amb indicador de revisió, però generalment són indicadors binaris ("sí" o "no").

1.2. Objectius

Per aquests motius, el propòsit d'aquest projecte és realitzar una aplicació per a telèfons Android i iOS, on controlar diferents paràmetres de la motocicleta. Aquesta proporcionarà comoditat a l'hora de comprovar l'estat del vehicle i notificarà en cas d'algun problema.

Els objectius inicials de l'aplicació són:

- Permetre crear múltiples usuaris.
- Afegir una motocicleta indicant marca i model
- Afegir una motocicleta “lliure” (Sense paràmetres predefinits).
- Configurar la mitjana de kilòmetres per any recorreguts
- Actualitzar el nombre de kilòmetres recorreguts totals del vehicle en qüestió
- Modificar les notificacions que es desitja rebre
- Modificar la informació i estat de cada revisió
- Notificar la necessitat de revisió

A més, un dels objectius principals del projecte és l’aprenentatge de tecnologies. Suposarà gran part del temps invertit, i es dividirà en dues parts: **aprenentatge inicial i aprenentatge continuat.**

Aprenentatge inicial

L'aprenentatge inicial és necessari abans d'iniciar el desenvolupament de l'aplicació, però després d'haver escollit les tecnologies utilitzades.

Suposa una inversió de temps dedicada exclusivament a l’aprenentatge, on es creen els conceptes inicials i fonaments amb els quals s’haurà de treballar posteriorment.

Aprenentatge continuat

L’aprenentatge continuat succeeix al llarg del desenvolupament de l’aplicació. Tot i ja tenir els fonaments per a l’ús de les tecnologies, l’habilitat i confort amb cada una d’elles augmenta i es treballa dia a dia al llarg de tot el projecte.

També suposa una inversió de temps, ja que el desenvolupament inicialment és més lent.

2. Estudi de viabilitat

Per desenvolupar aquest projecte són necessaris tres dispositius electrònics:

- Ordinador utilitzat per desenvolupar el projecte. Ha de tenir uns requisits mínims per ser capaç de programar i testejar l'aplicació.
- Dispositiu Android: Pot tractar-se d'un dispositiu real o un dispositiu virtual (a través d'emuladors).
- Dispositiu iOS: Pot tractar-se d'un dispositiu real o virtual. Si l'ordinador utilitzat no és MacOS, necessitarem estrictament un dispositiu real.

Ja disposo d'aquests dispositius, no suposarà cap despesa.

A l'hora de la implantació seria necessari un servidor accessible des d'internet. Això ho podrem realitzar a través d'un servei de "Cloud Computing" (p. ex. Google Cloud, Amazon Web Services, Microsoft Azure, etc).

Aquests serveis tenen un preu molt variable, però l'objectiu inicial de l'aplicació no necessita molts recursos. Mentre no superem el màxim permès a cada plataforma utilitzada, no hi haurà despeses.

També hi ha un temps estimat d'hores de desenvolupament que s'ha de contemplar. Per a poder fer un càlcul ens podem ajudar de l'eina "Cuánto Vale Mi Hora Como Freelance".

Amb una jornada de 6 hores al dia durant 5 dies a la setmana, i amb uns paràmetres aproximats de cara al sou, dies lliures a l'any... Ens dona que la tarifa mínima són 18.13 € / hora (*Imatge 1*).

Imatge 1. Càlcul tarifa mínima.

En total, en el projecte he invertit unes 270 hores. Però actualment, sense la necessitat de l'aprenentatge inicial i amb més agilitat a l'hora de programar, necessitaria aproximadament unes 210 hores. Així doncs ja podem calcular quant ens costaria:

- 210 hores * 18,13 € / hora = 3807,3€

Ens costaria un aproximat de **3800 €.**

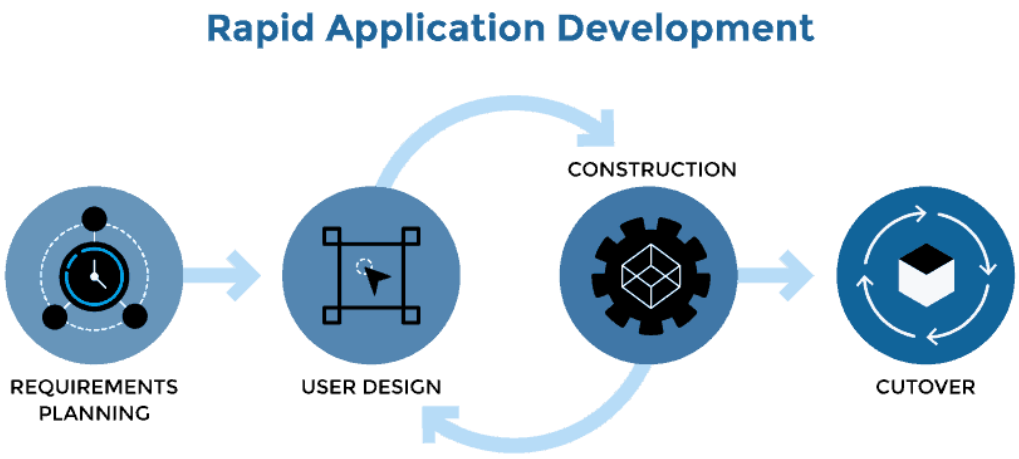
3. Metodologia

Hi ha molts tipus de metodologia en el desenvolupament de software. Durant el curs hem treballat i hem estudiat metodologies com SCRUM ("Scrum | Digite").

Donades les característiques d'aquest projecte: una única persona té totes les funcions i càrrecs, el temps de desenvolupament és curt. La metodologia a seguir serà una adaptació de l'anomenada Rapid Application Development (RAD) (Kissflow, "6 Essential Questions to Understand Rapid Application Development Methodology").

3.1. Rapid Application Development (RAD)

Va ser inventada per *James Martin* el 1991. És una metodologia enfocada al desenvolupament ràpid de software. És una forma de desenvolupament àgil i prioritza iteracions ràpides. (Singh)



Imatge 2. RAD (Simmons)

Beneficis

- Flexibilitat i adaptabilitat de codi. Es poden fer canvis ràpidament durant el procés de desenvolupament.
- Iteracions ràpides que acceleren el desenvolupament.
- Elevada comunicació entre els desenvolupadors, clients i usuaris.
- Facilitat d'arreglar problemes durant la producció.
- Menys possibilitat de sorpreses, per les integracions periòdiques.

Inconvenients

- Més complexa que altres metodologies.
- Només és viable en equips petits.
- Necessitat gran col·laboració en equip.
- El projecte s'ha de poder dividir en mòduls.
- Són necessaris desenvolupadors qualificats.
- És necessari actualitzar requeriments d'usuaris durant el cicle de producció.

Hi ha petites variacions d'aquesta metodologia, les quals divideixen el procés en 4 o 5 passos. En aquest cas els passos a seguir han estat els següents:

1. **Definir els requeriments del projecte:** Es defineixen objectius, expectatives, dates límit, mòduls...
2. **Desenvolupar:** Desenvolupar un dels mòduls i/o modificar algun dels ja realitzats.
3. **Recopilar comentaris d'usuaris:** En aquest pas m'he ajudat de la meva família i amics propers per recopilar comentaris i millorar cada un dels apartats.
4. **Test**: Testejar l'aplicació en cerca de problemes, errors i altres.
5. **Implantació:** Publicació de l'aplicació i API corresponent.

Els passos 2 i 3 es repeteixen contínuament fins a arribar al producte final.

4. Planificació

4.1. Divisió per mòduls

Com s'ha prèviament mencionat, per poder seguir la metodologia "RAD" és necessari dividir el projecte en diferents mòduls.

Autenticació d'usuari

Aquest apartat conté tot el relacionat amb l'autenticació de l'usuari. L'objectiu inicial que li pertoca és: **"**Permetre crear múltiples usuaris".

Llistat de motocicletes

En aquest mòdul s'ha de poder visualitzar totes les motocicletes que té un usuari.

Afegir nova motocicleta

Els objectius inicials vinculats a aquest mòdul són:

- Afegir una motocicleta indicant marca i model
- Afegir una motocicleta "lliure" (Sense paràmetres predefinits).

Visualitzar les revisions d'una motocicleta ja existent

Aquest mòdul s'encarrega de poder visualitzar els detalls de la motocicleta i, per tant, les revisions de cada una d'elles.

Modificar/Actualitzar la informació d'una motocicleta

Poder modificar les dades és necessari crear la funcionalitat, d'això se n'encarregarà aquest mòdul. Haurà de complir els objectius inicials següents:

- Configurar la mitjana de kilòmetres per any recorreguts.
- Actualitzar el nombre de kilòmetres recorreguts totals del vehicle en qüestió.
- Modificar la informació i estat de cada revisió.

Administrar informació d'usuari autenticat

Aquest mòdul no té cap objectiu inicial associat. Però és necessari en la majoria d'aplicacions que funcionen a base d'usuaris.

Notificacions

Aquest mòdul ha de permetre gestionar les notificacions (activar / desactivar), tant per dispositiu com per revisió. Els objectius inicials relacionats són:

- Modificar les notificacions que es desitja rebre
- Notificar la necessitat de revisió

4.2. Pla de treball

La idea del projecte va sorgir durant el curs acadèmic 2019-2020. Durant el qual, com a usuari d'una motocicleta, em vaig adonar de la quantitat de revisions que són necessàries i de la dificultat de recordar cada una d'elles.

Posteriorment, també m'he trobat amb altres problemes que s'haurien pogut evitar en cas de portar les revisions al dia. Concretament el dia 1 de maig del 2021, es va trencar una vàlvula d'admissió del motor, el qual s'hagués pogut evitar en cas de fer el balanç de vàlvules corresponent.

Amb tot això, vaig començar el projecte el 15 de març de 2021, al rebre el full de TFG aprovat. Aquest fet donava un termini aproximat de 11 setmanes pel desenvolupament.

Pla inicial

En el pla inicial vaig ser conscient dels possibles contratemps que podrien sorgir, i de 11 setmanes disponibles, vaig realitzar un esquema de 9.

- 1ª setmana: Requeriments del projecte i aprenentatge inicial.
- 2ª – 8ª setmana: Desenvolupament de l'aplicació (1 mòdul per setmana).
- 9ª setmana: Testing i modificacions finals.

Durant el projecte planejava treballar de dilluns a divendres de 8 h a 14 h, com ja estava acostumat de les Estades en Entorn Laboral (EEL). I vaig dividir cada setmana en el següent esquema:

- Dilluns a dijous: Desenvolupament de mòdul.
- Divendres
    - (8:00 – 10:00) Feedback d'usuaris (un sol usuari en aquest cas)
    - (10:00 – 14:00) Modificacions finals de mòdul

L'ordre de desenvolupament dels mòduls seguia l'ordre descrit en el punt anterior (*4.1. Divisió per mòduls*).

Pla final

Com era de suposar, van sorgir imprevistos:

- Les dues primeres setmanes vaig realitzar la meitat de les hores a causa de falta de temps. Em vaig trobar amb la necessitat de preparar-me per un examen d'anglès.

- Vaig necessitar aproximadament una setmana més per aprendre a el funcionaments bàsics d'algunes de les tecnologies i conceptes utilitzats.

- El mòdul de notificacions i el mòdul d'administració d'usuaris els vaig poder agrupar en una sola setmana.

Finalment ens trobem amb un pla de treball de 10 setmanes (9 de completes):

- 1ª i 2ª setmana: Requeriments del projecte.
- 3ª setmana: Aprenentatge inicial.
- 4ª – 9ª setmana: Desenvolupament de l'aplicació.
- 10ª setmana: Testing, modificacions finals i Deployment.

En el repartiment d'hores de la setmana durant el desenvolupament, generalment tot va seguir el previst.

Per acabar l'ordre de desenvolupament dels mòduls va ser el següent:

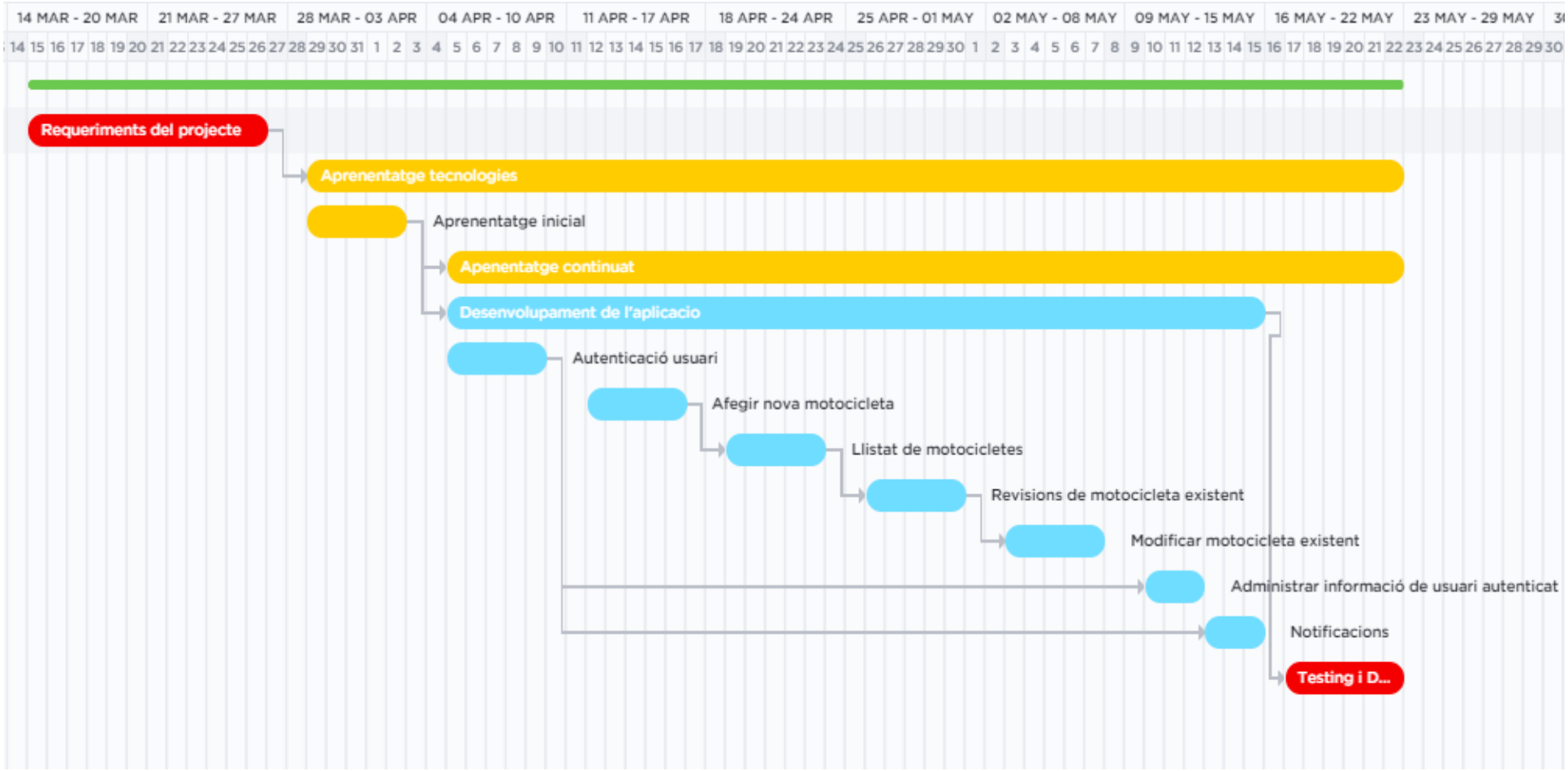
1. Autenticació d'usuari.
2. Afegir nova motocicleta.
3. Llistat de motocicletes.
4. Visualitzar les revisions de motocicleta existent.
5. Modificar / Actualitzar la informació d'una motocicleta.
6. Administrar informació d'usuari autenticat i Notificacions.

Hores totals

Treballava de dilluns a divendres 6 hores, de 8h a 14h, durant un total de nou setmanes completes.

Això ens dona un total de **270 hores**.

Diagrama de Gantt



Imatge 3. Diagrama de Gantt.

5. Marc de treball i conceptes previs

5.1. Conceptes previs

Per entendre correctament el projecte cal conèixer el significat d'algunes paraules en l'àmbit utilitzat:

- Llenguatge de programació: Llenguatge informàtic utilitzat pel desenvolupament d'una aplicació o software.

- Software: Conjunt de programes i rutines que permeten a un dispositiu executar determinades tasques. ("Oxford Languages")

- Framework: És un conjunt de conceptes, pràctiques i criteris estandarditzats per enfocar un tipus de problemàtica concreta. (colaboradores de Wikipedia, "Framework")

- Front-end: Denota la part d'un sistema informàtic o aplicació amb la qual els usuaris interactuen directament. ("Oxford Languages")

- Back-end: Denota la part d'un sistema informàtic o aplicació la qual no és accedida directament per l'usuari, normalment responsable de l'emmagatzematge i manipulació de dades.

- API (Application Programming Interface): És un software intermediari que permet que dues aplicacions es comuniquin entre si. ("What Is an API? (Application Programming Interface)")

- Push notifications: Són missatges curts que apareixen en un dispositiu mòbil. ("What is a push notification | buildfire")

- Collection: A una base de dades MongoDB, una "collection" és una agrupació de documents.

Cal també conèixer el funcionament i objectius de les revisions d'una motocicleta, o en general de qualsevol vehicle, així com el tipus d'aplicació que es desenvoluparà (Client-Servidor) i què són les aplicacions híbrides.

L'objectiu principal d'una **revisió** és comprovar el correcte funcionament de cada una de les parts del vehicle. En moltes ocasions s'ha de realitzar canvis de líquids, filtres o peces més importants. Altres vegades només és necessari ajustar algunes peces (p. ex. la tensió de la cadena). I en alguns casos només s'ha de revisar que tot estigui en ordre (p. ex. desgast dels pneumàtics).

D'altra banda, l'aplicació desenvolupada és de tipus **Client-Servidor** (Wikipedia contributors, "Client–Server Model"). Aquestes es divideixen en dues parts: el Front-end, situat al telèfon de l'usuari, i el Back-end, servidor extern on s'emmagatzemen les dades. És necessària connexió a través de la xarxa pel seu ús.

Per a la comunicació entre les dues parts, s'utilitzarà una API REST ("What Is a REST API?"), la qual determina una sèrie de normes i protocols per a la correcta interacció entre Client i Servidor.

Finalment, les **aplicacions híbrides** (Sierra, "Aplicaciones híbridas") són aquelles capaces de funcionar en diferents sistemes operatius. Així, la mateixa aplicació pot utilitzar-se en qualsevol dispositiu, sense importar el Sistema Operatiu.

5.2. Equip de treball

El projecte ha estat desenvolupat per una sola persona, Lluc Pagès Pérez, la qual té el paper de desenvolupador i líder del projecte.

El paper d'usuari va ser realitzat pel meu germà, Guillem Pagès Pérez, donant un feedback a la finalització de cada mòdul.

El tutor de pràctiques actua com a ajuda i suport tècnic.

6. Requisits del sistema

Els requisits del sistema es divideixen en **requisits funcionals** (comportament del software, activitats i serveis del sistema, etc) i **requisits no funcionals** (restriccions del client o sistema, temps d'entrega, documentació, etc).

6.1. Requisits funcionals

El software ha de permetre realitzar el següent:

- Autenticar usuaris.
- Crear nous usuaris.
- Recuperar la contrasenya en cas d'oblit.
- Administrar usuaris:
    - Modificar informació personal d'usuari.
    - Modificar dades d'accés d'usuari.
- Administrar motocicletes per a l'usuari identificat:
    - Afegir motocicleta amb paràmetres predefinits.
    - Afegir motocicleta sense paràmetres predefinits.
    - Eliminar motocicleta.
    - Administrar revisions de motocicleta:
        - Afegir revisions.
        - Eliminar revisions.
        - Modificar informació / estat de les revisions.
- Rebre notificacions.
- Administrar les notificacions que es desitja rebre:
    - Decidir si es vol rebre notificacions en el dispositiu des del qual l'usuari s'ha identificat. Cada dispositiu controla les notificacions del mateix dispositiu.
    - Decidir si es vol rebre les notificacions per cada una de les revisions d'una motocicleta.

6.2. Requisits no funcionals

De rendiment

- Els usuaris han de poder autenticar-se amb un nombre indefinit de dispositius al mateix temps.
- L'actualització de les dades es realitzarà cada dia a les 00 h (UTC +2).
- El temps de resposta ha de ser òptim.

De disseny

- Ha de funcionar en dispositius amb Android Q (versió 10.0) o superior.
- Serà necessari disposar de connexió internet per l'ús de l'aplicació.
- Les contrasenyes s'han de guardar en forma de "hash", encriptades.

Sobre interfícies externes

- La navegació principal serà a base d'un menú desplegable lateralment.
- Sense barres de navegació inferiors.
- A l'hora d'eliminar qualsevol motocicleta es demanarà confirmació.

Objectius de disseny

- L'estat d'una revisió de motocicleta ha de ser fàcil i ràpid de visualitzar i modificar.

6.3. Actors

Existiran dos tipus d'actors que interactuaran amb l'aplicació: **usuari identificat** i **usuari no identificat**.

Usuari no identificat

L'usuari no identificat ens el trobem només obrir l'aplicació. Només pot realitzar accions relacionades amb autenticació i creació d'usuaris.

Usuari identificat

L'usuari identificat té més privilegis, tindrà accés a pràcticament totes les accions disponibles, excepte les que pot realitzar l'usuari no identificat (p. ex. autenticar-se).

7. Estudis i decisions

Abans d'iniciar el projecte és necessari realitzar una sèrie de decisions. Haurem d'escollir les tecnologies utilitzades, així com la paleta de colors i logotip de l'aplicació. Podem dividir el projecte en dues parts principals: **Front-end** i **Back-end**.

