1. Desenvolupament de programari

1.1 Tecnologies utilitzades plantejament inicial

La primera tecnología utilizada va ser python tkinter.

Python Tkinter: És un binding de la biblioteca gràfica Tcl / Tk per al llenguatge de programació Python. Es considera un estàndard per a la interfície gràfica d'usuari per Python i és el que ve per defecte amb la instal·lació per a Microsoft Windows.

1.2. Tecnologies utilitzades actualment

Electron: Electron.js és una plataforma per desenvolupar aplicacions d'escriptori utilitzan tecnologies web (HTML, CSS i JavaScript) creada i mantinguda per Github.

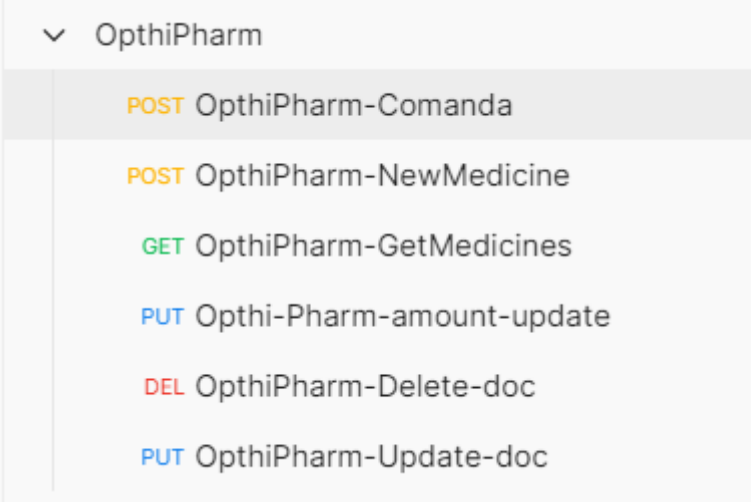
Node.js: És un entorn en temps d'execució multiplataforma, de codi obert, per a la capa servidor basat en el llenguatge de programació JavaScript, asíncron, amb E / S de dades en una arquitectura orientada a objectes i basat en el motor V8 de Google.

Firebase: Firebase és una plataforma per al desenvolupament d'aplicacions web i aplicacions mòbils llançada el 2011 i adquirida per Google el 2014. L'utilitza'm per guardar les dades de l'app. (Base de dades). Backend a as service (BaaS).

1.3. Programes utilitzats per el testing i desenvolupament

Postman: És una eina que s'utilitza, sobretot, per al testing d'API REST, encara que també admet altres funcionalitats que se surten del que engloba el testing d'aquest tipus de sistemes.

Rutes creades en el programa per realitzar proves:



Git: És un programari de control de versions dissenyat per Linus Torvalds, pensant en l'eficiència, la fiabilitat i compatibilitat de manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font.

L'utilitzem per exactament tindre un control de versions i de canvis de la app, on després la publiquem a Github.

Github: És una forja per allotjar projectes utilitzant el sistema de control de versions Git. S'utilitza principalment per a la creació de codi font de programes d'ordinador. El programari que opera GitHub va ser escrit en Ruby on Rails. Des de gener de 2010, GitHub opera sota el nom de GitHub, Inc.

Nosaltres l'utilitzem per tindre un historial de version i canvis, fins a tindre un seguiment del programa apart de poder compartir el projecte per a tots i així tenim la disposició de descarregar-lo des de qualsevol ordinador.

Sempre es poden consultar les modificacions del programa en aquesta direcció: https://github.com/Serlomo/OpthiPharm

Les primeres versions i cambis de programa es van realitza en un altre repositori, pero estan explicas més endavant.

1.4 Divisió del programari segons aplicació

La divisió del programa es una estructura que s'utilitza per les aplicacions web com per les aplicacions de movil, i està dividit en 3 parts.

Primerament disposem del front-end, és on es desenvolupa tot el que l'usuari pot veure i interactuar amb diferents components, en aquest cas seria l'aplicació d'escriptori (Interfície de l'usuari).

El back-end és on es desenvolupa l'has sortides i el diferent algoritme per poder executar les sortides, a part que també és un bypass entre la part del client (Front-end, Aplicació d'escriptori) i la connexió de la base de dades.

La base de dades és on es guarden totes les dades que genera l'app.

1.5 Comunicació entre les parts

Com havíem mencionat mes adalt, necesitarem una comunicació entre el front-end (Aplicació d'escriptori) back-end (Rasberry Pi) i Base de dades.