7.1. Eines a utilitzar | FRONT-END

Per al desenvolupament d'aplicacions híbrides s'utilitzen llenguatges de programació i frameworks diferents. Les solucions més utilitzades són: **Flutter, Xamarin i React Native.**

Imatge 4. Flutter logo (Maldini)

Flutter

És un kit de desenvolupament de software multiplataforma i de codi obert creat per **Google**. S'escriu en llenguatge Dart ("Dart Overview"). L'arquitectura es basa en programació reactiva. Un dels seus punts a favor més importants és la rapidesa d'escriptura de codi i el seu rendiment.
("Flutter - Beautiful Native Apps in Record Time")

Imatge 5. Xamarin logo (Rodríguez, Blog)

Xamarin

Xamarin és una aplicació de codi obert desenvolupada per **Microsoft** per a la creació aplicacions d'alt rendiment per Android i iOS. Està escrita en llenguatge C# i la plataforma .NET.
(Microsoft)

Imatge 6. React logo (Wikipedia contributors, "React Native")

React Native (JavaScript)

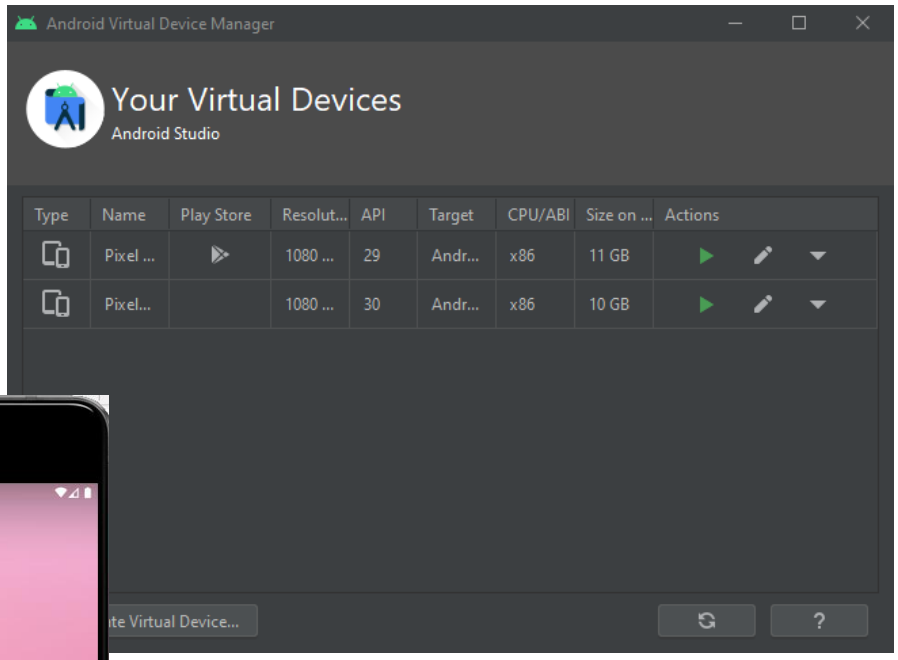
És un framework de JavaScript creat per **Facebook**. Està destinat al desenvolupament d'aplicacions per Android, iOS, Web, etc.
El lema sota el qual l'anuncien és: "Learn once, write anywhere".
("React Native")

Elecció

L'elecció final va ser **React Native**. Té un gran suport amb noves actualitzacions constants, així com moltes eines disponibles.
A més, és una eina molt utilitzada avui en dia, i l'aprenentatge de la mateixa proporciona moltes oportunitats laborals.

Testing

A l'hora de provar l'aplicació, utilitzarem l'aplicació **Android Studio** ("Meet Android Studio") . Aquesta eina té moltes funcions, però en aquest cas només l'utilitzarem per crear **emuladors** o **dispositius virtuals** de diferents versions d'Android.



Imatge 7. Android Studio Virtual Devices.

Imatge 8. Android Studio, Pixel 4.

7.2. Eines a utilitzar | FRONT-END | REACT NATIVE

Un cop vaig escollir el llenguatge de programació i framework amb el que desenvolupar el "front-end" del projecte, hi ha un seguit d'opcions per escollir:

Expo CLI v.s. React Native CLI

Pel desenvolupament d'aplicacions amb React Native, existeixen diferents eines a escollir. Els dos frameworks més utilitzats en aquest sentit són **Expo** i **React Native CLI**. Aquests són els avantatges i els inconvenients més grans a l'hora d'escollir:

Imatge 9. Expo CLI logo ("Expo, Logo Free Icon")

Expo CLI

Avantatges:

- La facilitat de crear un projecte.
- Una gran varietat d'eines i instruments.
- Facilitat de provar l'aplicació durant el desenvolupament.

Inconvenients:

- No es poden afegir mòduls natius.
- No es poden utilitzar llibreries en altres llenguatges.
- L'aplicació inicial ja és molt pesada.

Imatge 10. React logo (Wikipedia contributors, "React Native")

React Native CLI

Avantatges:

- Es poden afegir mòduls en codi natiu (Java / Objective-C).
- El desenvolupador té 100% control dels arxius.

Inconvenients:

- És necessari tenir instal·lat Android Studio i/o XCode.
- Impossible crear aplicacions iOS sense dispositiu mac.
- Per compartir l'app s'ha d'enviar l'arxiu sencer.

Per aquest projecte vaig escollir **Expo CLI** per la facilitat de crear el projecte, agilitzar el desenvolupament del mateix i per la falta d'un dispositiu macOS, el qual era necessari en cas d'escollir React Native CLI.

Push Notifications - Expo v.s. Firebase

Ens trobem amb dues opcions de cara a les "push notifications" del telèfon.

En entorns de producció, generalment és recomanable utilitzar el servei de "Firebase" ("Firebase Cloud Messaging"). Però en aquest cas, i al tractar-se d'un entorn de desenvolupament, utilitzarem **"Expo Push Notifications"** ("Push Notifications Overview") per la gran facilitat de configuració.

7.3. Eines a utilitzar | BACK-END

Pel funcionament de l'aplicació, és necessària una base de dades i una API. I el primer que s'ha d'escollir és la base de dades. Basant-se en aquesta elecció, podrem escollir amb quines tecnologies desenvoluparem l'API.

Bases de dades | SQL vs NoSQL

Avui en dia hi ha dos tipus de base de dades predominants: Base de dades relacional (SQL) o no relacional (NoSQL).

SQL

Són bases de dades que segueixen el model relacional. És una de les estructures més utilitzades avui en dia. El llenguatge utilitzat és l'SQL, però hi pot haver petites variacions. Alguns exemples són:

- MySQL ("MySQL")
- Oracle SQL ("Oracle SQL")
- Microsoft Access ("Microsoft Access")

NoSQL

No segueixen el model relacional. Aquest tipus de bases de dades han existit des dels anys 1960, tot i que el nom "NoSQL" és molt recent (segle 21). Són més escalables i ofereixen un rendiment superior en casos concrets. Un dels avantatges més importants és la flexibilitat que ofereixen.

Exemples de bases de dades NoSQL són:

- MongoDB (colaboradores de Wikipedia, "MongoDB")
- Cassandra ("Apache Cassandra")
- CouchDB ("Apache CouchDB")

Elecció

Tot i la gran popularitat i el gran suport que ofereixen algunes bases de dades SQL, per aquest projecte vaig escollir la utilització de **MongoDB**, una base de dades no relacional (NoSQL).

La raó fonamental darrera d'aquesta elecció és l'ús objectiu de l'aplicació. Al ser una aplicació enfocada a cada usuari de manera individual, MongoDB ens permet la possibilitat de filtrar i reduir molt les dades amb les quals l'API ha de tractar a cada petició. Això millorarà el temps de resposta i l'escalabilitat del software desenvolupat.

API

A l'haver escollit MongoDB, aquesta proporciona la seva pròpia llibreria per a la programació i el desenvolupament de l'API, s'anomena **Mongoose**. El llenguatge de programació és JavaScript.

Testing

A l'hora de realitzar el testing de l'API i la base de dades, utilitzarem una aplicació molt coneguda i prèviament utilitzada a classe: **Postman**.

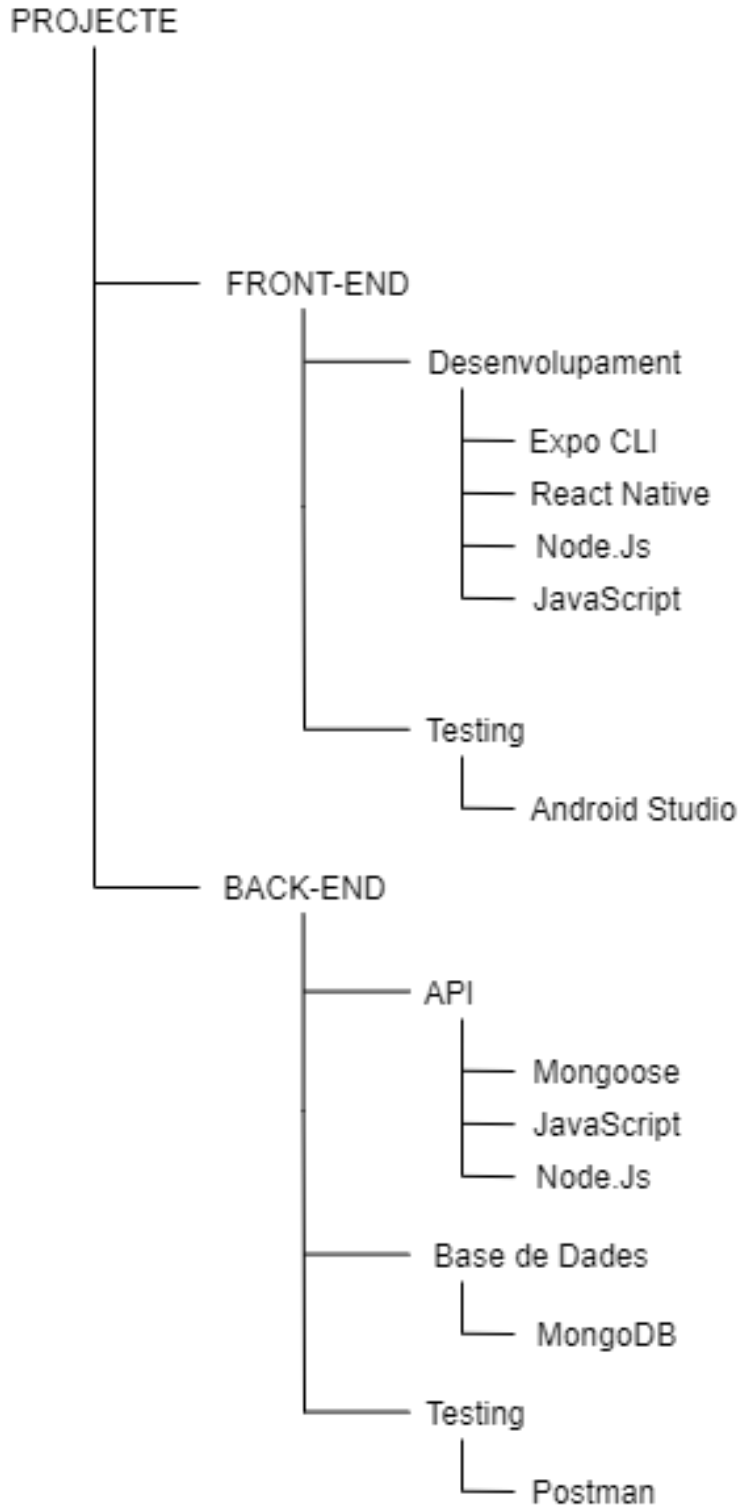
Imatge 11. Postman logo ("Postman | The Collaboration Platform for API Development")

Postman és una plataforma de col·laboració per al desenvolupament d'API. Simplifiquen els passos de creació i testing per accelerar la creació d'aquestes.

("Postman | The Collaboration Platform for API Development")

7.4. Eines a utilitzar | ESTRUCTURA FINAL

Aquest és un esquema de l'estructura final utilitzada. Cal remarcar que no estan incloses totes les llibreries, la llista de llibreries afegides i instal·lades a través de Node.Js es poden veure a l'annex (*Annex A. Dependències i llibreries Node.JS*).



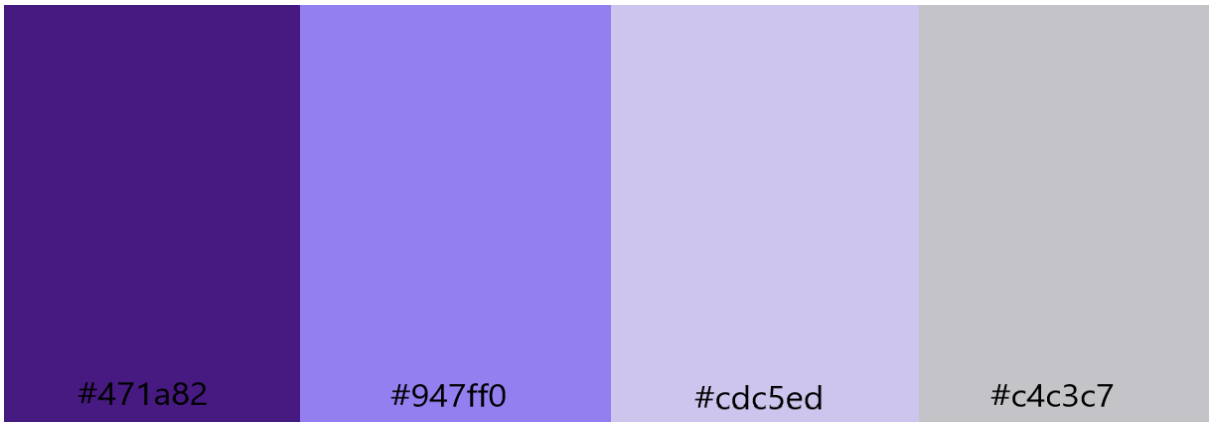
Imatge 12. Estructura final. Llibreries & Tecnologies

7.5. Aspectes visuals de l'aplicació

A l'hora de dissenyar i decidir diferents aspectes visuals de l'aplicació, vaig tenir ajuda del meu germà, Guillem Pagès Pérez, el qual ha estudiat "Màrqueting i publicitat". Em va guiar i ensenyar a utilitzar l'Adobe Illustrator per aconseguir el meu objectiu.

Paleta de colors

Per la paleta de colors vam escollir una escala de lila a gris.



Imatge 13. Paleta de colors APP

Logotip

Nom

A l'hora de dissenyar el logotip també havíem d'escollir el nom final de l'aplicació. Inicialment vaig anomenar al projecte "Bike Inspectioner" el qual resultava molt llarg per a qualsevol logotip. Finalment vam optar per retallar-lo a "**Bike Inspec**".

Tipografia

Per a la tipografia buscàvem un estil futurista, vam optar per "**Suissnord"** (no apte per ús comercial). D'altra banda, volíem remarcar les dues primeres lletres de cada paraula, per facilitar la creació d'un logotip de mida reduïda. Vam realitzar un seguit de proves:

Imatge 14. Proves tipografia

Ens vam decantar per la **primera** opció, "**Segoe Script"**.

Final

Per acabar, vam afegir decoració al nom escollit. Volíem sensació de “control” i un símbol que representes una motocicleta. Finalment el logotip va quedar de la següent manera:

Imatge 15. Logotip final APP

7.6. Idioma

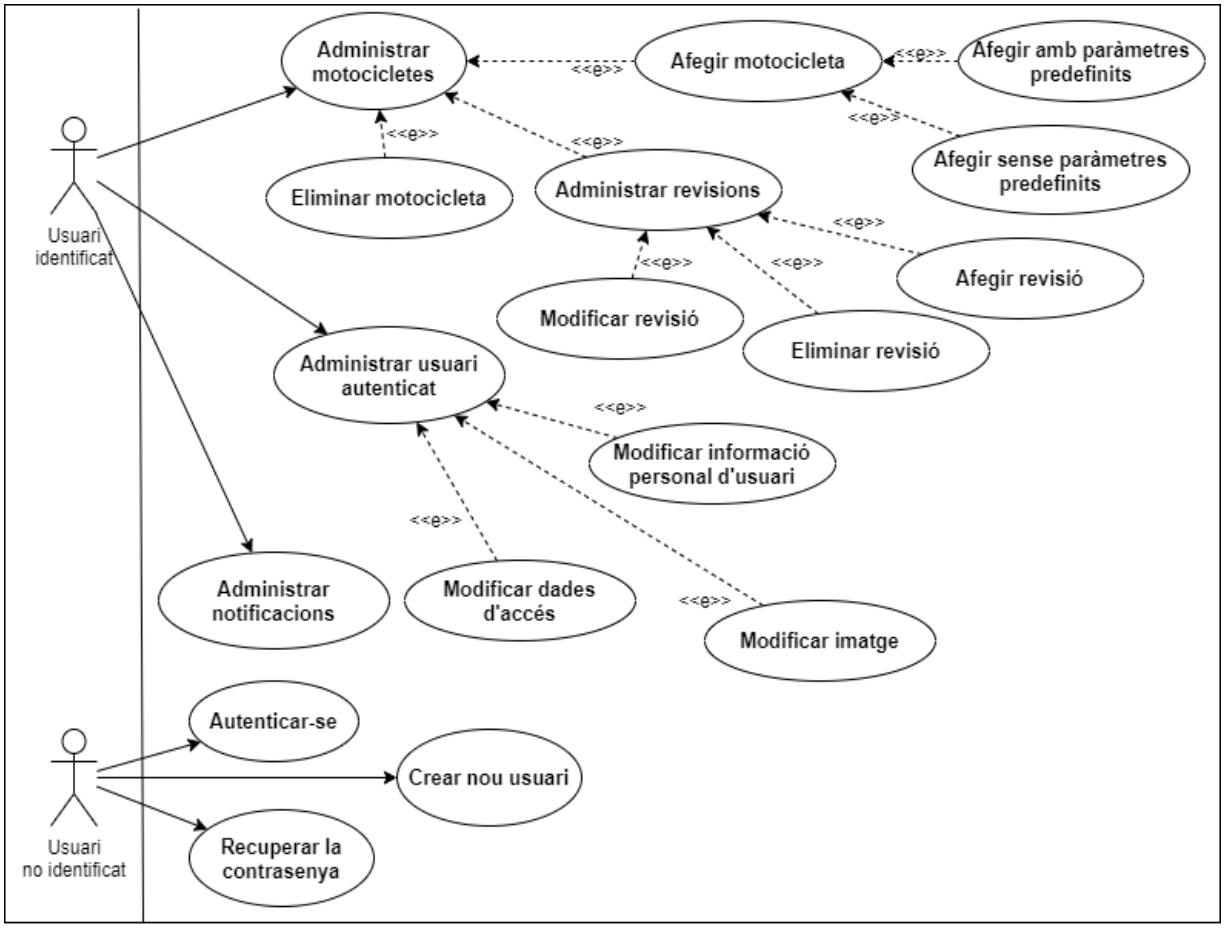
Abans de començar a programar, era necessari decidir l’idioma de l’aplicació. De cara a la possibilitat de portar a terme el projecte, l’aplicació hauria de ser plurilingüe per així arribar a més gent. Però inicialment, i per la gran quantitat d’angloparlants, l’aplicació es desenvoluparia en **Anglès**.

8. Anàlisi i disseny del sistema

Havíem de descriure des dels requeriments de l'aplicació fins a les estructures de dades utilitzades a la base de dades i l'aplicació.

8.1. Casos d'ús | Aplicació

En el diagrama de casos d'ús podem observar els actors i els requeriments associats a cada un d'ells. Aquests han estat descrits al punt 6 (*6. Requisits del sistema)*



Imatge 16. Casos d'ús

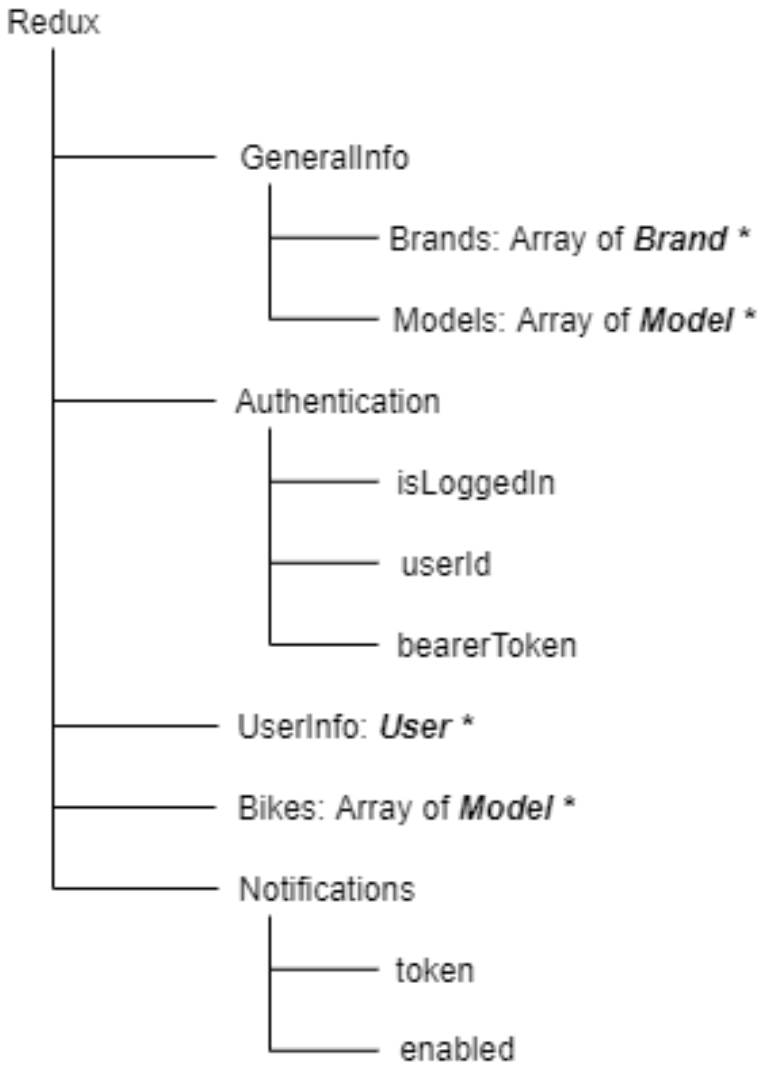
8.2. Model de dades

A l'hora de definir les dades hem de tenir en compte que la base de dades serà no relacional (NoSQL).

Front-end

Per a l'emmagatzematge de dades de l'aplicació utilitzem la llibreria "Redux" + "Async storage". Amb Redux guardarem un estat de l'aplicació no persistent, i farem ús de "Async Storage" per a la informació que necessitem conservar.

Les dades guardades i compartides a tota l'aplicació es mostren a continuació (*Imatge 17*).

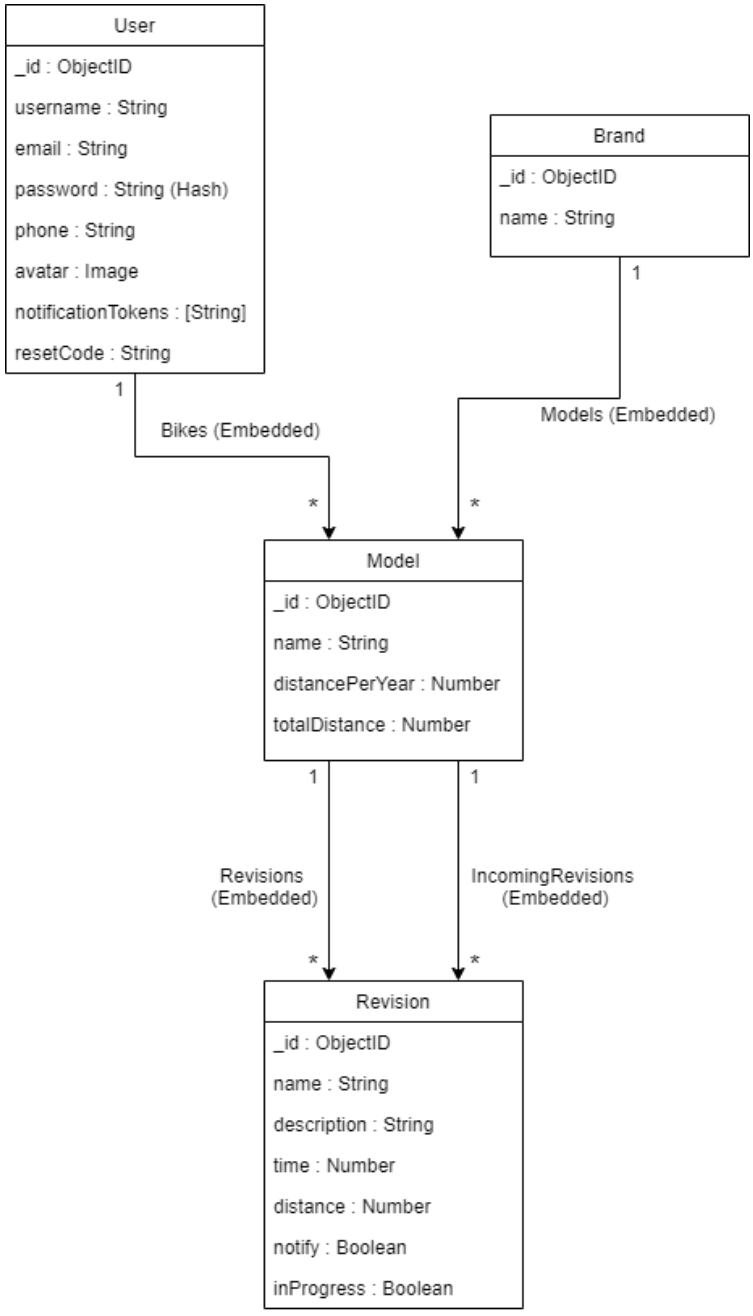


Imatge 17. Estructura dades front-end

Back-end

Per documentar i descriure una base de dades en MongoDB generalment es realitza en forma de codi (p. ex. JSON), però per facilitar la comprensió i fer-ho més visual podem veure a continuació un esquema semblant a un diagrama de classes (*Imatge 18*). Cal destacar el següent:

- Les col·leccions (collections) que ens trobem a l'arrel són 2: Usuaris (**User**) i Marques (**Brand**).
- Totes les referències *one-to-many*, no es tracten de referències (com per exemple l'id), sinó documents incorporats o interns (embedded).