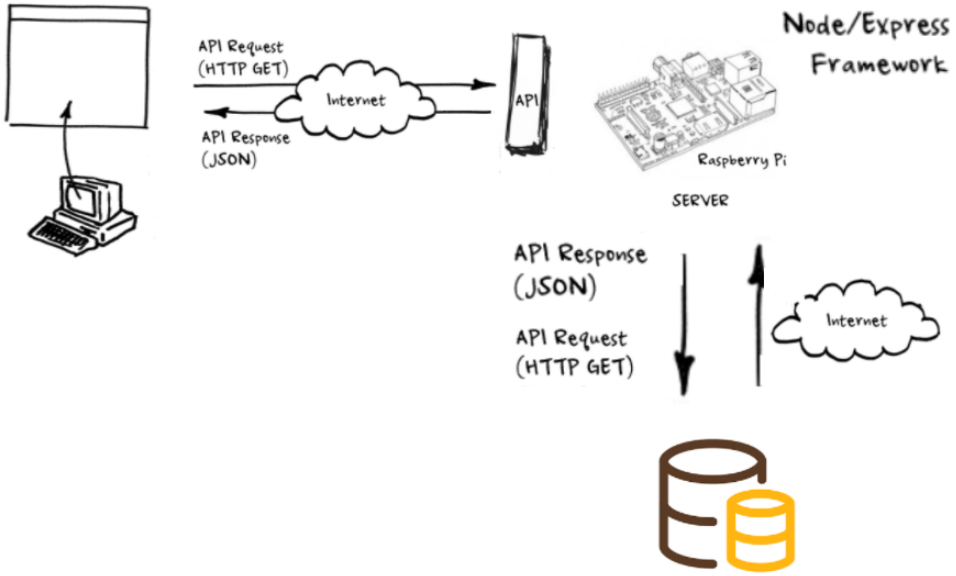
Principalment vam estar investiga'n diferents comunicacions, i havíem d'escollir entre dues comunicacions. MQTT o API REST. Cada una de les comunicacions funcionen diferent i estan pensades per diferents aplicatius, MQTT està destinada a dispositius IOT perquè aquesta utilitza una comunicació de web sockets que això són subscripcions a una comunicació, ja que si tenim un sensor en el dispositiu podem enviar les dades al moment, podríem dir que seria en temps real. A part que aquesta consumeix menys amplada de banda. Per a tot això és la millor per dispositius IOT. Però nosaltres no necessitem està tota

l'estona escoltant al back-end. Llavors arriba la comunicació API REST. Aquesta s'utilitza principalment per recuperar dades d'una bd o fer una sol·licitud a un determinat servei. Podem dir que el back-end estaria esperen a què algú actives aquest servei i llavors executaria el que està en aquell codi.

Per la nostra aplicació hem escollit l'API REST, bàsicament perquè només necessitem una comunicació amb el backend en el moment que nosaltres li diem d'executar el codi i per si en el futur es vol fer una base de dades al núvol, com més endavant expliquem perquè vam haver de fer la base de dades al núvol.

Principalment la comunicació es fa per peticions HTTP amb objectes json.

En aquest esquema podem trobar com es comunicar tot.



Com podem comprovar en la foto anterior, trobem que l'aplicació d'escriptori es comunica amb la rasberry pi atraves de peticions HTTP, aquesta poden ser PUT, GET, POST, DELETE aquest només defineixen l'acció que es realitza amb aquest endpoint. També trobem que la base de dades es comunica de la mateixa forma.

1.6. Referències i recursos (links)

Cursos realitzats per crear aquesta aplicació

https://www.youtube.com/watch?v=vDrz0OUook4&list=PL2PZw96yQChzi-lLw6rqqAPRkNXSOeqtr&ab_channel=JohnOrtizOrdo%C3%B1ez

Videos utilitzats

https://www.youtube.com/watch?v=0BWzZ6c8z-g&t=1321s

https://www.youtube.com/watch?v=D67Liy5C86s&t=3059s&ab_channel=FaztCodeFaztCode

https://www.youtube.com/watch?v=hAjltO1uw-A&t=1640s&ab_channel=FaztCodeFaztCode

https://www.youtube.com/watch?v=FiiwaJUS84Q&t=2587s&ab_channel=FaztCodeFaztCode

https://www.youtube.com/watch?v=zH8ilfplP7Y&t=1258s&ab_channel=AldoCaamalAldoCaamal

https://www.youtube.com/watch?v=b6KJ7FSMifw&ab_channel=FaztCodeFaztCode

https://www.youtube.com/watch?v=794Q71KVw1k&ab_channel=FaztFazt

2. Front End

2.1. Disseny inicial

El disseny inicial es va realitzar amb una aplicació de disseny que s'anomena adobe XD. És una aplicació per crear dissenys interactius. Ens va servir per poder donar una idea de com queriam i que requiere el software.

En la carpeta Disseny, dintre de la carpeta Programa, podem trobar el archiu XD.

Per poder obrir el arhiu sa de descarregar el programa que es pot trobar en aquets link: https://helpx.adobe.com/es/xd/get-started.html

2.2. Dependències del programa

Podem trobar els paquets utilitzats en els archiu package.json en l'apartat de dependències.

electron-fetch → Paquet per cridar a la API REST del servidor: https://www.npmjs.com/package/electron-fetch

electron-reload → Es un paquet només per quan estàs desarrollan, per no tindre de reinicia el server sempre que fas cambis en el codi: https://www.npmjs.com/package/electron-reload

mdb-ui-kit → Es un kit per tindre components creats amb l'estetica precreada: https://www.npmjs.com/package/mdb-ui-kit

electron-reload → És per crear un executable per l'aplicació d'escriptori: https://www.npmjs.com/package/electron-reload

2.3 Idees a desenvolupar

La part del client principalment el que volem aconseguir es poder crear una comanda a partir d'una aplicació d'escriptori. Que aquesta actives les sortides de la Raspberry pi.

Poder registra, modificar i eliminar medicaments des de l'aplicació d'escriptori. Tindre una pantalla des de l'aplicació d'escriptori on poguessis veure l'estat de la comunicacio de la maquina.

2.4. Components interactius

Menú de navegació

Trobarem un menú a la part superior on podem anar a diferents pàgines de l'app.

Aquesta barra d'eines té els diferents enllaços on podem navegar.

- Comanda
- Gestió de BD
- Estat de la máquina

Página comanda

En aquesta pàgina trobareu la búsqueda dels medicaments i donar li l'ordre per fer la comanda.

Input de cerca: Aquest es tracta de un buscador amb un autocompletat per poder trobar els medicament amb més facilitat, quan vas escribit et sortira el nom del medicament més proper que estigui registrat a la base de dades.

Tarjeta del medicament seleccionat: Aquesta tarjeta apareix després de buscar el medicament, aquesta surt tota la informació registrada previmanet del medicament. Dintre d'aquesta targeta tenim dos botons, un és per tancar la tarjeta i l'altre és per afegir la comanda el medicament de tarjeta.

Input de quantitat: Aquest input el que fa és donar la oportunitat de decidir quan medicament del mateix volem dispensar.

Taula comanda: Aquesta és una taula que s'emplena sobre els medicament seleccionats amb la tarjeta. En la taula ens trobem el nom, descripció, preu, quantitat de cada medicament seleccionat. En cada medicament ens trobarem un botó que diu: Eliminar de la comanda i aquest el que farà sera borrar el medicament de la taula de comanda.

Página registrar medicaments

En aquesta pàgina podrem afegir, esborrar i modificar els medicaments, és on es poden relacionar les files i les columnes als medicaments. Aquest s'afegiran a la base de dades.

Formulari de crear: Aquest formulari és per crear els medicaments i poder modificarlos, per crear un de nou hem d'introduir totes les dades del medicament i prémer el botó Crear nou medicament.

Taula de registre de base de dades: En aquesta taula el que trobarem seran tots els medicaments que estan registrats a la base de dades, amb tota la informació de cada un. A la part esquerra de la taula podem trobar dos botons que sortiran per cada fila de la taula. El que diu Eliminar, el que farà serà eliminar aquell medicament d'aquella fila a la base de dades i el botó que diu Modificar, el que farà és sortir una targeta a sota del formulari, que avisarà que en el formulari esta habilitat per poder modificar aquell medicament. Llavors emplenes els camps que vols modificar i premem el botó Modificar el medicament. Llavors el que trobarem és que modificarem aquell medicament seleccionat. Si prenem l'altre botó de la targeta que diu Cancel·lar modificació, el que farem serà deshabilitar el formulari per editar el medicament.

Página estat de la máquina

En aquesta pàgina podrem veure quin estat es troba la màquina, si està connectada i funciona correctament.

Disposem d'un quadre que simula un led. Si aquest està verd vol dir que la comunicació està establerta i podem interactuar amb la màquina. Si el led en surt de color vermell, voldrà dir que el sistema no està comunicat.

2.5. Històric de versions i canvis

Sempre es poden consultar les modificacions del programa en aquesta direcció: https://github.com/Serlomo/OpthiPharm on esta el botó de descarregar el codi a sota fica commits, si entres es veu un cronograma dels diferents canvis, fins i tot pots descarregar el codi d'aquella modificació.

El mes de desembre del 2020 → Creació del esbós de disseny del programa.

El primer programa que es va crear va ser el dia 14 de gener del 2021 → Aquest estava creat amb python i el que realitza es buscar dintre d'un document json, els noms dels medicaments a partir d'un input. Podem trobar aquest programa al GitHub en la carpeta Primeres versions.

El dia 18 de gener del 2021 → Ens vam donar conta que podiem crear el programa de formes diferents, vam estar investigan de nou.

El dia 23 de gener del 2021 → Vam començar a mirar

El dia 2 de febre del 2021 → Es va començar el programa que ara mateix tenim. Es va crear el primer executable amb Electron.js.

La setmana del 8 de febre del 2021 → Es va començar a crear la navegació, aquesta primer va ser amb la pròpia navegació d'electron i crea'n noves pestanyes, però al final de setmana va ser a partir d'html.

La setmana del 15 de febre del 2021 → Estructura de l'html en la pàgina Comanda. Creació de la base de dades en local.

La setmana del 22 de febre del 2021 → Bug de la base de dades en local.

El dia 3 de març del 2021 →És crear el primer repositori de GitHub.

La setmana 12 d'abril del 2021 → Canvi en els paquets de disseny de bootstrap 5, eliminació dels arxius js i css del paquet i reestructurar totes les carpetes.

El dia 12 de maig del 2021 → Creació de fetch i creació de la pàgina Gestió db.

El dia 21 de maig del 2021 → Creació de la pàgina Estat de la màquina.

El dia 25 de maig del 2021 →Millora estètica a la pàgina comanda.

El dia 27 de maig del 2021 →Millora de comentaris en el codi i actualització de components.

3. Back End

3.1. Funcions que ha de dur a terme

La funció principal que ha de fer el backend es la comunicació entre l'aplicació d'escriptori

En el programa a la carpeta **Backend-OptiPharm/server/route** trobem les rutes de la base de dades en l'arxiu Firebase i l'arxiu matriu, tovem tota la lògica per poder dispensar els medicaments que la màquina requereix.

En l'arxiu Firebase trobem les següents funcions:

Ruta **/new-medicine** → És per registrar un nou medicament a la base de dades.

Ruta **/get-medicines** → És per obtenir tots els medicaments que hi ha a la base de dades. L'utilitzem sempre que es carreguen qualsevol de les pàgines, ja que aquesta necessiten les dades pel seu correcte funcionament.

Ruta **/delete-medicines** → És per eliminar un medicament en concret de la base de dades. S'utilitza en la pàgina Registre d'un nou medicament per esborrar el medicament seleccionat.