Imatge 18. Estructura base de dades.

També podem veure els esquemes de cada classe en codi JavaScript:

User: {

```
    //Login details
    username: { type: String, required: true, unique: true, min: 6, max: 255 }
,
    email: { type: String, required: true, unique: true, min: 6, max: 255 },
    password: { type: String, required: true, max: 1024 },

    //Optional
    phone: { type: String, max: 15 },
    avatar: { type: String },

    //Generated
    notificationTokens: [String],
    bikes: [ModelSchema],
    resetCode: { type: String },
}
```

Brand: {

```
    name:   { type: String, required: true },
    models: [ ModelSchema ]
}
```

Model: {

```
  name: { type: String, required: true },
  distancePerYear: { type: Number, default: 0 },
  totalDistance: { type: Number, default: 0 },

  //Parameters
  incomingRevisions: [{ type: RevisionSchema, default: {} }],
  revisions: [{ type: RevisionSchema, default: {} }],
}
```

Revisio: {

```
  name: { type: String, required: true },
  description: { type: String },
  time: { type: Number },
  distance: { type: Number },
  notify: { type: Boolean, default: true },
  inProgress: { type: Boolean, default: false },
}
```

8.3. Connexions Front-end / Back-end

Per una correcta comunicació entre aplicació i l'API REST ("What Is a REST API?"), és necessari definir i estandarditzar un seguit de normes.

Prefix

En moltes APIs s'utilitzen prefixos per diferents raons, per exemple en cas de tenir múltiples versions de la mateixa API. En aquest cas **no afegirem cap prefix** a les rutes, ja que no ens és necessari.

Tipus de peticions

També hem de definir quines accions i quins tipus de peticions acceptarem. En aquest cas seran 4: **GET**, **POST**, **PATCH**, **DELETE**.

- GET: Ens servirà per demanar informació.
- POST: Afegirà o enviarà informació.
- PATCH: Modificarà part d'un model ja existent.
- DELETE: Eliminar informació.

Errors

En cas d'errors, l'estructura del missatge es mantindrà estable al llarg de les peticions, però existeixen unes variables:

- Codi de resposta: 401 Unauthorized, 404 Not found... ("HTTP Status Codes")
- Missatge d'error (*Message*).

Veiem un exemple de missatge d'error en cas d'error a l'autenticació:

```
{
    "status": "error",
    "message": "Username or Password incorrect!"
}
```

9. Implementació i proves

Descriuré el procés d'implementació dividit per cada mòdul i seguint la metodologia indicada (*3. Metodologia*). Tal com s'indica a la planificació, el desenvolupament del projecte es va dividir en 7 mòduls.

9.1. Autenticació d'usuari

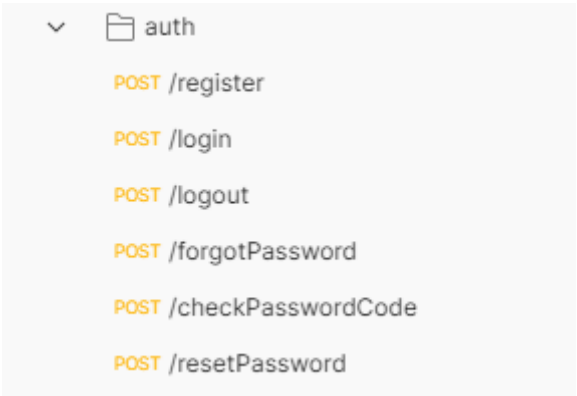
A l'hora d'autenticar l'usuari, vaig utilitzar l'anomenat "Bearer token" generat amb l'ajuda de la llibreria **JsonWebToken**. Per validar l'entrada de l'usuari, vaig utilitzar expressions **Regex**, mostrades a continuació.

Validacions Regex

- Email (Correu electrònic): `/(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])/`
  ("JavaScript : Email Validation")
- Password (Contrassenya): `/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/`
- Phone (Telèfon): `/^[+]*[(]{0,1}[0-9]{1,4}[)]{0,1}[-\s\./0-9]*$/`
- Username (Nom d'usuari): `/[a-zA-Z0-9._]{4,}$/`
- Default (Predeterminat): `/^[a-zA-Z0-9 -_]+$/`

API i Base de dades

De cara a l'API, vaig necessitar noves rutes per a cada una de les accions de l'aplicació (*Imatge 19*).



Imatge 19. Rutes autenticació POSTMAN

Mailing i correu de recuperació

Per enviar el correu recuperació de compte, vaig utilitzar la llibreria "**Express Mailer"** ("Npm: Express-Mailer"), juntament amb el motor de renderitzat de plantilles **pug** ("Getting Started – Pug") .

També vaig necessitar crear un correu "bikeinspectioner@gmail.com", i desactivar la seguretat d'aplicacions de tercers de Google per a poder enviar correus des de l'aplicació.

El codi "pug" que funciona com a plantilla és el següent:

```
doctype html
html
|  head
|  body
|    div
|    p Your BikeInspec recovery code:
|      i #{resetCode}
```

Per realitzar l'enviament del correu, i després d'inicialitzar la llibreria amb les nostres dades i generar el codi pertinent, enviem el correu.

```
const sendPasswordCode = async (mail, code) => {
  var mailOptions = {
    to: mail,
    subject: "BikeInspec - Recovery code",
    resetCode: code,
  };

  // Send email.
  app.mailer.send("forgotPassword", mailOptions, function (error, message) {
    error && console.log(error);
  });
};
```

9.2. Afegir nova motocicleta

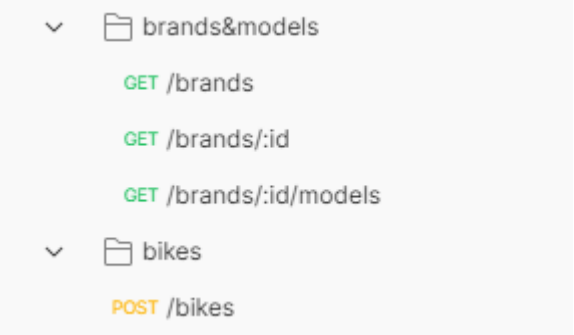
Un cop acabat el mòdul d'autenticació de l'usuari ja ens podem posar a treballar en la funcionalitat particular de l'usuari identificat. Podem veure els requeriments al diagrama de casos d'ús (*Imatge 16*).

Ens trobarem amb 2 processos principals:

- Afegir motocicleta amb paràmetres predefinits.
- Afegir motocicleta sense paràmetres predefinits.

API i Base de dades

En l'apartat de Back-end, haurem d'haver creat 4 noves rutes (*Imatge 20*).



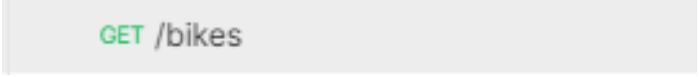
Imatge 20. Rutes mòdul 2. POSTMAN

9.3. Llistat de motocicletes

El llistat de les motocicletes el trobarem a la pantalla principal, en aquest mòdul no em vaig trobar amb cap problema específic.

API i Base de dades

En aquest mòdul ens trobarem amb només una nova petició a l'API (*Imatge 21*), en la qual agafarem la informació de motocicletes de l'usuari pertinent.



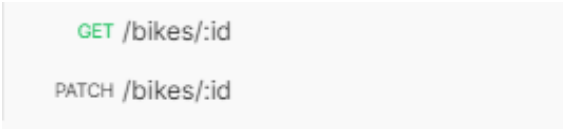
Imatge 21.Rutes mòdul 3. POSTMAN.

9.4. Revisions d'una motocicleta ja existent

La decisió més important d'aquest mòdul era el com permetre a l'usuari visualitzar i modificar l'estat d'una revisió de manera simple i visual. Vaig optar per afegir una barra, situada a sota de cada revisió, on es pot visualitzar i canviar l'estat.

API i Base de dades

Per a poder realitzar aquestes accions, només van ser necessàries dues noves rutes, una per agafar la informació d'una motocicleta i l'altra per modificar-la (*Imatge 22*).



Imatge 22. Rutes mòdul 4. POSTMAN.

9.5. Modificar/Actualitzar la informació d'una motocicleta

Per acabar amb l'administració de motocicletes, ens faltava poder modificar la informació de cada una d'elles (distancia per any, distancia total, etc).

API i Base de dades

Per aquest mòdul ja teníem pràcticament totes les rutes necessàries a l'API. Ens faltava únicament el d'eliminar motocicleta (*Imatge 23*).

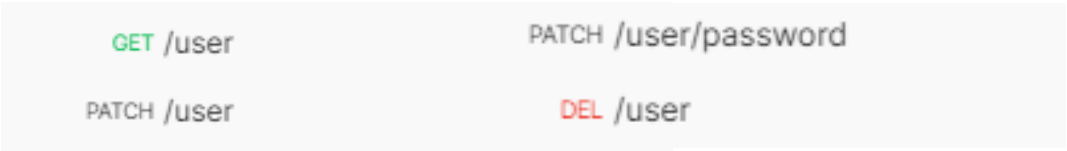
Imatge 23. Rutes mòdul 5. POSTMAN

9.6. Administrar informació d'usuari autenticat

Com la majoria d'aplicacions que guarden informació sobre usuaris, és necessari poder modificar i actualitzar aquesta informació. També és necessari que un usuari es pugui desconnectar de l'aplicació, el "Logout".

API i Base de dades

Aquí vaig crear quatre noves rutes (*Imatge 24*): una per agafar la informació, dues per modificar i una per eliminar l'usuari. Cal saber que eliminar l'usuari no és una acció realitzable des de l'aplicació, seria necessari contactar al responsable de l'aplicació (en cas de publicar-la).



Imatge 24. Rutes mòdul 6. POSTMAN

9.7. Notificacions

En arribar a aquest mòdul, l'aplicació ja era funcional. Les notificacions són una part molt útil de les aplicacions i milloren considerablement l'experiència de l'usuari.

Actualment, el temps i kilòmetres recorreguts per cada motocicleta s'actualitzen a partir d'unes dades entrades i determinades per l'usuari. Tot i això, la intenció futura del projecte és controlar aquests paràmetres des del mateix vehicle. Per això, s'utilitzaran les "push notificacions", enviades des del servidor.

No només era necessari enviar notificacions a l'usuari, també feia falta actualitzar automàticament el temps o distància restant per a cada revisió. Així que necessitàvem trobar com i quan fer aquesta "actualització automàtica".

Automatització

La solució que vaig trobar va ser, realitzar l'actualització cada dia a una hora concreta i, posteriorment a modificar les dades, enviar les notificacions pertinents.

Per programar que una funció s'executés cada dia a l'hora programada vaig utilitzar la llibreria **cron** ("Npm: Cron"). Podem observar que el codi s'executarà el segon 0, minut 0, i hora 0, de qualsevol dia.

```
//Notifications
cron.schedule("0 0 0 * * *", () => {
  updateIncomingRevisions()
    .then(checkNotifications)
    .catch(() => {
      console.log("Error updating incoming revisions");
    });
});
```

Push Notification

A l'hora d'enviar les notificacions, EXPO ens proporciona la seva pròpia API. El codi per enviar la notificació és el següent.

```
fetch("https://exp.host/--/api/v2/push/send", {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(message),
  })
```

10. Implantació i resultats

10.1. Implantació

Per implantar l'aplicació és necessari col·locar l'API i Base de dades en un servidor a internet. Hi ha múltiples serveis en aquest sentit, però els 3 més importants són: **Google Cloud** ("Cloud Computing, Hosting Services, and APIs"), **Amazon Web Service** ("Amazon Web Services (AWS) - Cloud Computing Services") i **Microsoft Azure** ("Azure Cloud Computing Services")**.**

Per la base de dades, que teníem instal·lada de forma local, podem utilitzar **MongoDB Atlas** (MongoDB), una eina específica per a instal·lar bases de dades MongoDB remotament. Per a l'API Rest, utilitzarem **Google Cloud**.

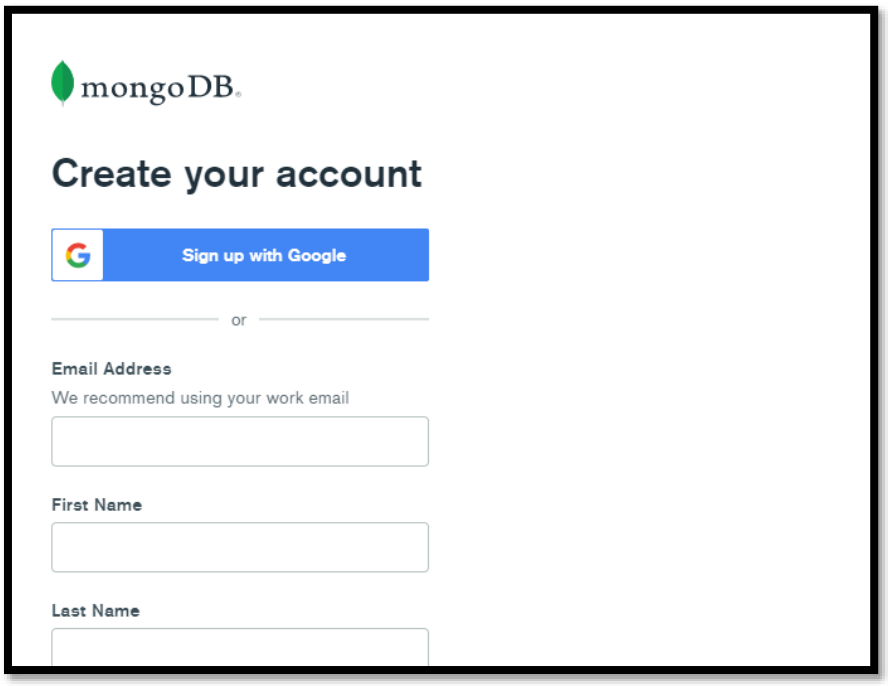
MongoDB Atlas

Per afegir una base de dades a MongoDB Atlas és molt senzill, necessitarem:

- Crear un compte de MongoDB Atlas.
- Afegir / Crear una nova organització, projecte i clúster.
- Configurar les adreces IP des d'on es permet accedir, i crear un usuari per al projecte específic.
- Guardar la cadena de caràcters per a la connexió a la base de dades.

Crear compte

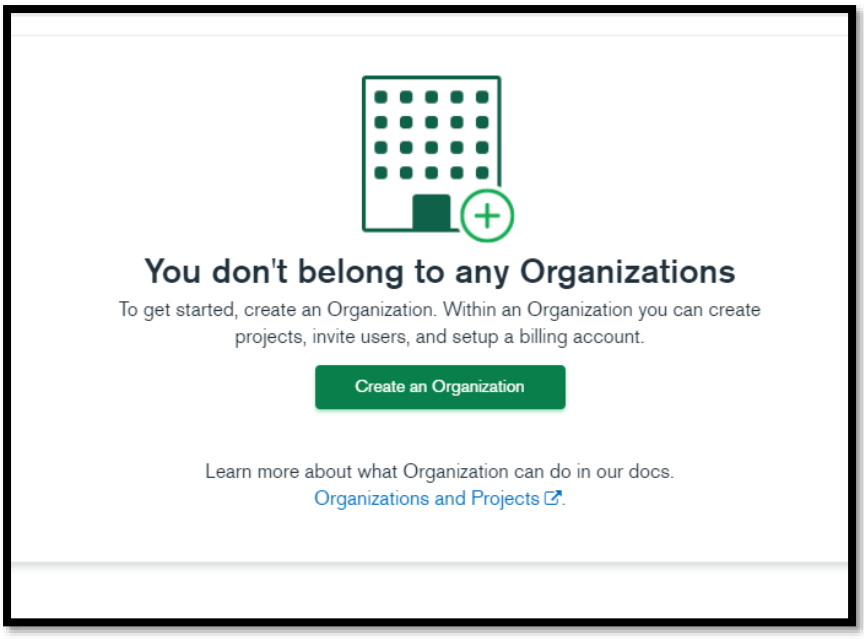
Podrem crear un compte a través de l'enllaç: "Create your accound MongoDB"



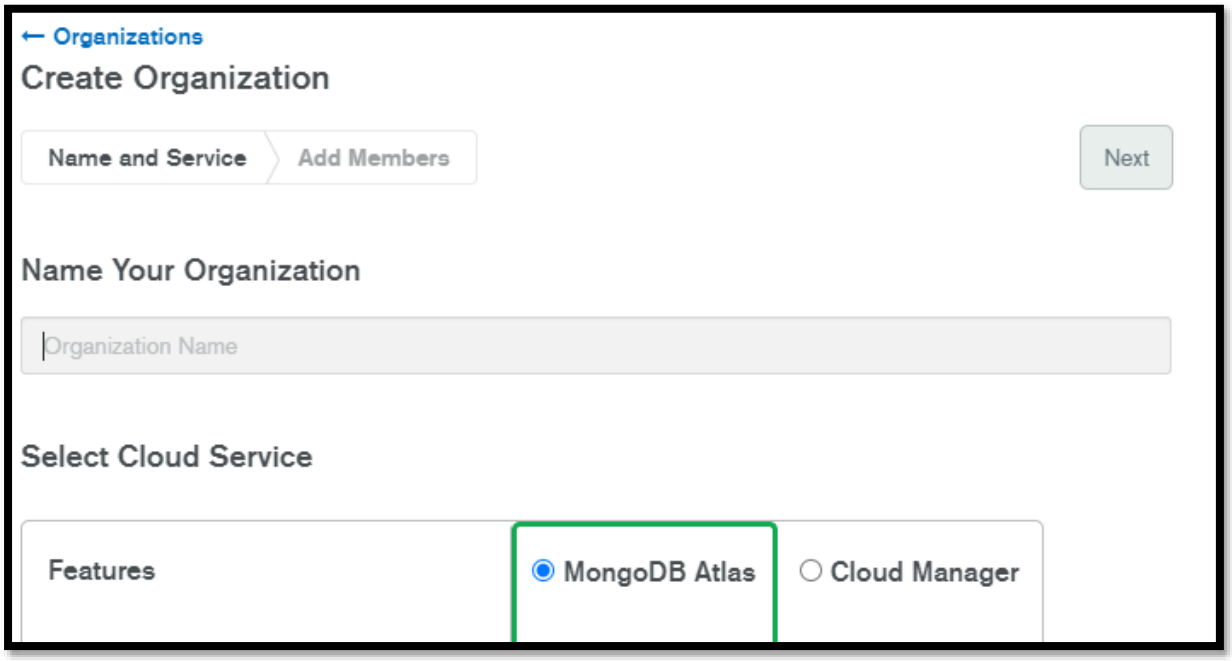
Imatge 25. MongoDB Cloud, Sign Up

Crear nova organització, projecte i clúster

Un cop ens hem creat el compte, ens trobarem que no pertanyem a cap organització (*Imatge 26*). En crearem una de nova de tipus “MongoDB Atlas” (*Imatge 27*), amb els usuaris i permisos predeterminats.

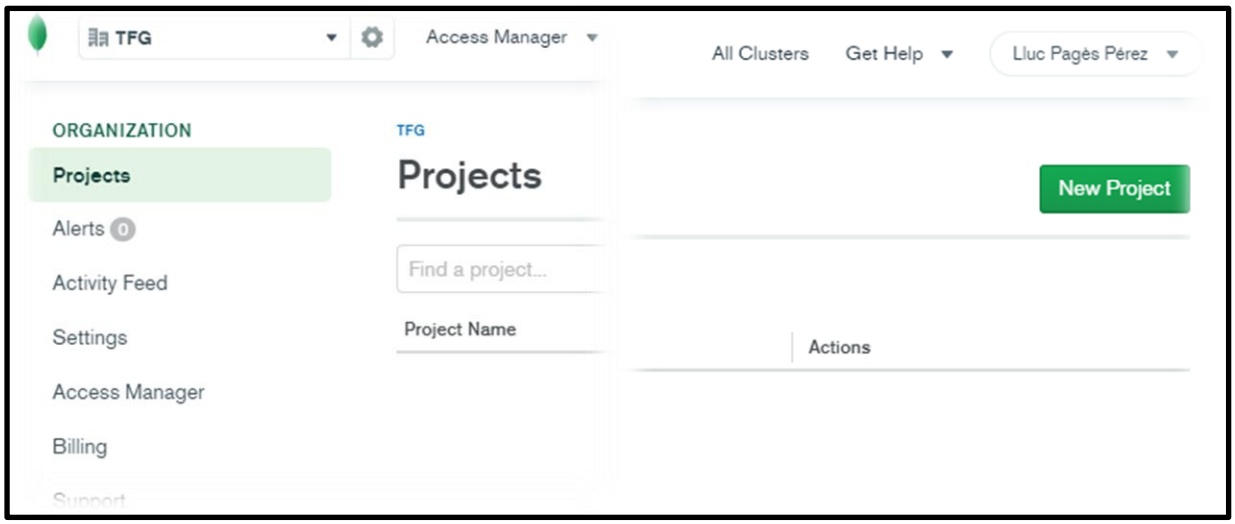


Imatge 26. MongoDB Cloud, No organizations



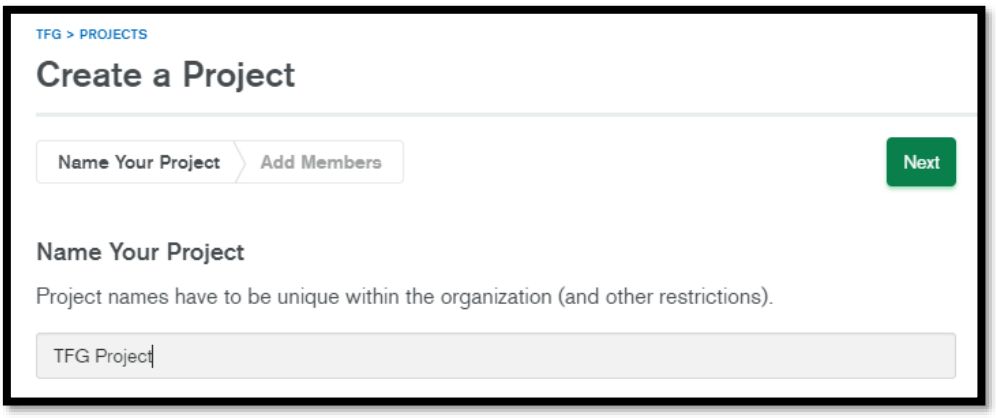
Imatge 27. MongoDB Cloud,
Create organization

Un cop tenim l'organització creada, ens trobarem automàticament dins d'aquesta, dins la pestanya de projectes “Projects” (*Imatge 28*). Crearem un nou projecte a través del botó “New Project”:

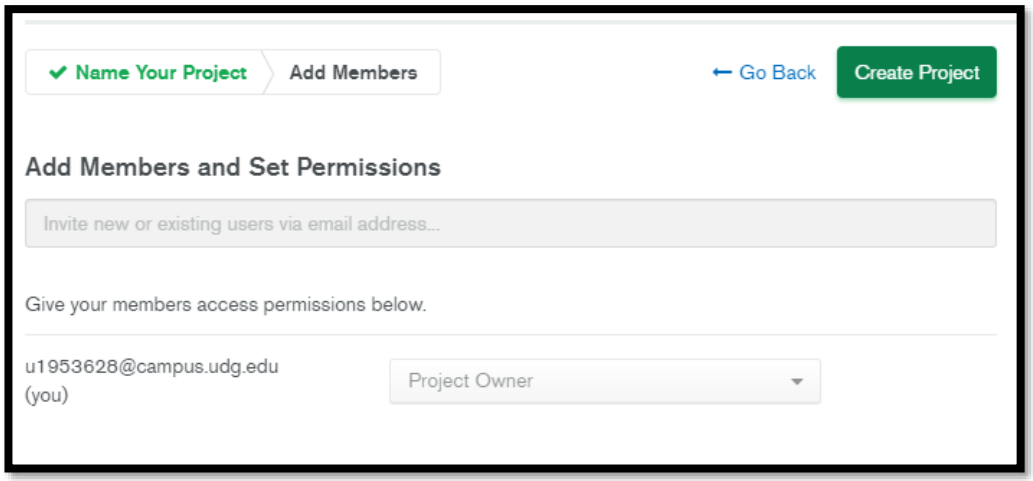


Imatge 28. MongoDB Cloud, Projects

Afegirem el nom del projecte que ens interessi (*Imatge* 29), i deixarem els membres predeterminats per la web (*Imatge 30*).