Ruta **/amount-update** → És per actualitzar l'estoc del producte que sa dispensat. S'utilitza sempre que s'executi una comanda, ja que restarà el valor de l'estoc en la base de dades del producte pertinent. (Utilitzem aquesta forma de saber l'estoc de la màquina per no haver de ficar sensors i encari la màquina).

Ruta **/update-doc** → És per actualitzar un medicament de la base de dades. Aquesta s'utilitza per actualitzar un medicament a la pàgina Registre d'un nou medicament.

En l'arxiu matriu ens trobarem la ruta **/comanda** → Aquest conte tot el codi per processar la comanda que l'envien. Les dades que rep són en forme de dos arrays que un son las files i l'altre són les columnes.

3.2. Arxiu matriu.js

Aquest arxiu es troba tota la lògica que fa activar les sortides de la Raspberry pi. La seqüència del programa està creat amb una funció tipus notbloking, que això vol dir que s'executaran a la mateixa vegada que les altres sortides, no és seqüencial sinó asíncron. S'utilitza per si en una fila trobem 3 columnes que han de dispensar a la vegada. En les columnes si hi ha més d'un medicament a dispensar, entre cada un passarà 2 segons en activar-se un altre cop.

Exactament la seqüència del programa és: Sempre comencem per la primera fila i quan acabem de dispensar tots el medicament de la fila passem a la següent fila. Fins a completar tota la seqüència.

3.3. Dependències del programa

Podem trobar els paquets utilitzats en els archiu package.json en l'apartat de dependències.

express → Paquet per crear el servidor en node.js:

https://www.npmjs.com/package/express

firebase-admin → Es un paquet que permet realitzar diferents funcions de firebase: https://www.npmjs.com/package/firebase-admin

morgan → Paquet per poder obtenir les dades del client en les diferents rutes de las API REST: https://www.npmjs.com/package/morgan

pm2 → Paquet per que sempre que s'encengui la raspberry pi executi el programa del backend: https://www.npmjs.com/package/pm2

En la raspberry pi tenim el sistema operatiu raspbian que ens el podem descarrega en aquesta pagina: https://www.raspberrypi.org/software/

3.4. Històric de versions i canvis

Sempre es poden consultar les modificacions del programa en aquesta direcció: https://github.com/Serlomo/OpthiPharm on està el botó de descarregar el codi a sota fica commits, si entres es veu un cronograma dels diferents canvis, fins i tot pots descarregar el codi d'aquella modificació.

El primer programa que es va crear va ser el dia 14 de gener del 2021 → Aquest estava creat amb python i el que realitza és buscar dintre d'un document json, els noms dels medicaments a partir d'un input. Podem trobar aquest programa al GitHub en la carpeta Primeres versions.

El dia 18 de gener del 2021 → Ens vam donar conta que podíem crear el programa de formes diferents, vam estar investiga'n de nou.

El dia 23 de gener del 2021 → Vam començar a mirar

La setmana del 22 de febre del 2021 → Es va començar a crear un backend amb python, comprovar les sortides com funcionaven amb una placa de proves i led's.

El dia 3 de març del 2021 →És crear el primer repositori de Github.

La setmana del 8 de març del 2021 → Ens vam donar conta que també es podria crear un servidor amb node.js i, ja que la majoria del codi està escrit en JavaScript vam optar per canviar de tecnologia.

La setmana del 15 de març del 2021 → Creació del servidor amb node.js, comprovar el paquet de sortides com funciona amb una placa de proves i led's.

La setmana del 22 de març del 2021 → Creació del servidor amb node.js, comprovar el paquet de sortides com funciona amb una placa de proves i led's.

La setmana 5 d'abril del 2021 →Creació de la ruta API Matriu.

La setmana 12 d'abril del 2021 →Bug API Matriu.

La setmana 19 de maig del 2021 → Canvi de la base de dades i creació de Firebase.

La setmana 26 de maig del 2021 → Creació de les rutes de la base de dades i comentaris en el codi.

El dia 27 de maig del 2021 →Millora de comentaris en el codi.

4. Database

4.1. Plantejament inicial

La primera vegada que vam plantejar la base de dades es volia fer una base de dades interna en el mateix Frontend.

Aquest és una base de dades integrada en un arxiu que seria en forma d'objectes json. Vam escollir primer aquesta perquè és molt semblen a com es demanen les peticions en una base de dades real. Ja que en aquell moment es pensava si en un futur és necessites d'una base de dades real al núvol.

El problema és que vam tindre problemes de versions i al final donava error, no podríem obtenir les dades. Llavors ja que teníem tota l'estructura creada, vam optar per crear una base de dades al núvol. Aquesta és un servei de Google que es diu Firebase, mencionat anteriorment.

4.2. Database externa

En aquesta base de dades ens podem connectar des d'on vulguem a partir d'unes credencials que estan un arxiu del codi. Les dades es guarden en una taula que conte tots els medicaments registrats per l'usuari.

No podem accedir al compte, ja que l'aplicació està registrada amb un mail d'un membre del grup i no podem fer el canvi a un altre correu. Per això deixarem un vídeo penjat en el Google Drive on es veurà la base de dades en aquest ellaç: https://drive.google.com/drive/folders/1x81dY5HEqpEwfjIqgh5ROZs5g4t9aKgB?usp=sharing

El link de la base de dades es aquet: https://opthi-pharm-default-rtdb.firebaseio.com/

Captura de pantalla de la base de dades

5. Executar el programa

Recomanem que s'insta-li Visual Code, per poder entendre millor el codi i poder entendre els comentaris. Podeu descarregar-lo en aquest enllaç: https://code.visualstudio.com/download

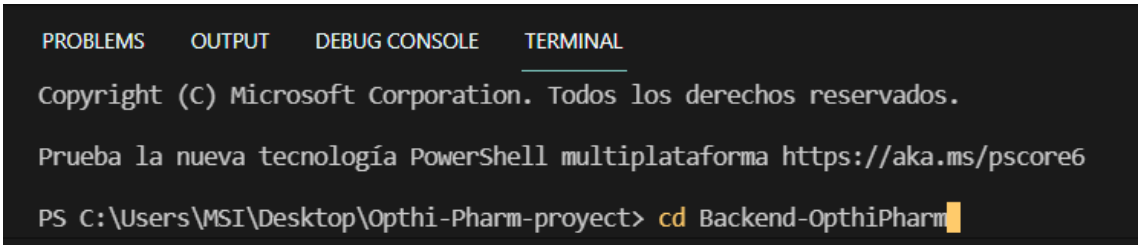
Primer de tot hem de baixar el programa, el podem trobar tant en el pen entregat com a Git Hub en aquest link: https://github.com/Serlomo/OpthiPharm

En el Git Hub està explicat com baixar se'l programa per poder fer modificacions i contribuir en el projecte. Però el que hem de fer és baixar-nos en format zip. Que es mostra a dalt a l'esquera "code" i seleccionar Dowload Zip.

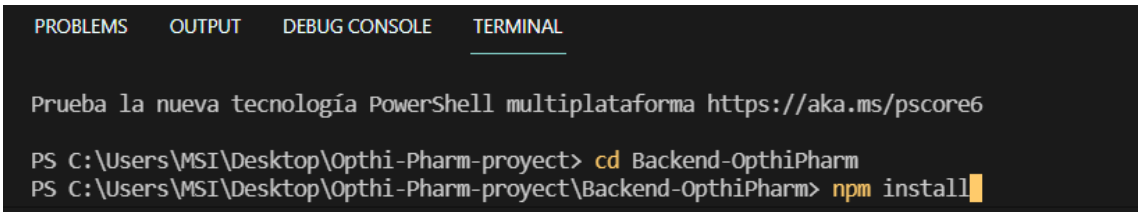
Quan tinguem descarregat hem d'instal·lar totes les dependències que necessita aquest projecta, ja que no és puja'n a la núvol perquè ocuparia més espai.

Primer de tot el que farem serà dirigir-nos a la carpeta Backend-OpthiPharm amb la cmd. En la cmd per poder-nos moure hem d'executar els següents codis:

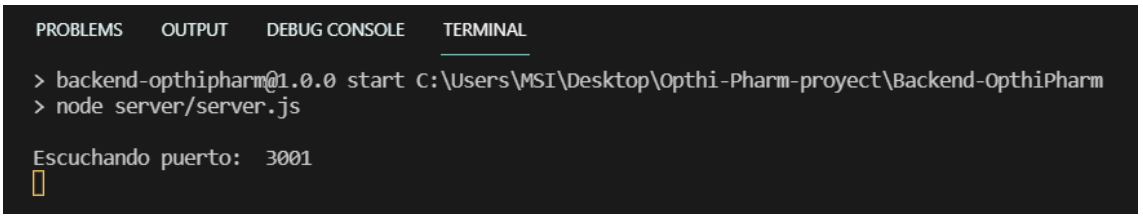
cd "Nom de la carpeta" → Ens dirigim a la carpeta escrita.
cd .. → Tornar endarrere.



Quan siguem al directori Backen-OpthiPharm el que hem de fer és executar npm instal, aquests el que farà serà descarregar totes les dependències del programa explicades mes endarrere.

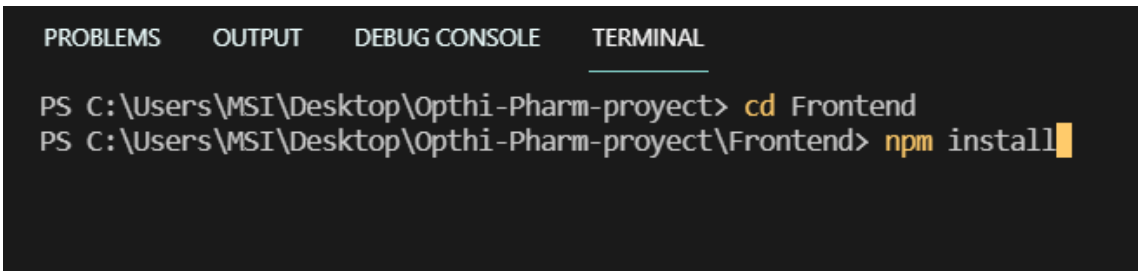


Quan la instal·lació acabi, podrem executar el codi amb npm start. En la cmd ens apareixerà exactament això:



Llavors voldrà dir que el servidor (Backend) estarà actiu. No hem de tancar la cmd on s'està executant el programa.

Després d'això hem de fer el mateix procés però per la carpeta Frontend. Obrirem un altre cmd i anem a la carpeta Frontend instal·larem les dependències.



Després de l'instal·lació farem executar el programa amb npm start

Si tot va bé, s'obrirà una pestanya que aquesta serà l'aplicació d'escriptori.

5.1 Com posar tot en marcha per el client

Endollar la raspberry pi al corrent i obrir l'executable de la màquina. La rasberry pi i l'ordinador que es vol provar ha d'estar al mateix wifi.