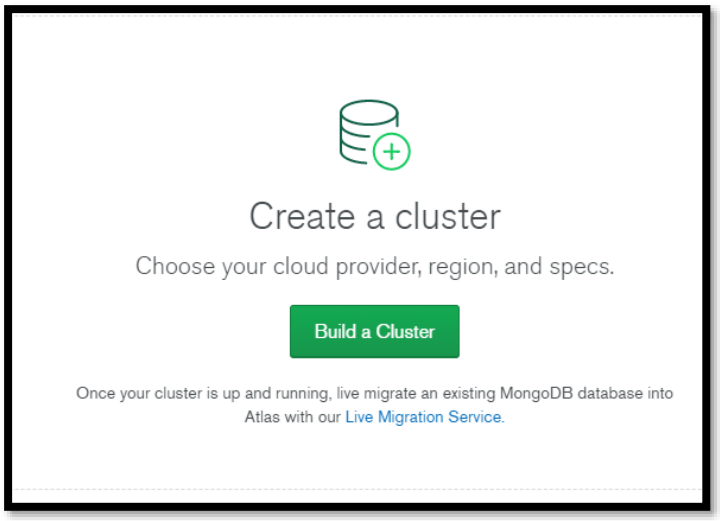


Imatge 29. MongoDB Cloud, Create project 1



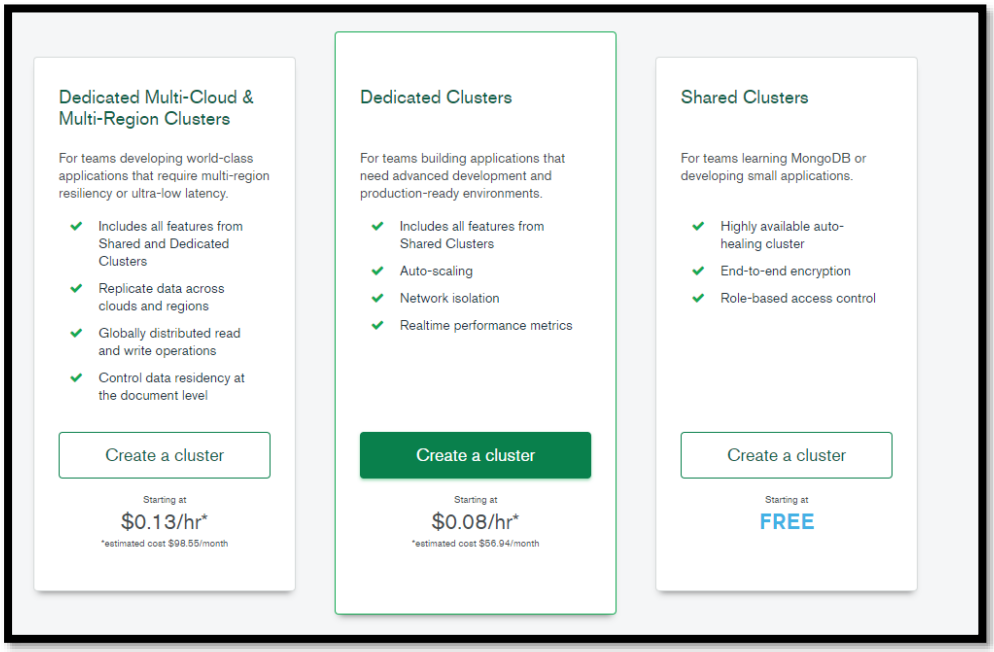
Imatge 30. MongoDB Cloud, Create project 2

Ara que ja hem creat una organització i un projecte, també necessitem afegir un "Cluster" dins del projecte. En la següent pantalla (*Imatge 31*) haurem de clicar sobre "Build a Cluster".



Imatge 31. MongoDB Cloud, Create cluster

Ens trobarem amb 3 opcions diferents (*Imatge 32*): **Dedicated Multi-Cloud & Multi-Region Cluster**, **Dedicated Clusters**, **Shared Clusters**. Escollirem la tercera opció, ja que no necessitarem realitzar cap inversió de capital.

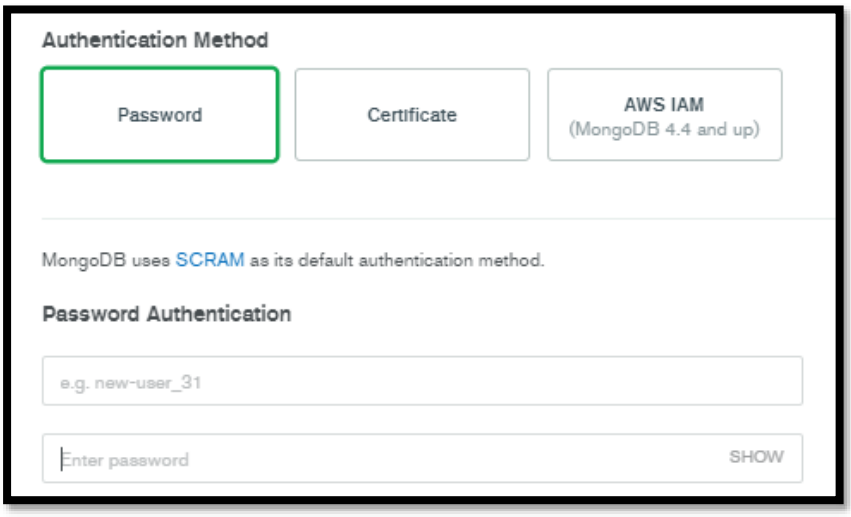


Imatge 32. MongoDB Cloud, Create cluster 2

Les accions necessàries per a completar la creació del clúster són opcionals a gust de cadascú. Finalment, **ja haurem creat Organització, Projecte i Cluster!**

Crear usuari i configurar IPs

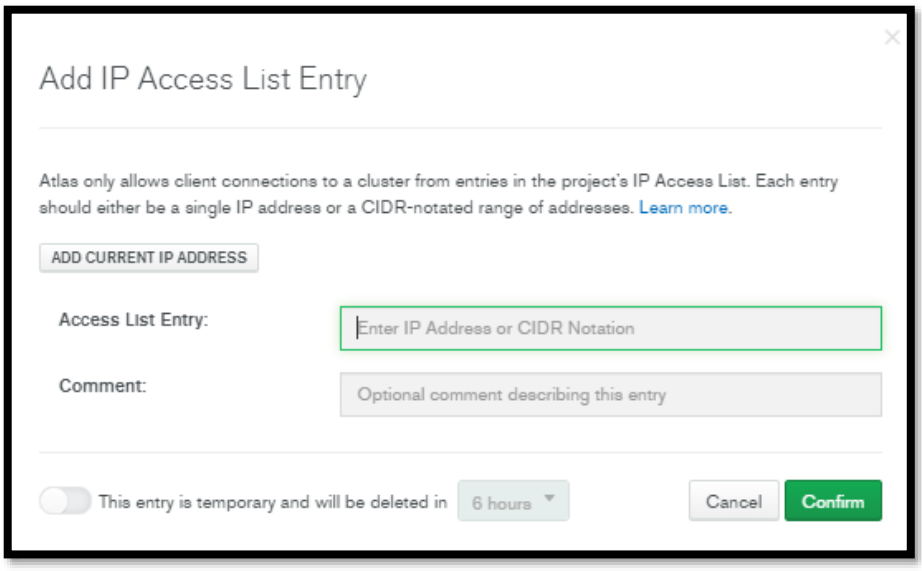
Per a crear un usuari ens haurem de dirigir a la ruta “Database Access > Database Users > ADD NEW DATABASE USER”.



Imatge 33. MongoDB Cloud, Create user

Tenim diferents opcions per crear l’usuari, en aquest cas utilitzarem Usuari + Contrasenya (**Password**). Haurem de recordar els valors per poder connectar la aplicació a la base de dades posteriorment.

Per configurar les IPs permeses a la base de dades ho farem a través de la pestanya “**Network Access”**. Inicialment no hi trobarem cap IP, així doncs, hem d’afegir l’IP del servidor des del qual s’accedirà a la base de dades. També podem permetre accés a tothom.

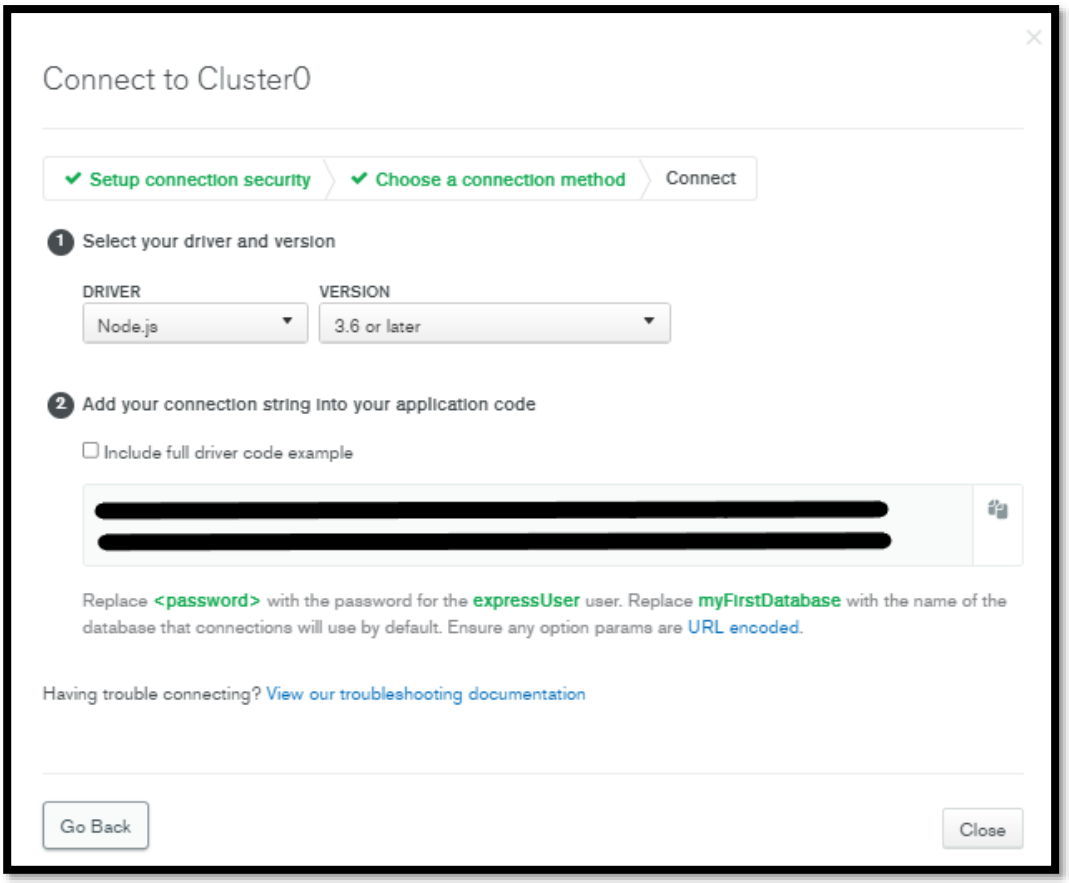


Imatge 34. MongoDB Cloud, Add IP

Connexió a la base de dades

Per a poder connectar l'API Rest que posteriorment implementarem en un altre servidor, necessitem una cadena de caràcters que ens proporcionarà MongoDB.

A la pestanya de **Clusters**, en el botó de "**Connect**" i seleccionant l'opció de "**Connect your application"** obtindrem la cadena necessària.



Imatge 35. MongoDB Cloud, Connect to cluster

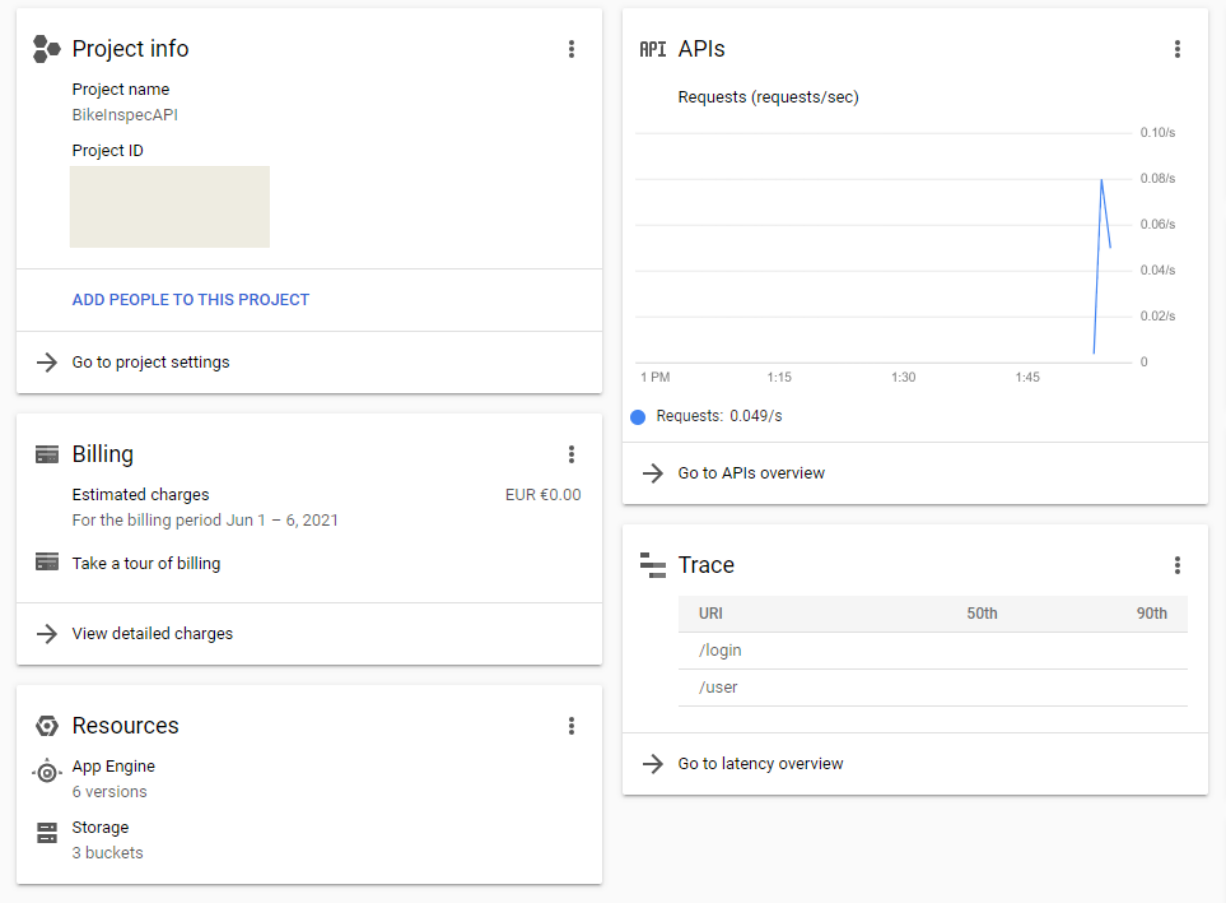
Google Cloud

Per instal·lar l'API REST al servidor a Google Cloud cal el següent:

- Crear compte Google Cloud.
- Crear projecte i el servei pertinent.
- Instal·lar Google Cloud SDK.
- Configurar el projecte i modificar el codi final.
- Enviar els fitxers pertinents al servidor.

Per a realitzar tots aquests passos vaig seguir els passos descrits en la pàgina web [“Building a Node.Js App on App Engine”](#) .

Veiem la pantalla del projecte final (*Imatge 36*):

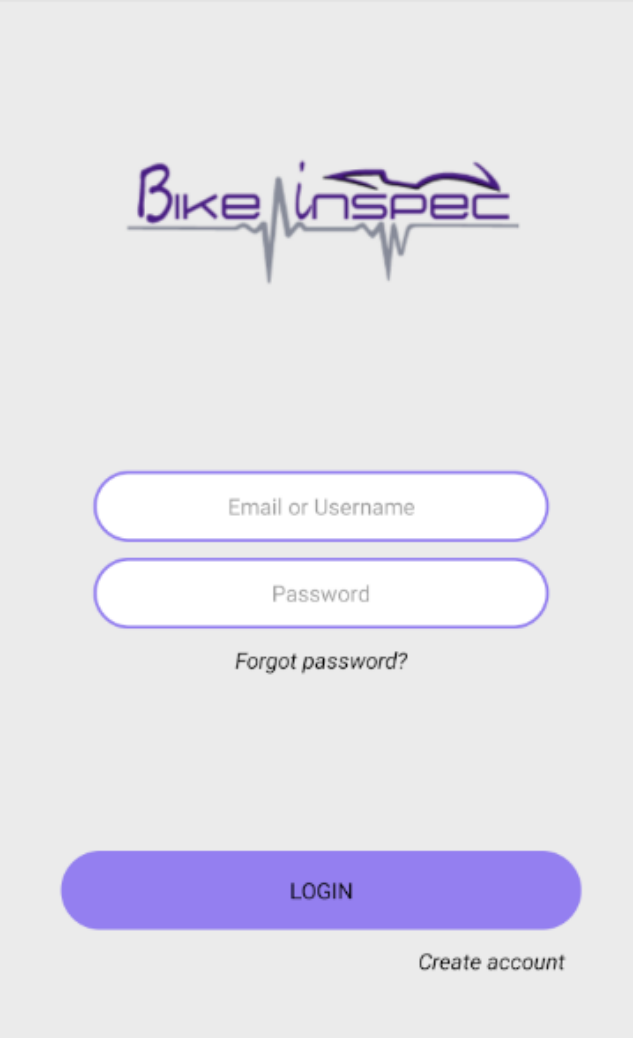


Imatge 36. Google cloud project

10.2. Resultats

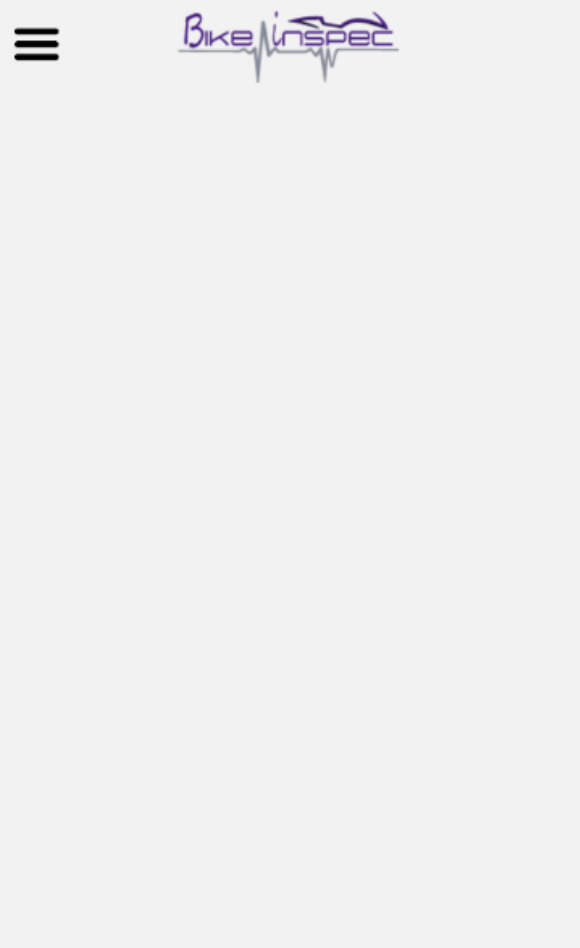
Inici de sessió (Login)

A l'iniciar l'aplicació per primera vegada ens trobem amb la pantalla de login (*Imatge 37*). Aquí ens demanarà les dades d'autenticació.

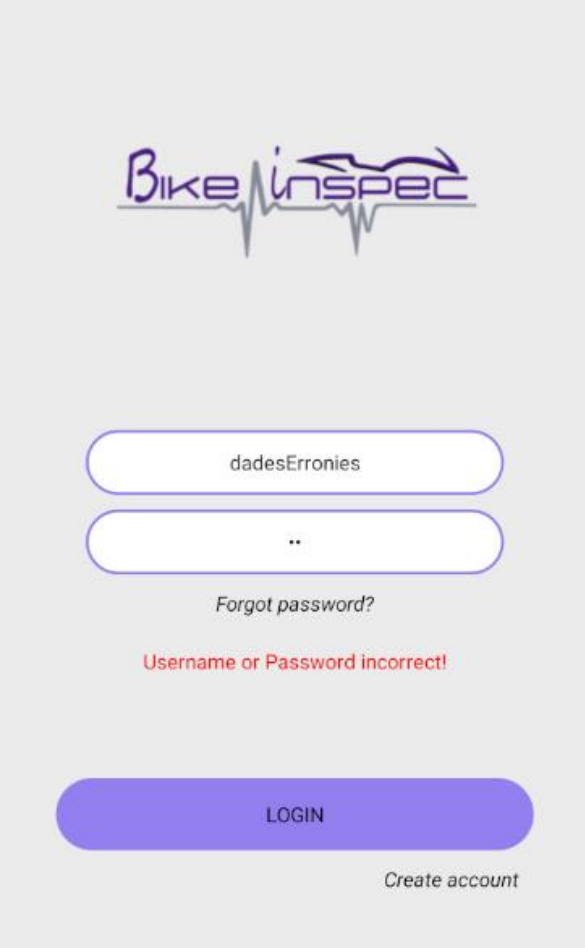


Imatge 37. Pantalla login

En cas d'entrar les dades correctes, accedirem a l'aplicació (*Imatge 38*). En cas que les dades siguin errònies ens mostrarà un missatge (*imatge 39*).

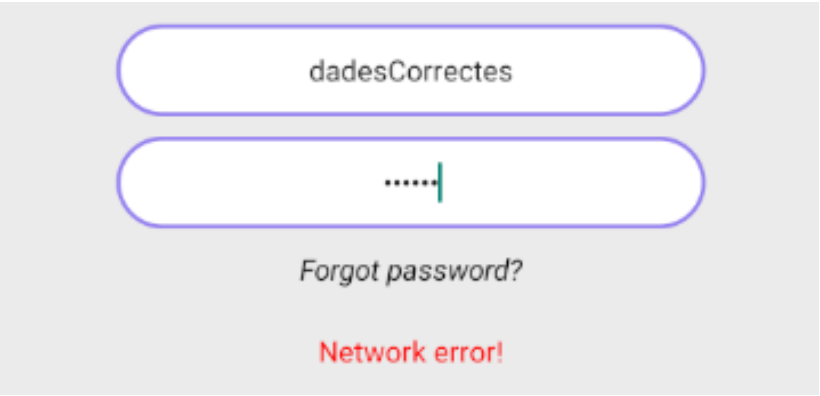


Imatge 38. Pantalla usuari autenticat



Imatge 39. Pantalla dades errònies

També podem veure que en cas de no tenir connexió, ens mostra el missatge pertinent (*Imatge 40*).

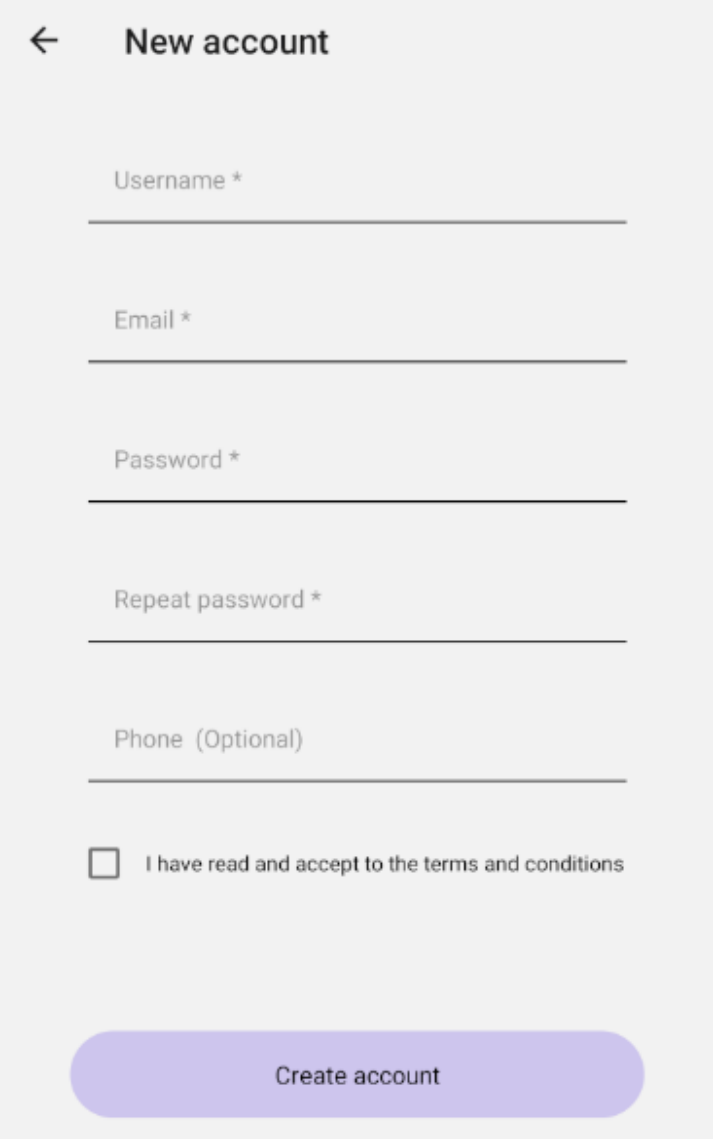


Imatge 40. Pantalla usuari autenticat

Registre

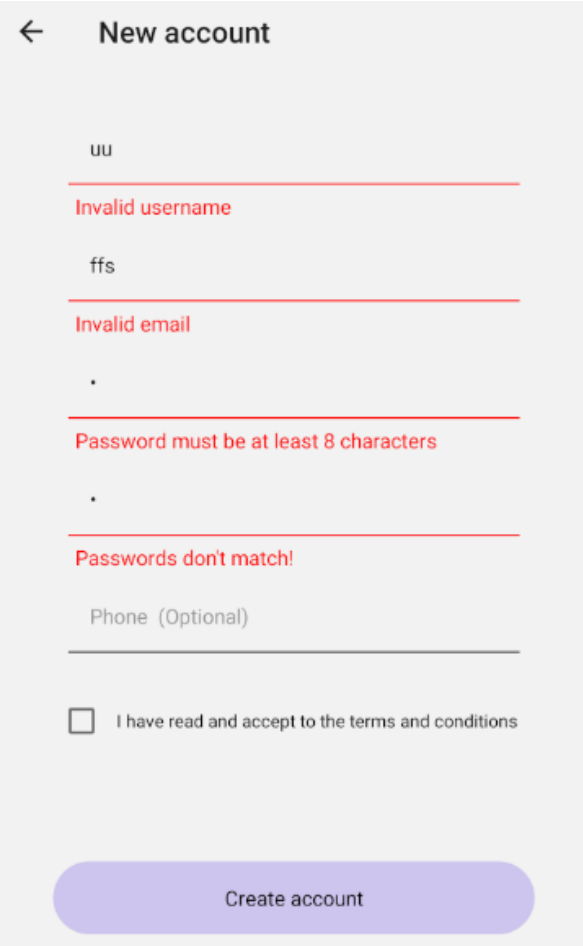
Si a la pantalla de login (*Imatge 37*), l'usuari prem sobre "Create account" apareixerà una pantalla per registrar un nou usuari (*Imatge 41*). Es validaran tots els camps:

- L'usuari (username), correu (email), contrasenya (password) i repetir la contrasenya (repeat password) són camps obligatoris, i serà necessari tenir-los marcats en verd.
- El número de telèfon (Phone) serà opcional.
- És necessari acceptar els termes i condicions (per causes de desenvolupament encara no estan creats).

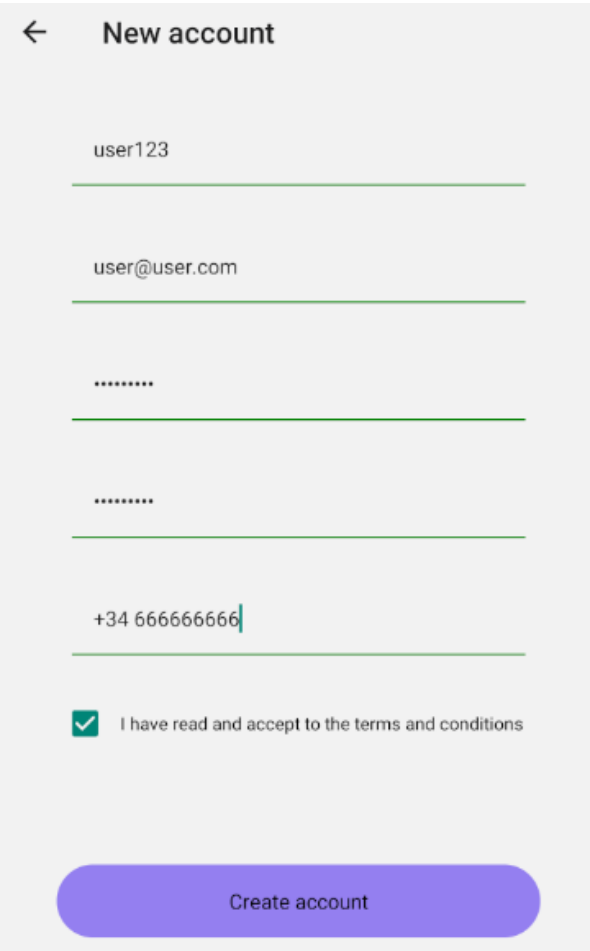


Imatge 41. Pantalla de registre d'usuari

En cas de tenir tots els camps validats, el botó s'activarà i es podrà crear el nou usuari (*Imatge 42*). En cas de tenir algun camp invàlid, es marcarà de color vermell i es desactivarà el botó de registre (*Imatge 43*).



Imatge 42. Pantalla de registre d'usuari. Dades errònies

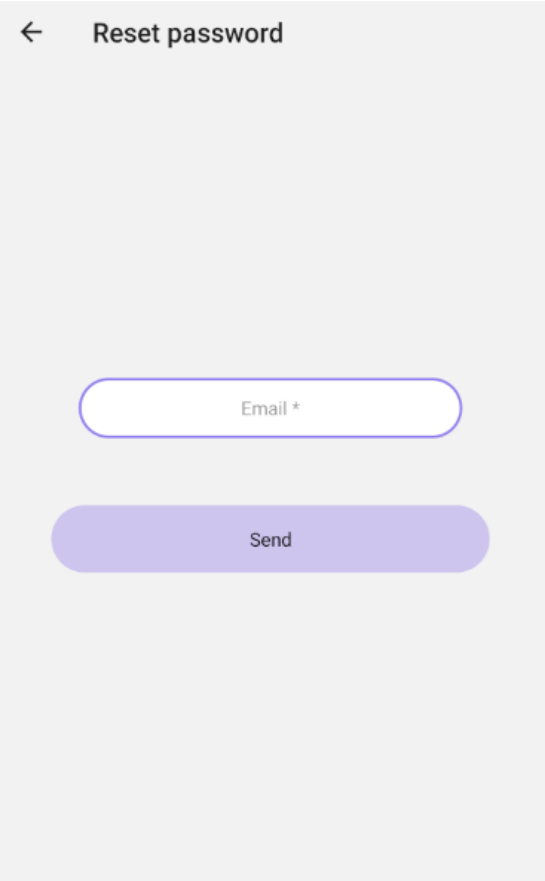


Imatge 43. Pantalla de registre d'usuari. Dades vàlides

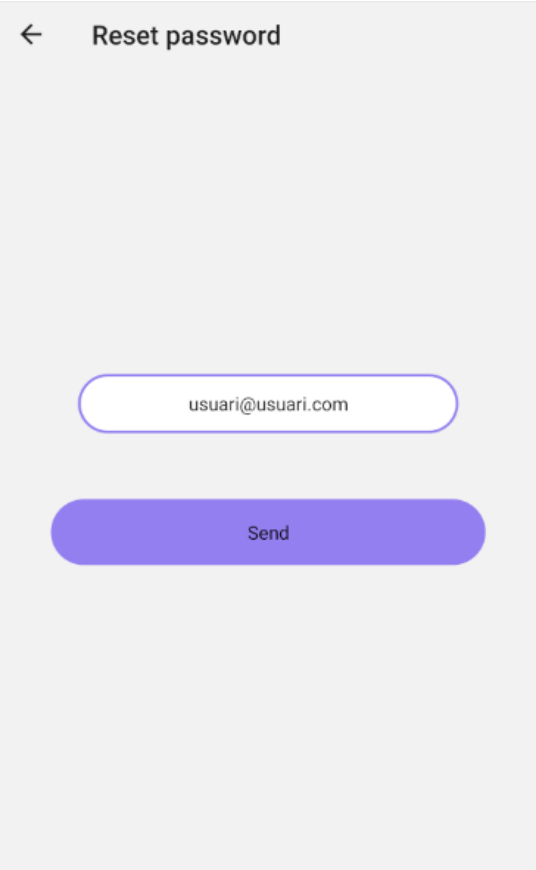
Forgot password

Si a la pantalla de login (*Imatge 37*) l'usuari prem sobre "Forgot password?", s'iniciarà el procés per recuperar la contrasenya. Per aquest procés s'utilitzarà el correu electrònic.

A la pantalla inicial (*Imatge 44*) ens trobarem amb un camp, el qual s'haurà d'omplir amb el correu electrònic del nostre usuari. A l'escriure el correu, s'activarà el botó per enviar codi de recuperació (*Imatge 45*).

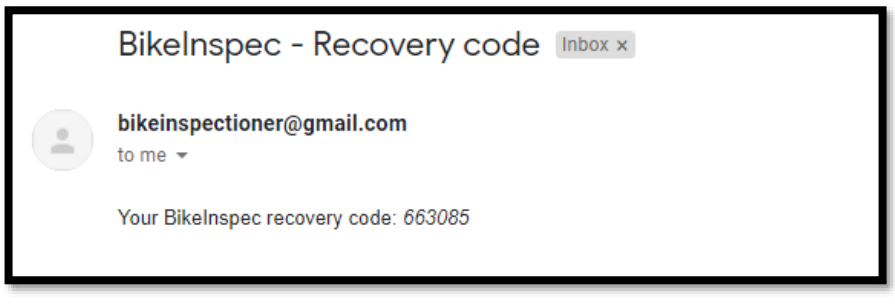


Imatge 44. Pantalla "Forgot password".



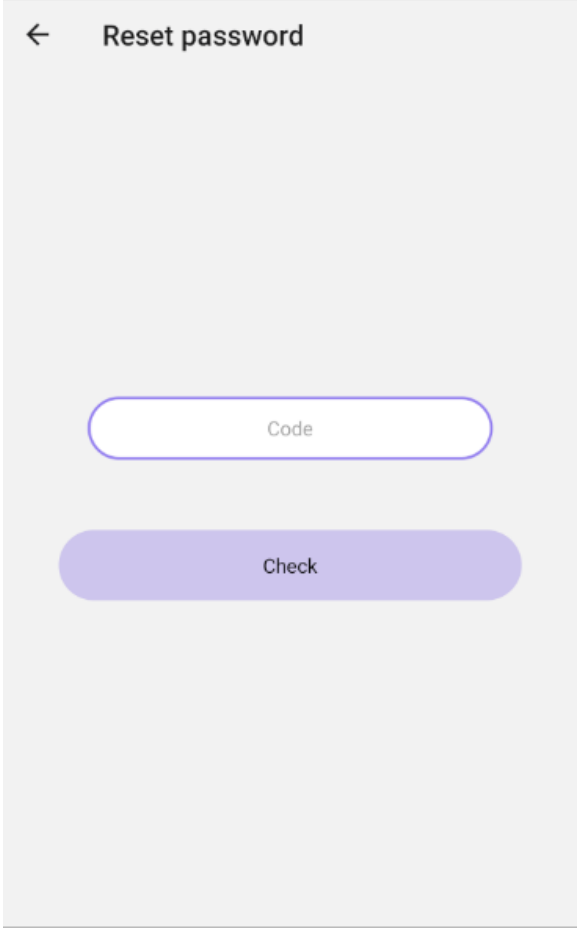
Imatge 45. Pantalla "Forgot Password". Correu Vàlid

Un cop es prem al botó "Send" (enviar), canviarem de pantalla. L'aplicació no proporcionarà informació de si l'usuari existeix, però, en cas de no existir usuari amb el correu corresponent, no s'enviarà cap missatge. Podem veure el correu rebut (*Imatge 46*):

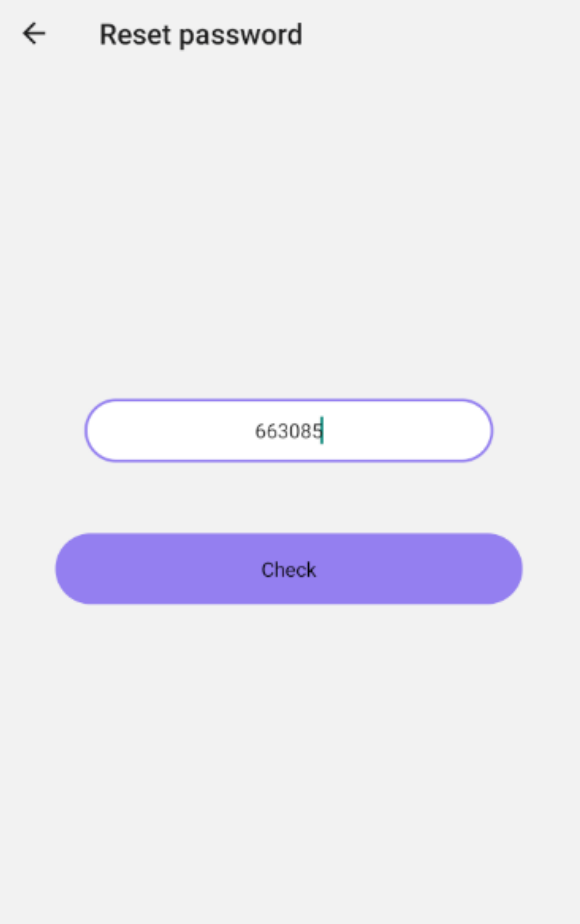


Imatge 46. Correu de recuperació.

Posteriorment, ens trobarem amb la pantalla de comprovació de codi (*Imatge 47*). De la mateixa manera que les pantalles anteriors, un cop entrem informació vàlida s'activarà el botó (*Imatge 48*).



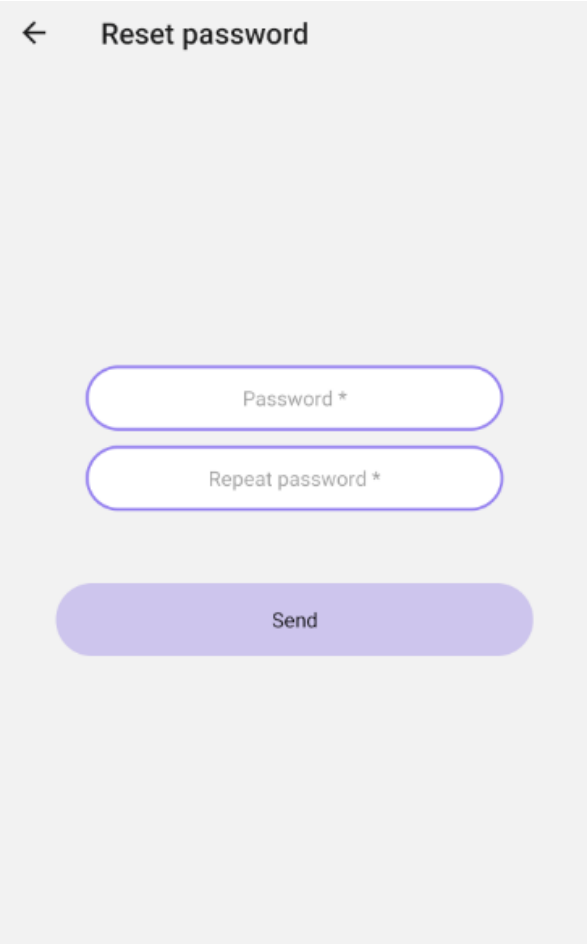
Imatge 47. Pantalla de comprovació de codi de recuperació.



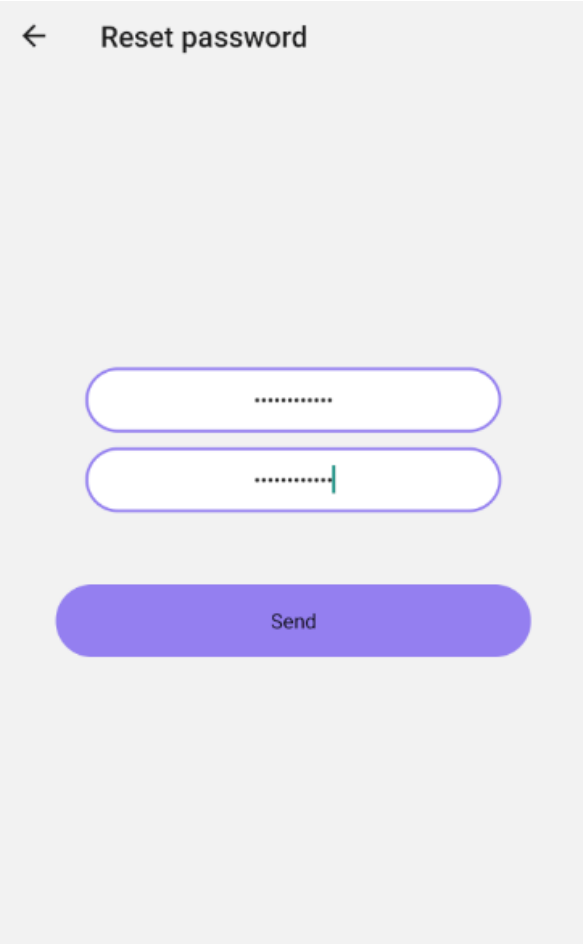
Imatge 48. Pantalla de comprovació de codi de recuperació. Informació vàlida.

En cas d'entrar un codi invàlid, no succeeix res, en cas d'afegir el codi vàlid avançarem de pantalla i ens permetrà finalment modificar la contrasenya.

Finalment a la pantalla de modificació de contrasenya (*Imatge 49*), entrarem la contrasenya desitjada als dos camps disponibles i s'activarà el botó (*Imatge 50*).



Imatge 49. Pantalla de modificació de contrasenya.

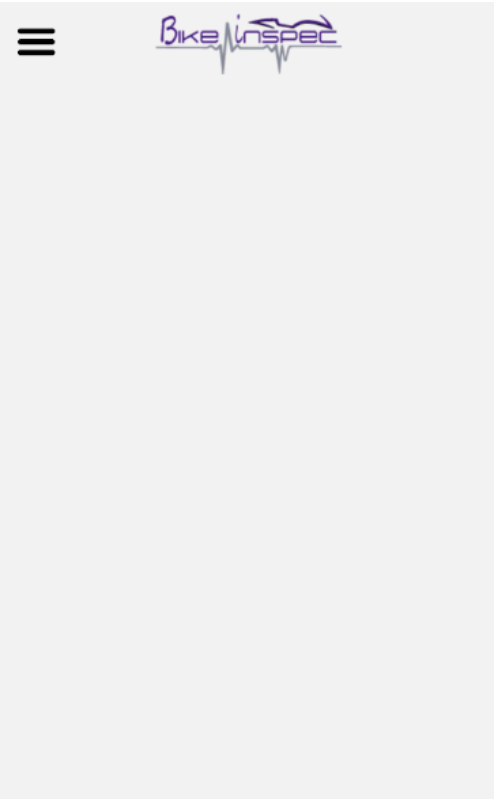


Imatge 50. Pantalla de modificació de contrasenya. Informació vàlida.