6. Codi font del programa

6.1 Backend

6.1.1 Arxiu firebase.js

És trobar totes les API'S per poder interactuar amb la base de dades

```
const express = require("express"); //Importar express
const app = express();

const admin = require("firebase-admin"); //Importar el paquet de
Firebase.

//Variable per portar les claus per poder-se connectar a la base de
dades.
var serviceAccount =
require("../../opthi-pharm-firebase-adminsdk-3kefy-74e6670030.json");

//Inicialització per connectar-se a la base de dades.
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://opthi-pharm-default-rtdb.firebaseio.com/",
});

//Constant que importa totes les funcions que podem fer a la db.
const db = admin.database();
```

```
//Post Crear una nova medicina
app.post("/new-medicine", function (req, res) {
  console.log(req.body);
  //Objecta amb els paramentres que s'enviaran a la base de dades, que
els rebem per req.
  const newMedicine = {
    name: req.body.name,
    row: req.body.row,
    colum: req.body.colum,
    des: req.body.des,
    price: req.body.price,
    amount: req.body.amount,
  };

  db.ref("medicines").push(newMedicine);

  res.send({ ok: true });
});

//GET obtenir totes les medicines que hi ha en la base de dades.
app.get("/get-medicines", function (req, res) {
  let Array_data = [];

  db.ref("medicines").once("value", (snapshot) => {
    data = snapshot.forEach((element) => {
      const el = element.val();
      const object = {
        name: el.name,
        des: el.des,
        colum: el.colum,
        row: el.row,
        id: element.key,
        price: el.price,
        amount: el.amount,
      };
      console.log(object);

      Array_data.push(object);
    });
    res.send(Array_data);
  });
});
```

```
//GET per eliminar medicaments de la base de dades.
app.delete("/delete-medicines", function (req, res) {
  console.log(req.body.id);
  db.ref("medicines/" + req.body.id).remove();
  res.send({ ok: true, delete: true });
});

//PUT per cambiar el contador dels medicaments de la base de dades.
app.put("/amount-update", function (req, res) {
  const id = req.body.id;
  const amount = req.body.amount;

  db.ref("medicines/" + id).update({ amount: amount });
  res.send({ ok: true, update: true });
});

app.put("/update-doc", function (req, res) {
  const id = req.body.id;
  const update = {};

  const amount = req.body.amount;
  const name = req.body.name;
  const des = req.body.des;
  const colum = req.body.colum;
  const row = req.body.row;
  const price = req.body.price;

    //Comprovem quins paramentres estan omplerts i els substiturem.
(Aquesta funcio es podria fer amb un for,
  // no es pot ficar els noms del paramentres, que estiguin guardats en
un array)
  if (amount !== "") {
    update.amount = amount;
  }
  if (name !== "") {
    update.name = name;
  }
  if (des !== "") {
    update.des = des;
  }
  if (colum !== "") {
    update.colum = colum;
```

```
  }
  if (row !== "") {
    update.row = row;
  }
  if (price !== "") {
    update.price = price;
  }

  console.log(update);

  db.ref("medicines/" + id).update(update);

  res.send({ ok: true, update: true });
});

module.exports = app;
```

6.1.2 Matriu.js

En aquest arxiu és on es troba tota la lògica per executar les sortides de la màquina.
La forma que ens entren les dades se'n forma de matriu entre dos Arrays.

Files: [0,0]
Columnes: [0,1]