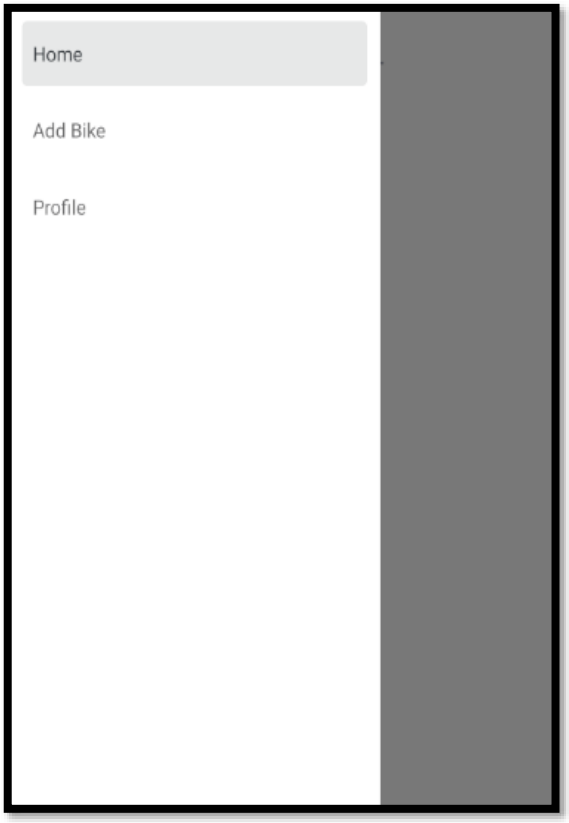
Quan enviem la contrasenya, tornem a la pantalla de login (*Imatge 37*).

Navegació

En clicar el botó de la cantonada de dalt a l'esquerra (*Imatge 51*), es desplegarà el menú lateral (*Imatge 52*).



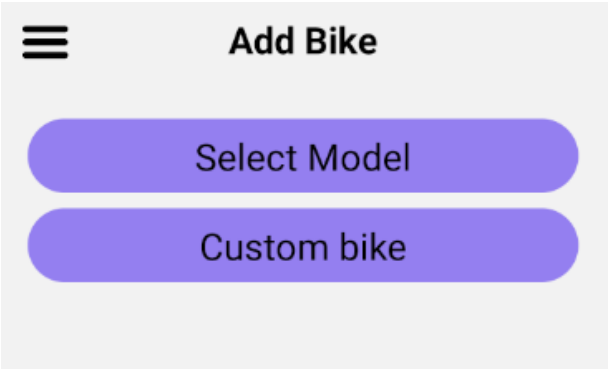
Imatge 51. Pantalla amb botó pel menú desplegable.



Imatge 52. Menú desplegable 1.

Afegir motocicleta

En prémer sobre "Add bike" al menú de navegació, canviarem de pantalla (*Imatge 53*) i podrem escollir la manera d'afegir una motocicleta nova.

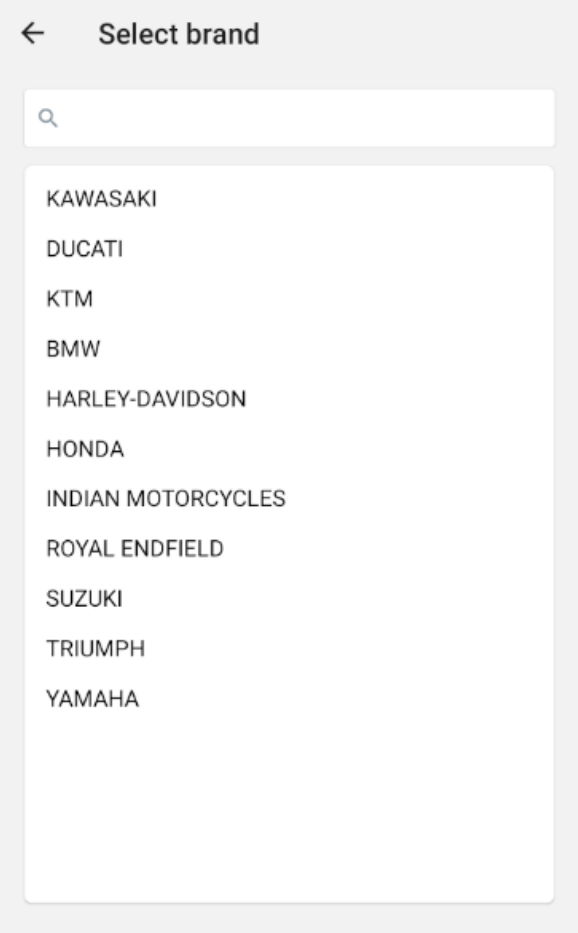


Imatge 53. Pantalla afegir motocicleta. Escollir mode.

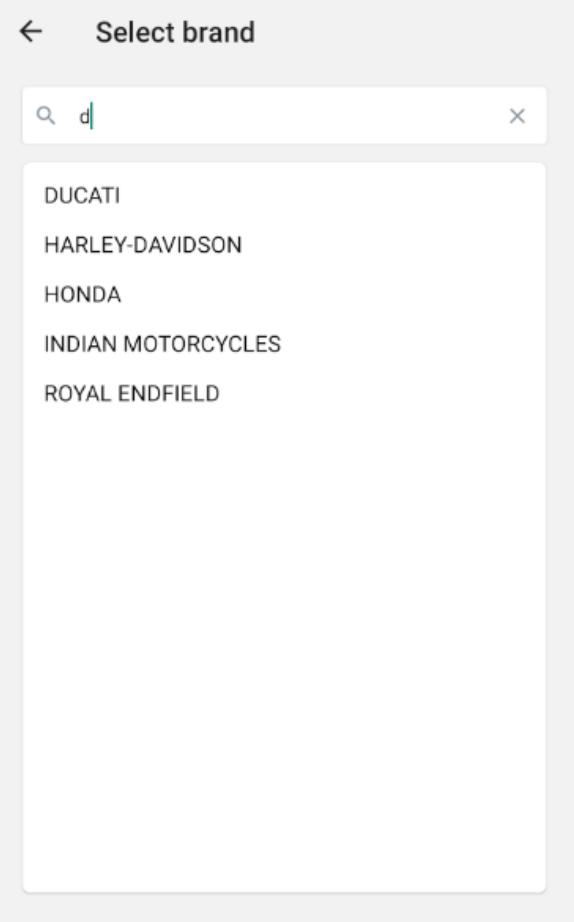
Afegir motocicleta amb paràmetres predefinits

Un cop a la pantalla d'elecció de mode (*Imatge 53*), en cas de prémer **"Select Model"** (Selecciona model), s'iniciarà el procés d'afegir motocicleta amb paràmetres predefinits.

A continuació ens apareixerà una nova pantalla per **seleccionar una marca de motocicleta** (*Imatge 54*), on tenim disponible un buscador (*Imatge 55*).



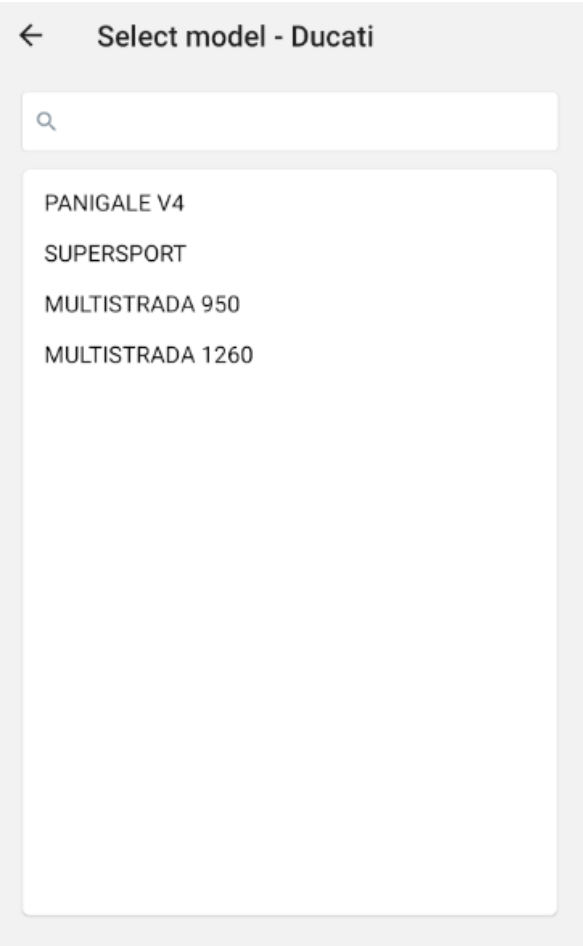
Imatge 54. Pantalla escollir marca.



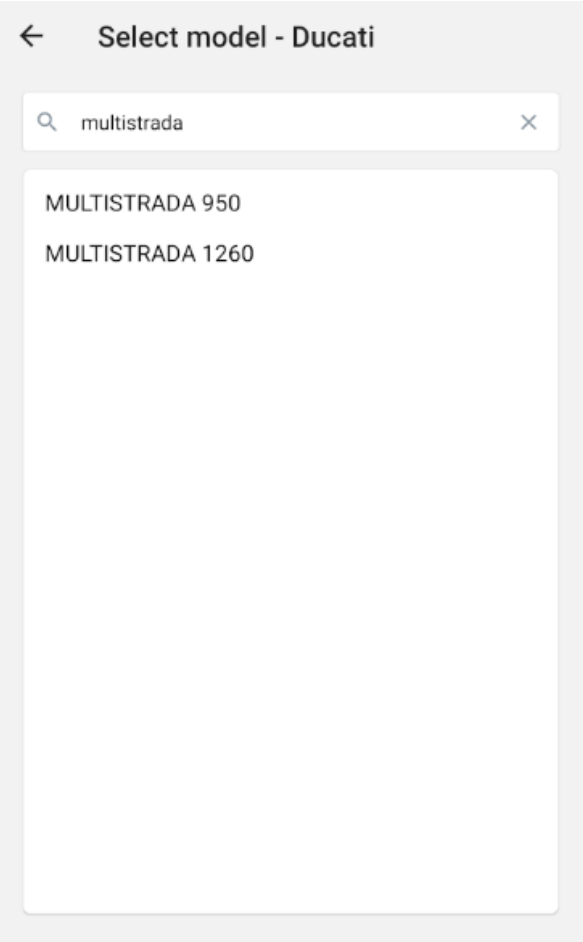
Imatge 55. Pantalla escollir marca. Filtre.

En escollir la marca prement sobre el nom, la pantalla canviarà per **escollir el model**. En la imatge que veiem a continuació (*Imatge 56*), hem escollit la marca “Ducati”.

De la mateixa manera que podem filtrar els resultats de les marques, també podem filtrar els models que ens apareixen (*Imatge 57*).



Imatge 56. Pantalla escollir model.

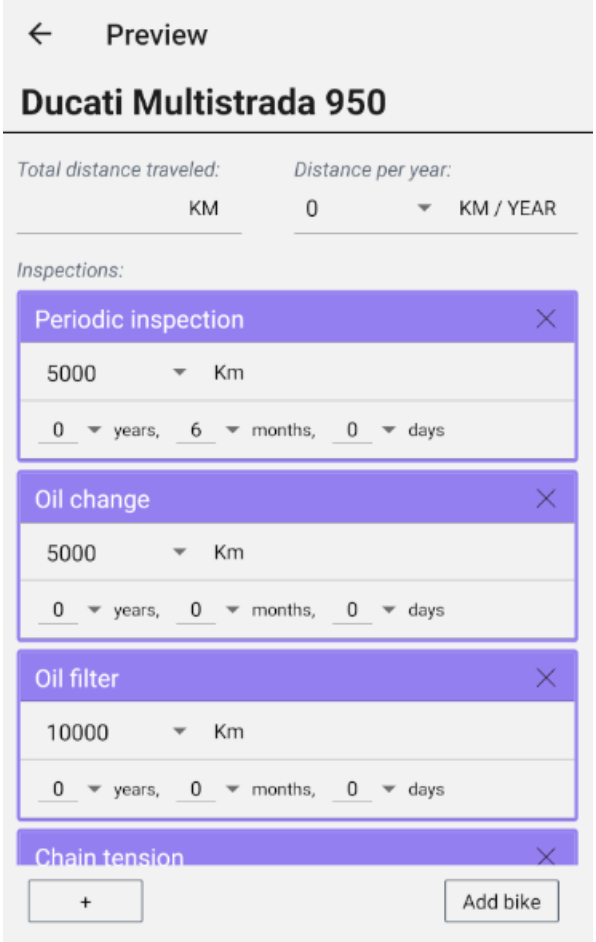


Imatge 57. Pantalla escollir model. Filtre.

Al prémer sobre el model de motocicleta que desitgem, canviarem de pantalla.

Per acabar, ens trobarem amb la pantalla final abans d'afegir definitivament la motocicleta, aquí podrem modificar tots els paràmetres específics de la mateixa i de les respectives revisions.

La intenció d'afegir una motocicleta amb paràmetres predefinits és una experiència més còmode per l'usuari. Tot i això, cada marca i model és diferent i, en la majoria de casos, és recomanable comprovar-ho amb el respectiu manual.



Imatge 58. Pantalla Preview

Imatge 59. Pantalla Preview. Scroll

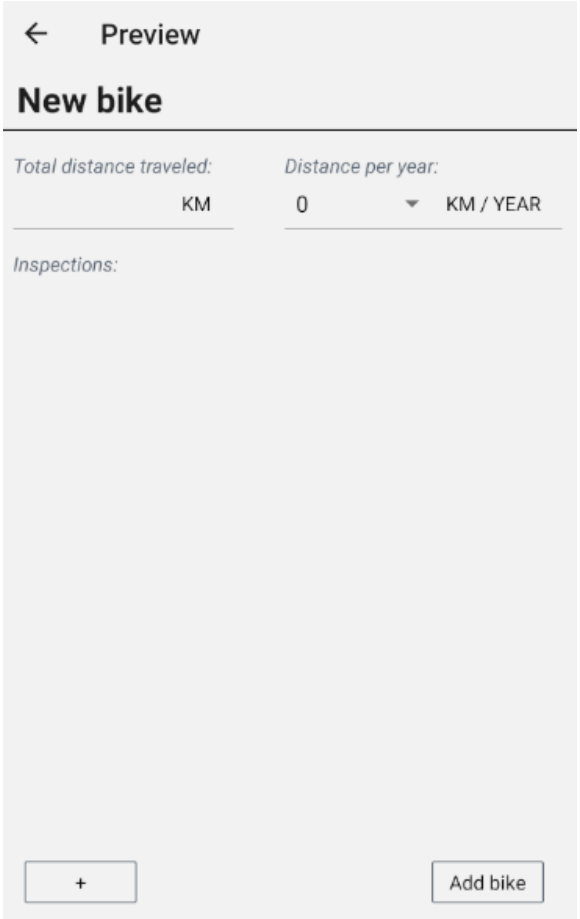
Per finalitzar, simplement és necessari prémer a la cantonada inferior-dreta, "Add bike". D'aquesta manera s'enviarà la informació a la base de dades i la motocicleta quedarà afegida!

Afegir motocicleta sense paràmetres predefinits

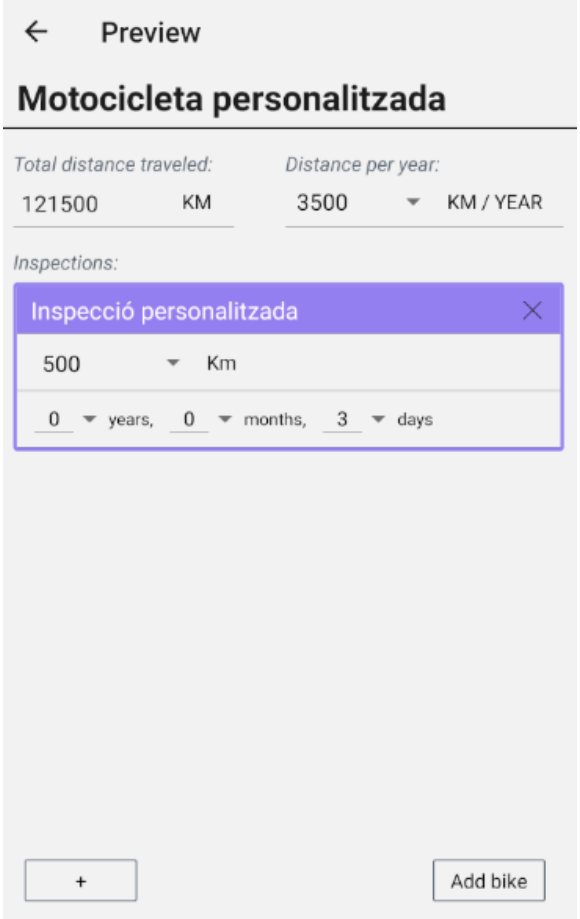
En aquest cas, si l'usuari vol personalitzar-se totes les revisions, o la seva motocicleta no es troba a la base de dades, té l'opció d'afegir-la manualment.

Un cop a la pantalla d'elecció de mode (*Imatge 53*), en cas de prémer **"Custom bike"**, s'iniciarà el procés d'afegir motocicleta sense paràmetres predefinits.

Aquest procés reutilitza l'anterior (Afegir motocicleta amb paràmetres predefinits), però va directament a l'última pantalla (*Preview*). I aquí ens trobarem amb tota la informació que haurem d'emplenar (*Imatge 60*).



Imatge 60. Pantalla Preview. Custom bike.

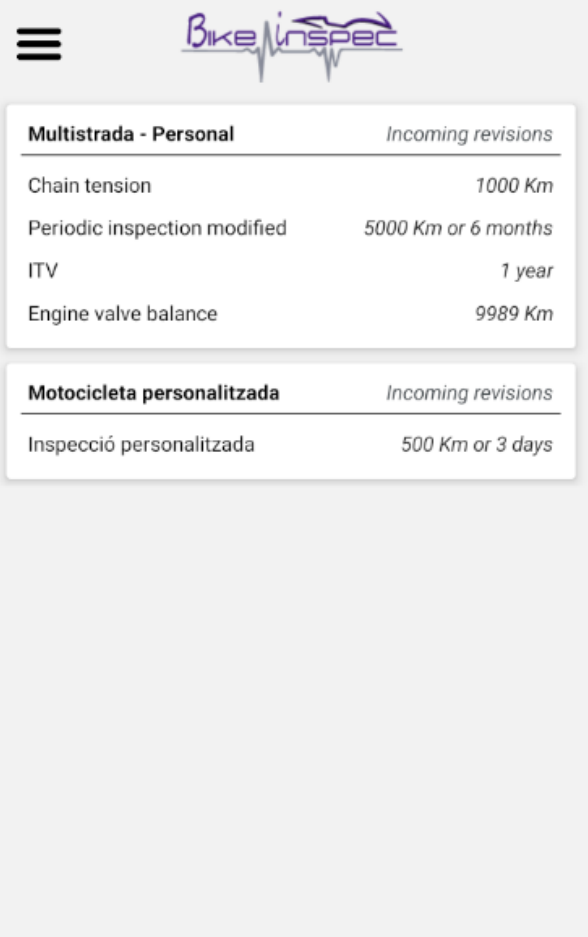


Imatge 61. Pantalla Preview Custom bike. Info modificada.

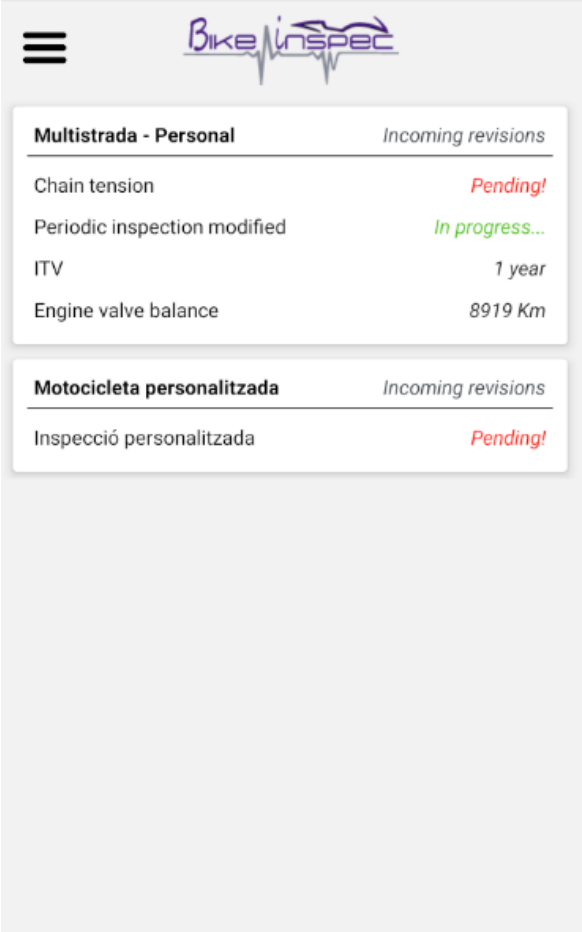
Per afegir revisions podrem prémer sobre el botó inferior-esquerre. I, de la mateixa manera que en procés anterior, podrem afegir definitivament la motocicleta amb el botó inferior-dreta.

Llistat de motocicletes

En aquest mòdul vaig optar per només mostrar part de la informació. Com es pot veure a la següent imatge (*Imatge 62*), només es mostra el nom i les revisions més properes (màxim de 4). També mostra de colors els diferents estats de cada una de les revisions.



Imatge 62. Llistat de motocicletes



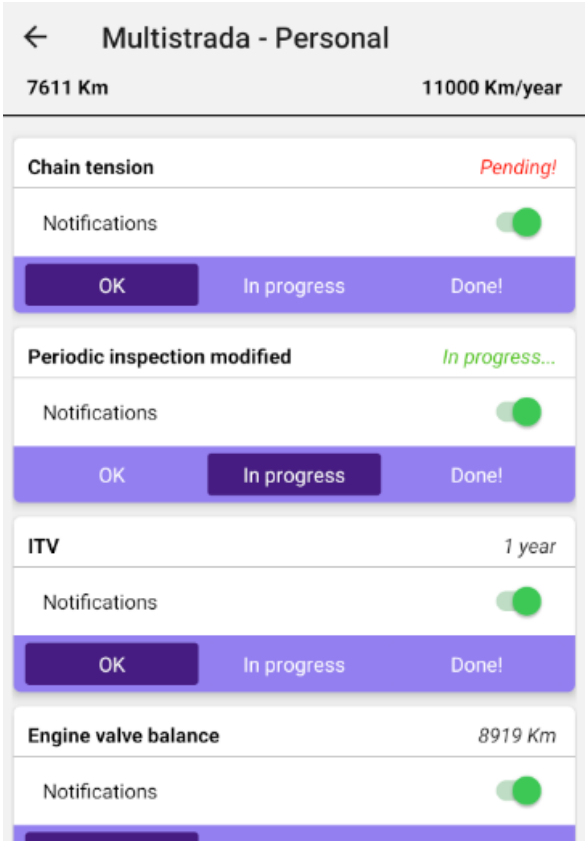
Imatge 63. Llistat de motocicletes. Colors.

Visualitzar les revisions d'una motocicleta ja existent

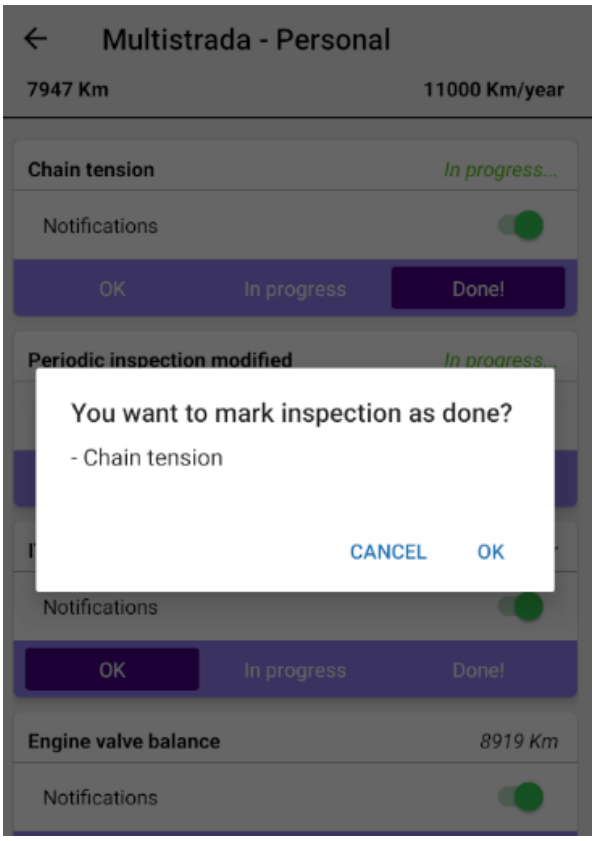
En prémer sobre una motocicleta en el llistat inicial (*Imatge 62*), se'ns obriria la pantalla de detalls de motocicleta (*Imatge 64*).

Per modificar l'estat, ho podrem fer a través dels botons inferiors a cada una de les revisions ("OK", "In progress", "Done!").

A més, en cas de marcar una revisió com a feta ("Done!") ens demanarà una confirmació per evitar errors (*Imatge 65*). Posteriorment, la revisió es donarà per acabada i es reiniciarà el període.



Imatge 64. Pantalla detalls motocicleta.

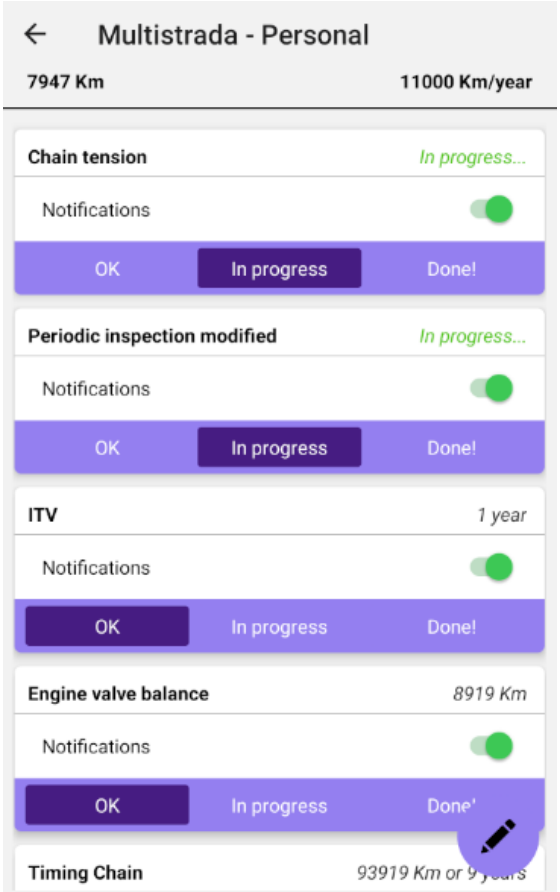


Imatge 65. Pantalla detalls motocicleta. Confirmació.

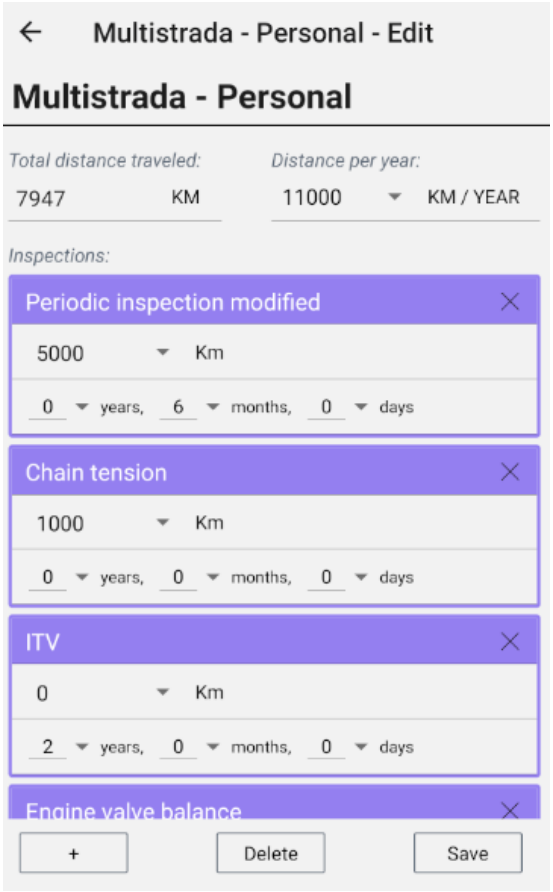
Modificar/Actualitzar la informació d'una motocicleta

Prement el botó a la cantonada inferior-dreta dels detalls de la motocicleta (*Imatge 66*), ens portarà a la pantalla de modificació de la mateixa.

Per a aquesta pantalla vaig reutilitzar la pantalla final del procés "Afegir motocicleta" (*Afegir motocicleta*). Podem veure el resultat a la imatge següent (*Imatge 67*).

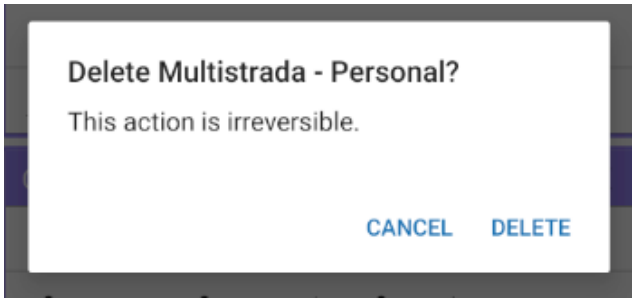


Imatge 66. Pantalla detalls motocicleta. Botó afegit.



Imatge 67. Pantalla modificar motocicleta.

En aquesta última pantalla (*Imatge 67*) també és visible un botó central per eliminar la motocicleta. En cas de voler eliminar-la, ens demanarà confirmació (*Imatge 68*).

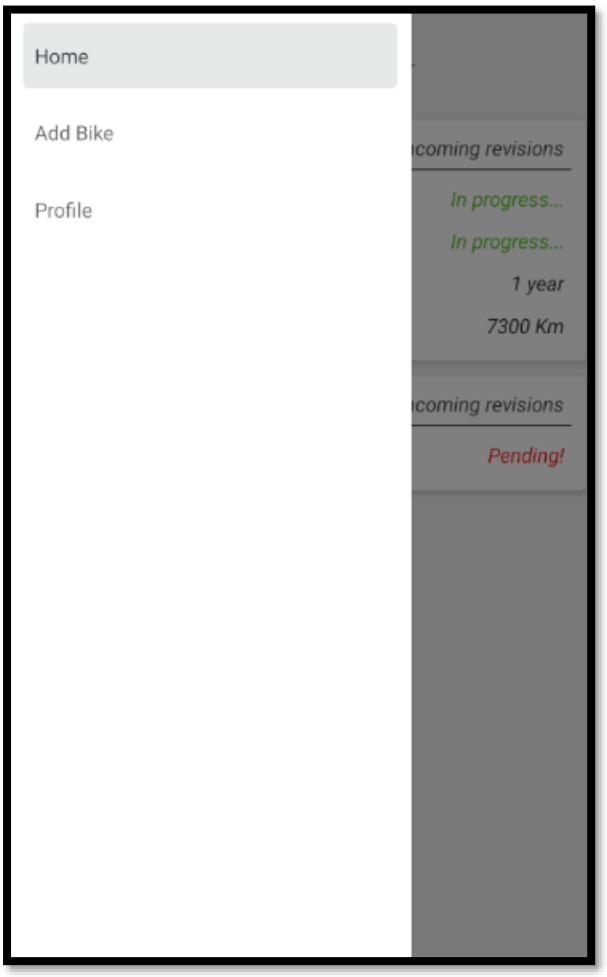


Imatge 68. Confirmació eliminar motocicleta.

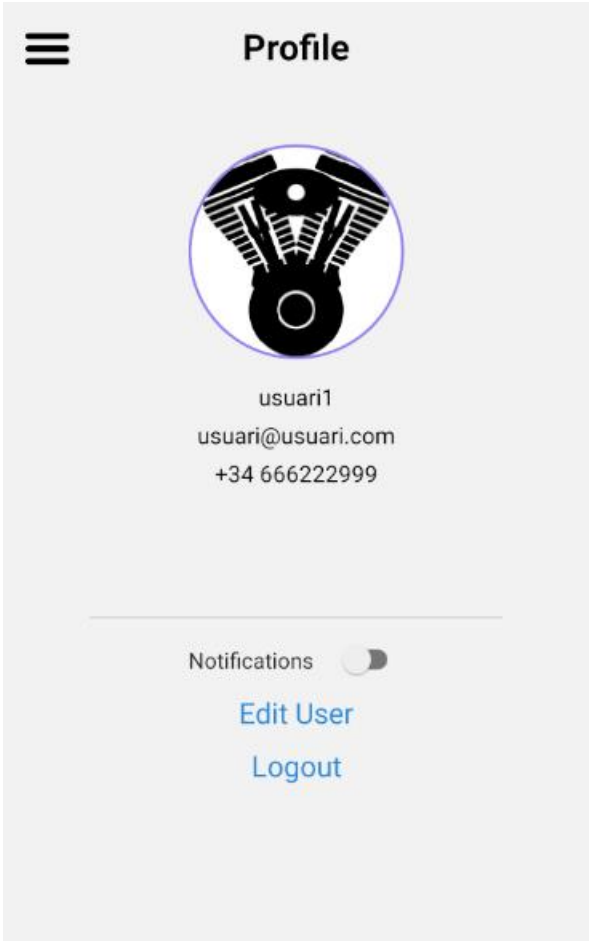
Administrar informació d'usuari autenticat

Per poder visualitzar la pantalla de l'usuari, primer hem de navegar fins ella. Al desplegable lateral creat anteriorment (*Navegació*) podrem prémer sobre sobre "Profile" (*Imatge 69*), el qual ens portarà al següent procés.

En la pantalla de visualització mostrarem tota la informació disponible de l'usuari (*Imatge 70*), excepte la contrasenya.



Imatge 69. Desplegable lateral (2)



Imatge 70. Visualització d'usuari

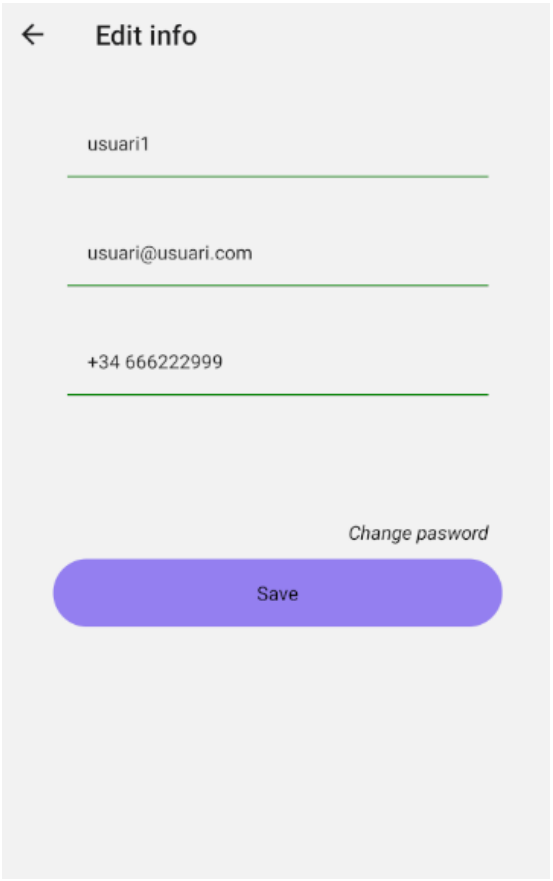
En aquest procés de visualització també tenim disponibles dos botons, un que ens porta al procés de "Modificació" i un altre que ens permet la desconnexió de l'usuari.

Modificació

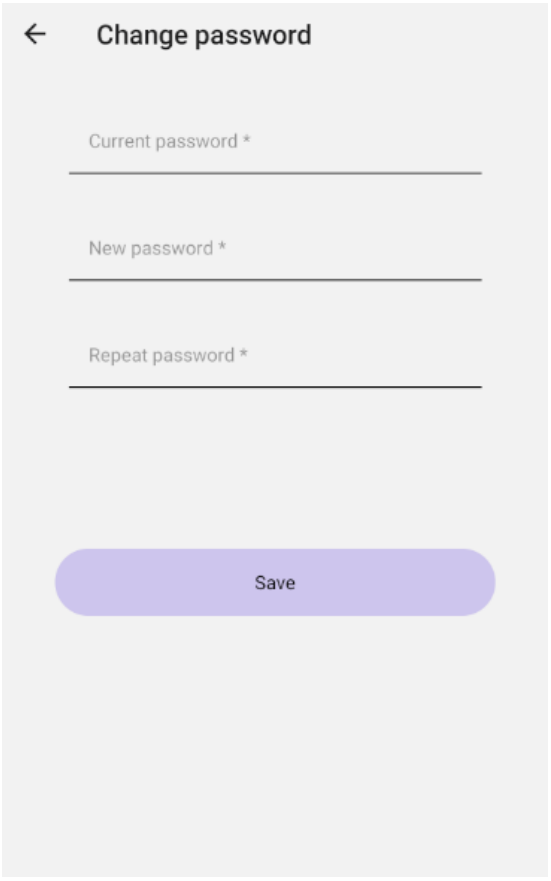
En cas de prémer el botó “Edit user”, ens apareixerà una nova pantalla que ens permetrà modificar la informació (*Imatge 71*).

Si volem modificar la informació, haurem de tenir en compte que tots els paràmetres siguin vàlids (camps de color verd) i prémer sobre el botó “Save”.

Si volem modificar la contrasenya, haurem de realitzar un pas més i prémer sobre “Change password” (Canvia contrasenya). Inicialment el botó estarà desactivat fins a emplenar tots els camps (*Imatge 72*). Un cop ingressada la contrasenya antiga, i la nova en els dos camps, s’activarà el botó i podrem guardar, “Save” (*Imatge 73*).

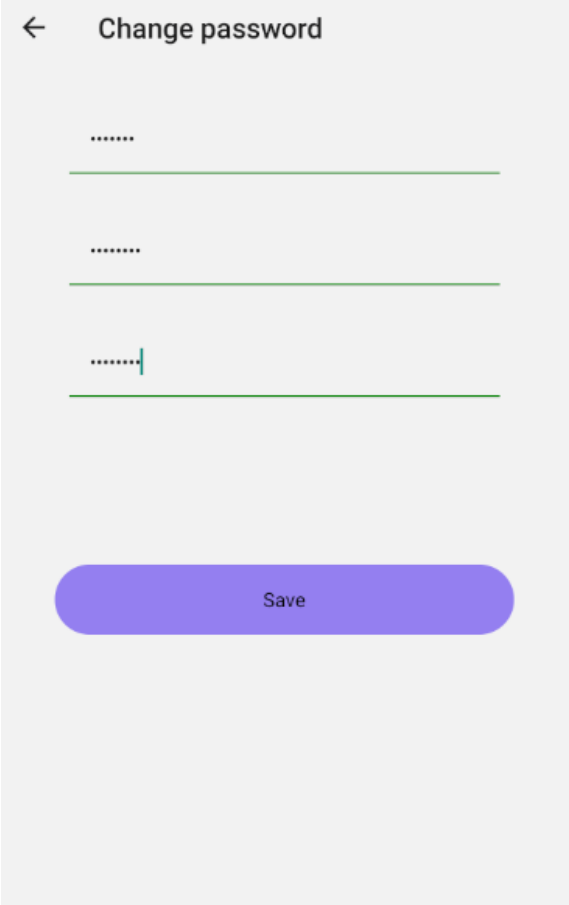


Imatge 71. Modificació usuari.

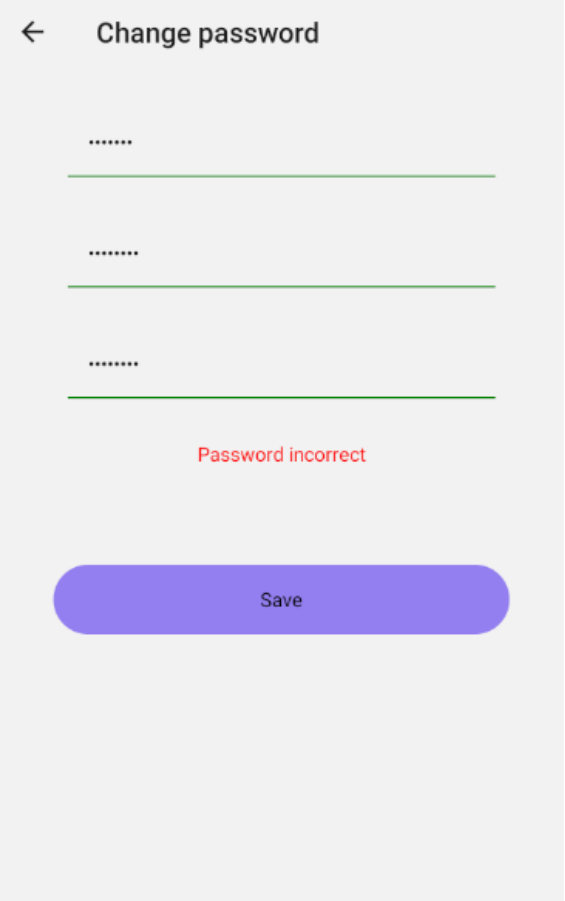


Imatge 72. Modificació contrasenya.

L'acció només es realitzarà correctament en cas d'ingressar la contrasenya actual correctament. En cas d'error, ens mostrarà un missatge (*Imatge 74*).



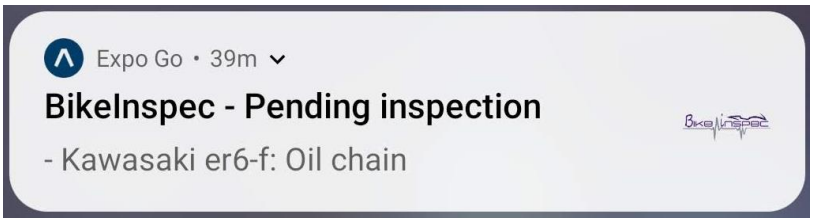
Imatge 73. Modificació contrasenya. Botó activat



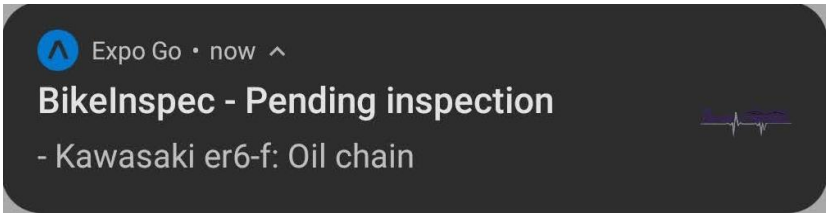
Imatge 74. Modificació contrasenya. Contrasenya incorrecta.

Notificacions

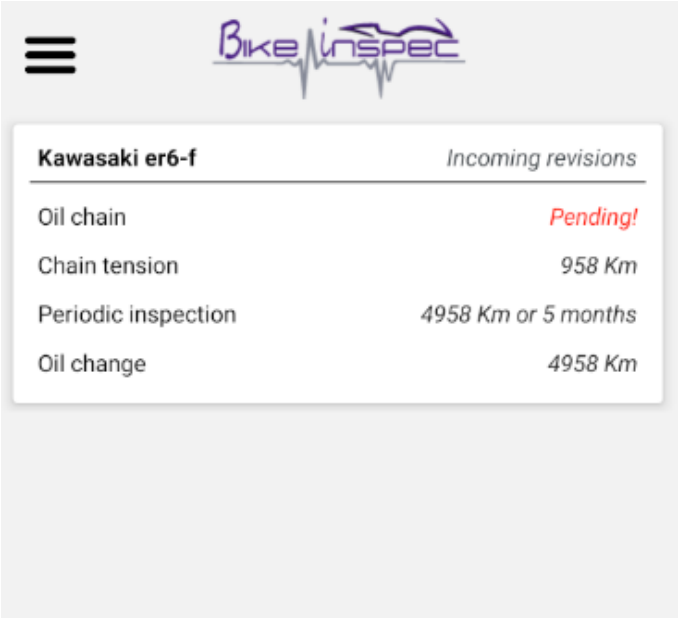
Per poder provar aquest mòdul, era necessari un dispositiu “real”. Vaig realitzar les proves amb el meu mòbil personal. Podem veure la notificació corresponent a continuació (*Imatge 75 i 76*), juntament amb la pantalla que ens apareix en obrir l’aplicació (*Imatge 77*).



Imatge 75. Push notification. Light.



Imatge 76. Push notification. Dark.

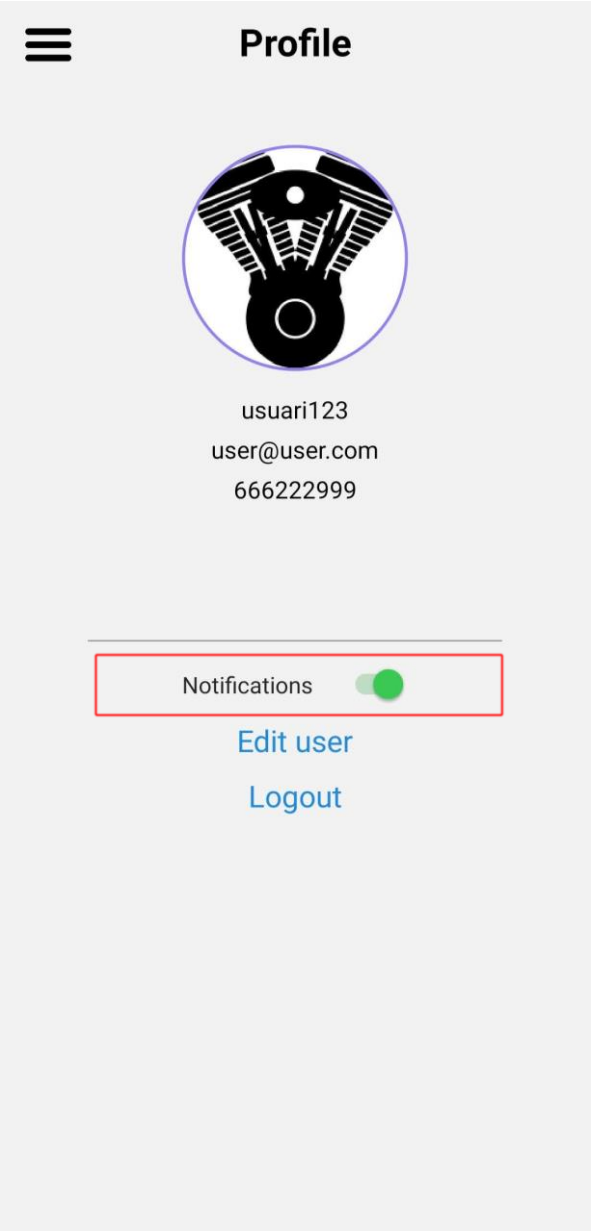


Imatge 77. Pantalla inici. Oberta per push notification.