Vol dir que s'executarà en la primera posició la fila 1 i la columna 1, la segona posició es fila 1 columna 2.

```
const express = require("express"); //Importar els requeriments per
muntar les rutes API
const app = express();
const Gpio = require("onoff").Gpio; //Inclou onoff per interactuar amb
el GPIO

app.post("/comanda", function (req, res) {
  //Constants per poder canviar els paràmetres de la màquina.
  const Interval_Mateix_Calaix = 3000;
  const Interval_Sortida_Activada = 1000;

  //Constans per recollir la informació que arriba del client.
  const row = req.body.row;
  const colum = req.body.colum;
```

```
  //Sortidas del GPIO
  //Columnes
  const Colum_1_s = new Gpio(13, "out");
  const Colum_2_s = new Gpio(19, "out");
  const Colum_3_s = new Gpio(26, "out");
  //Files
  const Row_1_s = new Gpio(16, "out");
  const Row_2_s = new Gpio(20, "out");
  const Row_3_s = new Gpio(21, "out");

  //Matrius de sortida per afegir l'has sortides que s'executaran
  const colum_ = [Colum_1_s, Colum_2_s, Colum_3_s];
  const row_ = [Row_1_s, Row_2_s, Row_3_s];

  //Pins de las sortides del GPIO
  const sortidas_row = [16, 20, 21];
  const sortidas_colum = [13, 19, 26];

  let Array_row = []; //Array per col·locar les files en ordre

  for (let i = 0; i < 3; i++) {
     const Row = orden(i); //Executem la funcio per fer el triatge de
cada una de las filas.

    Array_row.push(Row); //Ficar el resultat a l'array.
  }

  //Funció per ordenar les files i columnes.
  //Per què es realitza aquesta funció? Perquè si la maquina només vol
activar la fila 1 i la 3. Ha de seguir aquest ordre, igual per las
columnes.
  function orden(Fila) {
    let Array_Colum = [];
    let Array_Row = [];

    for (let index = 0; index < row.length; index++) {
      const row_ = row[index];

      if (row_ == Fila) {
        Array_Colum.push(colum[index]);
        Array_Row.push(row[index]);
      }
```

```
    }

    return { Array_Colum, Array_Row };
  }

    //Plantilla per poder fer un esquema de las sortides que san
d'executar.
  //Per què es fa això? Perquè necessitem més endavant les vegades que
hem d'executar cada calaix de la màquina.
  let Rep = { Numero_0: 0, Numero_1: 0, Numero_2: 0 };
  let Array_Final = []; //Array que obtindrà l'objecte "Rep" ple de les
vegades que tindrem d'executar las sortides

   //For per recórrer cada una de les posicions i crear els objectes
descrits anteriorment
  for (let index = 0; index < Array_row.length; index++) {
    Rep = { Numero_0: 0, Numero_1: 0, Numero_2: 0 };

    const Colum_Array = Array_row[index].Array_Colum;

    console.log(Colum_Array);

    Colum_Array.forEach(function (numero) {
      switch (numero) {
        case "0":
          Rep.Numero_0 = Rep.Numero_0 + 1;
          break;
        case "1":
          Rep.Numero_1 = Rep.Numero_1 + 1;
          break;
        case "2":
          Rep.Numero_2 = Rep.Numero_2 + 1;
          break;
        default:
          break;
      }
    });

    Array_Final.push(Rep);
  }

  //Mostrem per consola el Array_Final
  console.log(Rep);
```

```
  console.log(Array_Final);

  let Colum = "";
  let Row = "";

  //Funció per activar les sortides
  const Ex_Sortides = (again, colum, row) => {
    if (again != 0) {
      // Guardem l'identificador d'interval per després cancel·lar-lo.
      var id_intervalo = setInterval(hazAlert, Interval_Mateix_Calaix);
          var  i_contador  =  0;  //Comptador  per  saber  quantes  portem
executades

       //Fució que es crida cada cert temps si és que la comanda hi ha
més d'una execució per columna
      function hazAlert() {
          // Quan arribem al número indicat per aquesta columna no es
tornarà a executar sinó seguirà executen-se la sortida.
        if (i_contador >= again) {
          clearInterval(id_intervalo);
        } else {
          //Comentaris explicats mes entre la línia de codi 79 a 90.
          console.log("Activo la Fila: " + sortidas_row[row]);
          console.log("Activo la Columna: " + sortidas_colum[colum]);
          //Fiquem les sortides que volem activar en les variables.
          Colum = row_[row];
          Row = colum_[colum];
          //Activem las sortides seleccionades (GPIO)
          Colum.writeSync(1);
          Row.writeSync(1);
                 //Fem  que  les  sortides  estiguin  activades  un  temps
determinat.
                              var   id_descativar   =   setInterval(des,
Interval_Sortida_Activada);
          //Funció per desactivar les sortides.
          function des() {
                    clearInterval(id_descativar);  //Reset  de  iterador
id_descativar.
            Colum.writeSync(0);
            Row.writeSync(0);
          }

          i_contador++;
```

```
        }
      }

      return "ok";
    }
  };

  // Funció per saber el temps que hem d'esperar entre fila i fila.
   // El que fem és agafar totes les dades que necessitem del Array i
posar-los en un array que amb la funció Math.max obtenim el número més
gran del Array.
   // Seguidament el multipliquem per saber l'estona que hauria de
passa.
  function max(numero) {
    const Colum_0 = Array_Final[numero].Numero_0;
    const Colum_1 = Array_Final[numero].Numero_1;
    const Colum_2 = Array_Final[numero].Numero_2;

    const Array_max = [];

    Array_max.push(Colum_0, Colum_1, Colum_2);
    const max = Math.max(...Array_max);
    const x = max * Interval_Mateix_Calaix;

    console.log(x);

    return x;
  }

  let Array_Control = [];

  control();

  //Funció que es la qui porta el control d'executar les sortides
  function control() {
    let id_0 = 0;
    let id_1 = 0;

     //For per determinar en quina fila hem de començar a executar el
codi.
    for (let index = 0; index < 3; index++) {
      if (
        Array_Final[index].Numero_0 |
```

```
        Array_Final[index].Numero_1 |
        (Array_Final[index].Numero_2 != 0)
      ) {
        Array_Control.push(index);
      }
    }

     //Tres possibilitats de començar. Els setIntervals són per seguir
el codi en execució,
     // respectivament en cada fila amb el temps determinat pel nombre
màxim de medicament dispensat en una columna.
    if (Array_Control[0] == 0) {
      Execut(0);
      const t1 = max(0);
      id_0 = setInterval(row_1, t1);
    }

    if (Array_Control[0] == 1) {
      Execut(1);
      const t2 = max(1);
      id_1 = setInterval(row_2, t2);
    }

    if (Array_Control[0] == 2) {
      row_2();
    }

    //Diferents opcions que ens podrem trobar
     //Si només hem d'executar la última fila o començar des de la final
1.
    function row_1() {
      clearInterval(id_0);
      Execut(1);
      const t3 = max(1);
      id_1 = setInterval(row_2, t3);
    }

    function row_2() {
      clearInterval(id_1);
      Execut(2);
    }
  }
```

```
  //Funció per executar les sortides de les files.
   //Aquesta funció executa el codi notBlockin mencionat a la memòria,
el que farà és executar l'has sortits a la vegada.
  function Execut(num) {
    const Colum_0 = Array_Final[num].Numero_0;
    const Colum_1 = Array_Final[num].Numero_1;
    const Colum_2 = Array_Final[num].Numero_2;

    const num_0 = 0;
    const num_1 = 1;
    const num_2 = 2;

    Ex_Sortides(Colum_0, num_0, num, () => {});
    Ex_Sortides(Colum_1, num_1, num, () => {});
    Ex_Sortides(Colum_2, num_2, num, () => {});
  }

  //Funció per avisar que la comanda ha sigut executada
  //Calcula el temps total que tardarà a realitzar la comanda.
  const t = max(0);
  const tm = max(1);
  const tmp = max(2);

  const temps_total = t + tm + tmp;

   const id_total = setInterval(send, temps_total); //Quan passi el
temps calculat enviarà el send.

  //Funció per enviar una resposta al client.
  function send() {
    clearInterval(id_total); //Fem reset de l'iteració id_total.
      res.send({ ok: true }); //Enviem un objecte JSON al consumidor
d'aquesta API. "Comanda realitzada"
  }
});

module.exports = app;
```