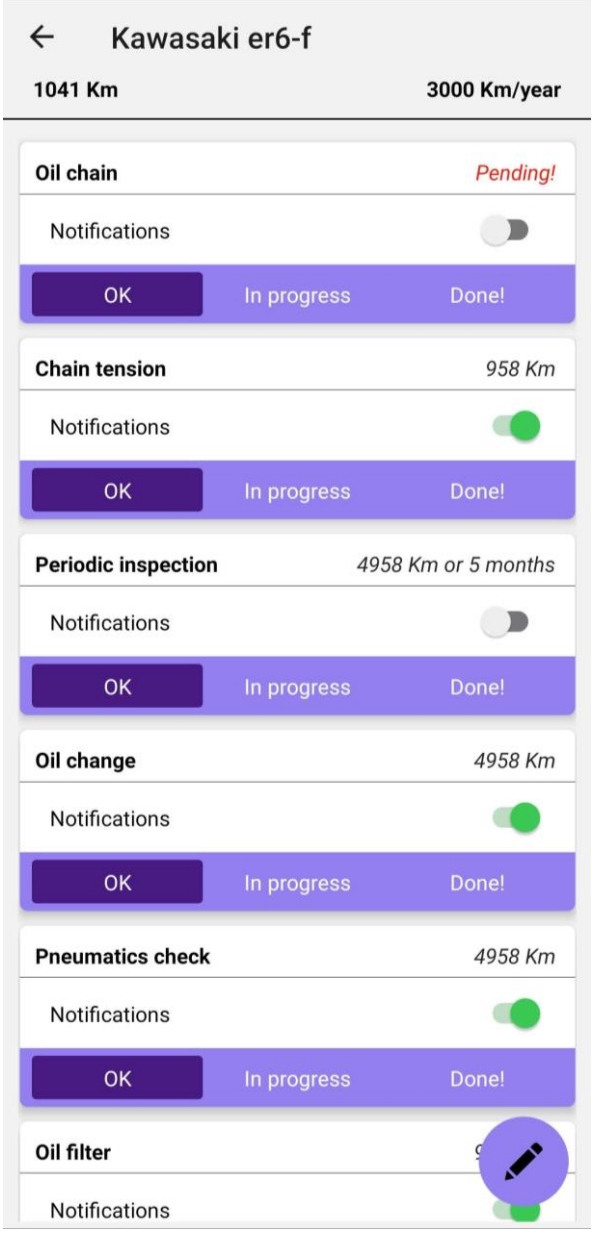
Activar / Desactivar

Per activar o desactivar les notificacions des de l'aplicació, hi ha dues opcions:

- Activar / desactivar **totes** les notificacions **d'un dispositiu** (*Imatge 78*).
- Activar / desactivar les notificacions **individualment de cada revisió** (*Imatge 79*).



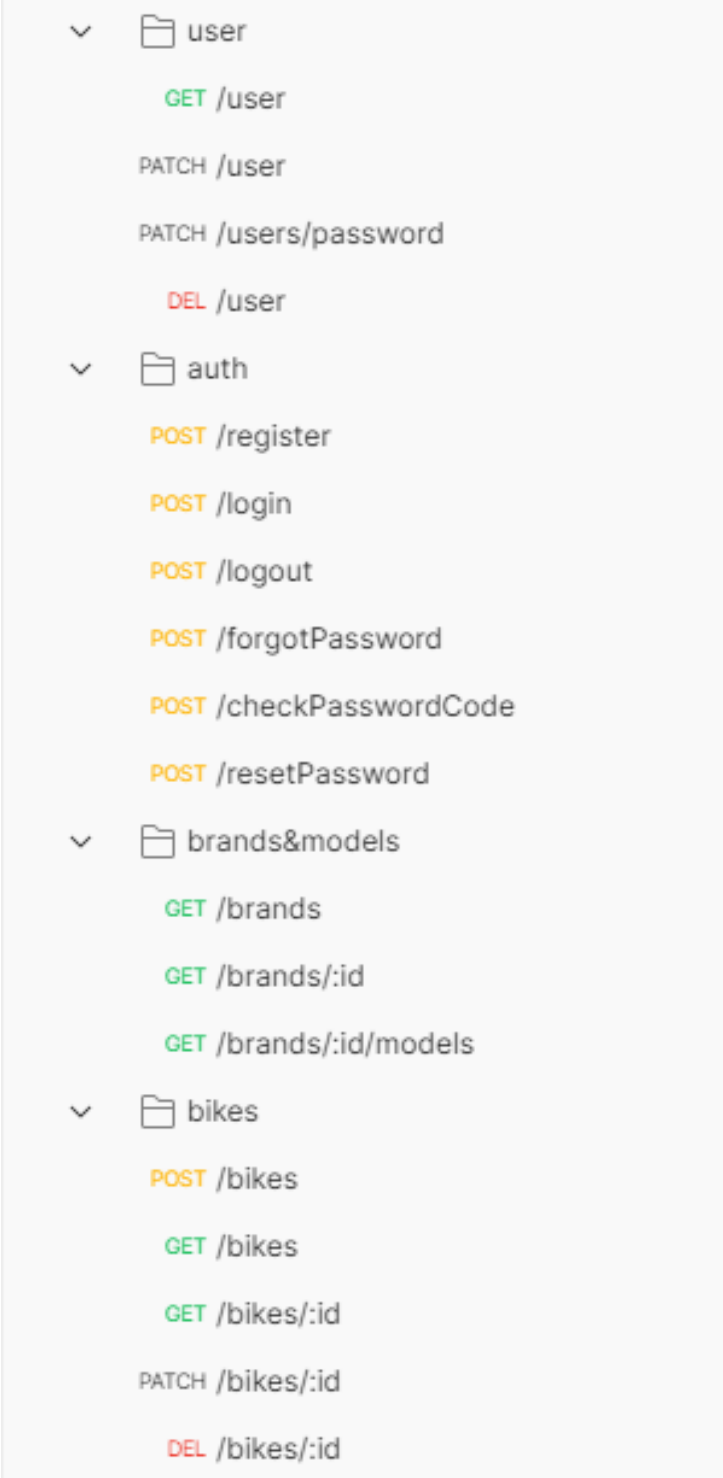
Imatge 78. Notification switch. Perfil



Imatge 79. Notification switch. Revisió

API

Al final del projecte ens trobem amb 18 peticions o rutes diferents. Es pot accedir a l'arxiu postman a través de l'enllaç a l'Annex *(Annex B. Links codi projecte*).



Imatge 80. Rutes POSTMAN final.

11. Conclusions

Com ja es comenta en l'apartat d'objectius, un dels punts més importants era l'aprenentatge de tecnologies. Gràcies a aquest projecte puc afirmar amb seguretat que em sento còmode i conec moltes eines i recursos de cada una de les tecnologies utilitzades:

- JavaScript
- Regex (Validacions d'usuari)
- JSON (Fitxers de configuració, base de dades i comunicacions API)
- Node.JS
- Expo CLI
- React
    - React Native
    - Redux
    - React Navigation v5
    - Expo push notifications

- MongoDB
- MongoDB Cloud
- Mongoose
- Cron
- Android Studio (Emulador dispositius Android)
- Postman
- Google Cloud

Algunes d'aquestes tecnologies ja les coneixia i hi havia tractat amb anterioritat (a classe o per comte pròpia):

- JavaScript, JSON, Node.JS, Postman, Android Studio.

En canvi, altres llibreries i tecnologies han suposat una inversió temporal d'aprenentatge més gran a causa de la meva **inexperiència**:

- **Regex, React Native, Redux, React Navigation v5, Expo, Expo push notifications, MongoDB, MongoDB Cloud, Mongoose, Cron, Google Cloud**.

A més, tot i no estar inicialment en els objectius, l'apartat d'**implantació del servidor** ha estat un punt molt important, tant pel gran ús que en fan les empreses avui en dia, com per la satisfacció de tenir una aplicació final utilitzable des de "qualsevol lloc" del món.

Aquest apartat d'aprenentatge ha ocupat un aproximat de **20-25% del treball**. He invertit unes 30 hores en l'aprenentatge inicial, i unes 30 hores al llarg de tot el desenvolupament i implantació. En aquest sentit considero els objectius d'aprenentatge amb un alt grau d'assoliment, tot i que sempre hi ha marge de millora.

De cara a l'aplicació, tots els requeriments s'han assolit.

Tot i el projecte estar finalitzat i que els objectius inicials s'han complert, considero que simplement l'aplicació, no està preparada pel públic per una raó molt concreta: **la distància recorreguda per cada motocicleta s'actualitza a través d'un paràmetre entrat per l'usuari**.

Aquesta aplicació és part d'un projecte a major escala, el qual s'explica en el següent apartat (*12. Treball futur*).

12. Treball futur

Hi ha moltes possibilitats d'ampliar i millorar el projecte. Personalment, aquesta aplicació i projecte forma part d'un projecte més gran en el qual poder controlar tots els paràmetres de la motocicleta des de la comoditat del smartphone.

El següent pas seria desenvolupar un aparell electrònic al qual connectar a la motocicleta. Aquest es comunicaria amb una API específica, la qual compartiria la Base de Dades amb l'actual projecte elaborat. D'aquesta manera, podríem accedir a la temperatura, quilometratge exacte, mitjana de kilòmetres diaris, rutes realitzades i moltes més característiques per a una millor experiència d'usuari.

I, finalment, m'agradaria desenvolupar la meva motocicleta personalitzada i integrar-hi aquesta aplicació, i tot el ventall de possibilitats que m'ofereix la tecnologia.

13. Bibliografia

"Amazon Web Services (AWS) - Cloud Computing Services." *Amazon Web Services, Inc.*, aws.amazon.com. Accessed 6 June 2021.

"Apache Cassandra." *Apache*, cassandra.apache.org. Accessed 8 June 2021.

"Apache CouchDB." *Apache*, couchdb.apache.org. Accessed 8 June 2021.

Arvindpdmn, GuruBhat. "Kotlin (Language)." *Devopedia*, 20 Jan. 2019, devopedia.org/kotlin-language.

---. "Kotlin (Language)." *Devopedia*, 20 Jan. 2019, devopedia.org/kotlin-language.

"Azure Cloud Computing Services." *Microsoft Azure*, azure.microsoft.com/en-us. Accessed 6 June 2021.

"Building a Node.Js App on App Engine." *Google Cloud*, 2021, cloud.google.com/appengine/docs/standard/nodejs/building-app.

"Cloud Computing, Hosting Services, and APIs." *Google Cloud*, cloud.google.com/gcp?utm_source=google&utm_medium=cpc&utm_campaign=emea-es-all-en-bkws-all-all-trial-e-gcp-1010042&utm_content=text-ad-none-any-DEV_c-CRE_500236788684-ADGP_Hybrid%20%7C%20BKWS%20-%20EXA%20%7C%20Txt%20%7E%20GCP%20%7E%20General%23v1-KWID_43700060384861654-aud-606988877934%3Akwd-6458750523-userloc_1005467&utm_term=KW_google%20cloud-NET_g-PLAC_&gclid=Cj0KCQjw5PGFBhC2ARIsAIFIMNchjrXLRl6OKoWpr7IRdeVM9HBGDHKbuYVs3AkSns6MEH0m-_DLMo0aAnoDEALw_wcB&gclsrc=aw.ds. Accessed 6 June 2021.

colaboradores de Wikipedia. "Base de datos relacional." *Wikipedia, la enciclopedia libre*, 18 Mar. 2021, es.wikipedia.org/wiki/Base_de_datos_relacional.

---. “Framework.” *Wikipedia, la enciclopedia libre*, 6 May 2021, es.wikipedia.org/wiki/Framework.

---. “MongoDB.” *Wikipedia, la enciclopedia libre*, 18 Mar. 2021, es.wikipedia.org/wiki/MongoDB.

“Cuánto Vale Mi Hora Como Freelance.” *Calculadora Freelance*, www.calculadorafreelance.com. Accessed 7 June 2021.

“Dart Overview.” *Dart*, 2021, dart.dev/overview.

“Data Model Design — MongoDB Manual.” *MongoDB*, 2021, docs.mongodb.com/manual/core/data-model-design.

“Deploying Your Web Service.” *Google Cloud*, 2021, cloud.google.com/appengine/docs/standard/nodejs/building-app/deploying-web-service.

Digite. “What Is Scrum Methodology? & Scrum Project Management.” *Digite*, 5 Nov. 2020, www.digite.com/agile/scrum-methodology/#:%7E:text=Scrum%20is%20an%20agile%20development,the%20development%20of%20the%20project.

“Expo, Logo Free Icon.” *Icons-Icons*, icon-icons.com/icon/expo-logo/145293. Accessed 8 June 2021.

“Firebase Cloud Messaging.” *Firebase*, firebase.google.com/docs/cloud-messaging. Accessed 8 June 2021.

“Flutter - Beautiful Native Apps in Record Time.” *Flutter*, 2021, flutter.dev.

“Flutter VS Xamarin VS React Native: What’s Best in 2021?” *EGO*, 2021, www.ego-cms.com/post/flutter-vs-xamarin-vs-react-native-whats-best-in-2021.

Fonts, 1001. “Suissnord Font ·.” *1001 Fonts*, 2021, www.1001fonts.com/suissnord-font.html.

“Getting Started – Pug.” *Pug*, pugjs.org/api/getting-started.html. Accessed 7 June 2021.

"HTTP Status Codes." *Restapitutorial*, www.restapitutorial.com/httpstatuscodes.html. Accessed 8 June 2021.

"Installing Google Cloud SDK | Cloud SDK Documentation." *Google Cloud*, 2021, cloud.google.com/sdk/docs/install.

"Java :: LeanXcale Documentation." *LeanXcale*, 2021, docs.leanxcale.com/leanxcale/1.6/development/java/index.html.

"JavaScript : Email Validation." *W3resource*, 9 Sept. 2020, www.w3resource.com/javascript/form/email-validation.php.

Kissflow. "6 Essential Questions to Understand Rapid Application Development Methodology." *Kissflow*, 27 May 2021, kissflow.com/low-code/rad/rapid-application-development-methodology-essentials/#:%7E:text=RAD%20methodology%20is%20a%20software,the%20initial%20development%20is%20complete.&text=Rapid%20Application%20Development%20has%20changed%20the%20trend%20of%20how%20software%20is%20developed.

---. "Rapid Application Development (RAD): Changing How Developers Work." *Kissflow*, 15 May 2021, kissflow.com/low-code/rad/rapid-application-development/#pros-and-cons.

"Lenguajes de programación para Android | Blog Deusto Formación." *Blog Deusto Formación*, 2021, www.deustoformacion.com/blog/apps-moviles/lenguajes-para-programar-aplicaciones-android.

Maldini, Mattia. "Functional Bits in Flutter - Flutter Community." *Medium*, 30 Nov. 2020, medium.com/flutter-community/functional-bits-in-flutter-dc0881a99161.

"Meet Android Studio |." *Android Developers*, 2021, developer.android.com/studio/intro.

Microsoft. “What Is Xamarin? | .NET.” *Microsoft*, 2021, dotnet.microsoft.com/learn/xamarin/what-is-xamarin.

MongoDB. “MongoDB Atlas: Cloud Document Database.” *MongoDB*, 2021, www.mongodb.com/cloud/atlas/lp/try2?utm_source=google&utm_campaign=gs_emea_spain_search_core_brand_atlas_desktop&utm_term=mongodb%20atlas&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=12212624563&gclid=Cj0KCQjwnueFBhChARIsAPu3YkTpUNZuFgBu9t1otJbVQoOQRuqPSi2fObiRREnyjMZT19S7TCYha-oaAkOJEALw_wcB.

“MySQL.” *MySQL*, www.mysql.com. Accessed 8 June 2021.

“Npm: Cron.” *Npm*, 24 Jan. 2020, www.npmjs.com/package/cron.

“Npm: Express-Mailer.” *Npm*, 17 July 2016, www.npmjs.com/package/express-mailer.

“Oracle SQL.” *Oracle*, www.oracle.com/es/database/technologies/appdev/sqldeveloper-landing.html. Accessed 8 June 2021.

“Oxford Languages and Google - Spanish | Oxford Languages.” *Oxford Languages*, 10 Mar. 2021, languages.oup.com/google-dictionary-es.

“Postman | The Collaboration Platform for API Development.” *Postman*, 2021, www.postman.com.

“Push Notifications Overview.” *Expo Documentation*, docs.expo.io/push-notifications/overview. Accessed 8 June 2021.

Rachele, Warren. “Ejemplos de programas para la administración de bases de datos.” *Techlandia*, 20 Nov. 2017, techlandia.com/ejemplos-programas-administracion-bases-datos-lista_126285.

“React Native · Learn Once, Write Anywhere.” *React Native*, 2021, reactnative.dev.

“Register MongoDB Cloud.” *MongoDB Cloud*, account.mongodb.com/account/register. Accessed 6 June 2021.

Rodríguez, Santiago Porras. “Introducción al desarrollo de Apps Multiplataforma con Xamarin.Forms.” *Piensa en software, desarrolla en colores*, 19 Oct. 2015, blogs.encamina.com/piensa-en-software-desarrolla-en-colores/desarrollando-apps-multiplataforma-con-xamarin.

Sierra, Yhorman. “Aplicaciones híbridas: qué son, frameworks, ejemplos y ventajas.” *#ADNCLOUD*, 30 July 2019, blog.mdcloud.es/aplicaciones-hibridas-frameworks-ejemplos-y-ventajas/#:%7E:text=Las%20aplicaciones%20h%C3%ADbridas%2C%20a%20diferencia,de%20su%20marca%20o%20fabricante.

Simmons, T. “Rapid Application Development: What It Is and What’s Next.” *Plutora*, 12 Nov. 2020, www.plutora.com/blog/rapid-application-development-what-it-is-and-whats-next.

Singh, Ankita. “What Is Rapid Application Development (RAD)?” *Capterra*, 6 Dec. 2019, blog.capterra.com/what-is-rapid-application-development/#:%7E:text=Rapid%20Application%20Development%20(RAD)%20is,strict%20planning%20and%20requirements%20recording.

SPEC INDIA. “Kotlin vs React Native: Which Is Best For Cross-Platform App Development.” *SPEC INDIA*, 5 Nov. 2020, www.spec-india.com/blog/kotlin-vs-react-native#:%7E:text=React%20Native%20is%20faster%20and,both%20Android%20and%20iOS%20platforms.&text=Kotlin%2C%20on%20the%20other%20side,logic%20already%20written%20in%20Kotlin.

Tema2ReqAnal | Enginyeria Del Software I. UdG | Enginyeria del Software I, 2014, moodle36.udg.edu/pluginfile.php/918164/mod_resource/content/3/Tema2ReqAnal.pdf.

"What Is a REST API?" *Red Hat*, www.redhat.com/en/topics/api/what-is-a-rest-api. Accessed 7 June 2021.

"What Is an API? (Application Programming Interface)." *MuleSoft*, 2021, www.mulesoft.com/resources/api/what-is-an-api#:%7E:text=API%20is%20the%20acronym%20for,to%20talk%20to%20each%20other.

"What Is the Difference between Expo CLI and React Native CLI?" *Stack Overflow*, 25 Feb. 2019, stackoverflow.com/questions/54862388/what-is-the-difference-between-expo-cli-and-react-native-cli.

Wikipedia contributors. "Client–Server Model." *Wikipedia*, 7 June 2021, en.wikipedia.org/wiki/Client%E2%80%93server_model.

---. "Flutter (Software)." *Wikipedia*, 2 June 2021, en.wikipedia.org/wiki/Flutter_(software).

---. "Java (Programming Language)." *Wikipedia*, 24 May 2021, en.wikipedia.org/wiki/Java_(programming_language).

---. "Kotlin (Programming Language)." *Wikipedia*, 15 May 2021, en.wikipedia.org/wiki/Kotlin_(programming_language).

---. "Microsoft Access." *Wikipedia*, 6 June 2021, en.wikipedia.org/wiki/Microsoft_Access.

---. "React Native." *Wikipedia*, 22 May 2021, en.wikipedia.org/wiki/React_Native.

---. "Vehicle Inspection." *Wikipedia*, 3 June 2021, en.wikipedia.org/wiki/Vehicle_inspection#:%7E:text=The%20inspection%20includes%20checking%20of,and%20presence%20of%20mandatory%20equipment.

14. Annexos

Annex A. Dependències i llibreries Node.JS

Front-end

```
"@firebase/messaging": "^0.6.5",
"@react-native-community/async-storage": "^1.12.1",
"@react-native-community/masked-view": "0.1.10",
"@react-native-community/picker": "^1.8.1",
"@react-navigation/drawer": "^5.12.4",
"@react-navigation/native": "^5.9.4",
"@react-navigation/stack": "^5.14.3",
"axios": "^0.21.1",
"expo": "~40.0.0",
"expo-constants": "^9.3.5",
"expo-notifications": "^0.8.2",
"expo-permissions": "^10.0.0",
"expo-status-bar": "~1.0.3",
"firebase": "7.9.0",
"react": "16.13.1",
"react-dom": "16.13.1",
"react-native": "https://github.com/expo/react-native/archive/sdk-40.0.1.tar.gz",
"react-native-appearance": "^0.3.4",
"react-native-elements": "^3.3.2",
"react-native-firebase": "^5.6.0",
"react-native-gesture-handler": "~1.8.0",
"react-native-input-validator": "^1.0.12",
"react-native-push-notification": "^7.2.3",
"react-native-reanimated": "~1.13.0",
"react-native-safe-area-context": "3.1.9",
"react-native-screens": "^2.15.2",
"react-native-web": "~0.13.12",
"react-redux": "^7.2.3",
"redux": "^4.0.5",
"redux-persist": "^6.0.0",
"redux-thunk": "^2.3.0",
"uuid": "^3.4.0"
"@babel/core": "^7.9.0"
```

Back-end

```
"bcryptjs": "^2.4.3",
"cors": "^2.8.5",
"dotenv": "^8.2.0",
"express": "^4.17.1",
"express-mailer": "^0.3.1",
"firebase-admin": "^9.8.0",
"joi": "^17.4.0",
"jsonwebtoken": "^8.5.1",
"mongoose": "^5.12.2",
"morgan": "^1.10.0",
"multer": "^1.4.2",
"node-cron": "^3.0.0",
"node-fetch": "^2.6.1",
"nodemon": "^2.0.7",
"passport": "^0.4.1",
"pug": "^3.0.2"
```

Annex B. Links codi projecte

Aplicació / Front-end: https://github.com/LukeP13/BikeInspectioner_ReactNative

Back-end: https://github.com/LukeP13/TFG_apirest

Postman: https://www.getpostman.com/collections/99b61f4cb72c32914539