6.2 Frontend

6.2.1 Arxiu index.js

En aquest arxiu es on es troba tota la lógica del crear la comanda i la primera pàgina de la maquina.

```
const API_DEV = "http://localhost:3001"; // Direcció de proves
const API_PRO = "http://192.168.50.52:3001"; // Direcció de producció
(Rasberry Pi - Back-end).

const API_DIRECTION = API_PRO; // Configuració d'on apuntarà el client
per poder-se comunicar amb la base de dades.

const most = document.getElementById("targeta"); //Importació de
l'element targeta, on sortirà el medicament buscat.
const spiner_ = document.getElementById("spiner"); // Importacció del
element spiner, es el espiner de carrega comanda.
const btnBus = document.getElementById("btn-bus"); // Importació de
l'element btn-bus, buto de busqueda del medicament.
const registres = document.querySelector("#registres"); // Importació
de l'element registres, taula on sortira la comanda.
const btn_comanda = document.getElementById("btn-comanda"); //
Importació de l'element btn-comanda, boto per executar la comanda.

const quan = document.getElementById("quantitat"); // Importació de
l'element quantitat, és l'input que trobem per col·locar la quantitat
de medicaments que volem.

let Array_Medicaments = []; // Array que s'emplena amb tots els
medicaments de la base de dades.
let Array_NomMedica = []; // Array amb només els noms dels
medicaments.;

GetMedicaments(); // Executem la funció GetMedicaments per portar els
medicaments de base de dades.

// Funció GetMedicaments per portar els medicaments de base de dades.
function GetMedicaments() {
  // console.log("GetMedicaments");

  //Petició GET http per obtenir els medicaments, aquesta ruta apunta a
la API /get-medicines del back-end.
```

```
  fetch(`${API_DIRECTION}/get-medicines`, {
    method: "GET",
    headers: {
      "Content-Type": "application/json",
    },
  }) // Quan obtenim els medicaments, recorrem l'array i col·loque'm el
medicament en els Arrays.
    .then((res) => res.json())
    .then((data) => {
      data.forEach((element) => {
        Array_Medicaments.push(element);
        Array_NomMedica.push(element.name);
      });
    });
}

//Autocompletar busquedas
autocomplete(document.getElementById("marcas"),        Array_NomMedica);
//Obtenim el id.

//Funció  per autocompletar busquedas.
function autocomplete(inp, arr) {
  var currentFocus;
  inp.addEventListener("input", function (e) {
    var a,
      b,
      i,
      val = this.value;
    closeAllLists();
    if (!val) {
      return false;
    }
    currentFocus = -1;
    a = document.createElement("DIV");
    a.setAttribute("id", this.id + "autocomplete-list");
    a.setAttribute("class", "autocomplete-items p-2");
    this.parentNode.appendChild(a);
    for (i = 0; i < arr.length; i++) {
                  if   (arr[i].substr(0,   val.length).toUpperCase()   ==
val.toUpperCase()) {
        b = document.createElement("DIV");
            b.innerHTML  =  "<strong>"  +  arr[i].substr(0,  val.length)  +
"</strong>";
```

```
      b.innerHTML += arr[i].substr(val.length);
      b.innerHTML += "<input type='hidden' value='" + arr[i] + "'>";
      b.addEventListener("click", function (e) {
        inp.value = this.getElementsByTagName("input")[0].value;
        closeAllLists();
      });
      a.appendChild(b);
    }
  }
});

inp.addEventListener("keydown", function (e) {
  var x = document.getElementById(this.id + "autocomplete-list");
  if (x) x = x.getElementsByTagName("div");
  if (e.keyCode == 40) {
    currentFocus++;

    addActive(x);
  } else if (e.keyCode == 38) {
    currentFocus--;

    addActive(x);
  } else if (e.keyCode == 13) {
    e.preventDefault();
    if (currentFocus > -1) {
      if (x) x[currentFocus].click();
    }
  }
});
function addActive(x) {
  if (!x) return false;
  removeActive(x);
  if (currentFocus >= x.length) currentFocus = 0;
  if (currentFocus < 0) currentFocus = x.length - 1;

  x[currentFocus].classList.add("autocomplete-active");
}
function removeActive(x) {
  for (var i = 0; i < x.length; i++) {
    x[i].classList.remove("autocomplete-active");
  }
}
function closeAllLists(elmnt) {
```

```
    var x = document.getElementsByClassName("autocomplete-items");
    for (var i = 0; i < x.length; i++) {
      if (elmnt != x[i] && elmnt != inp) {
        x[i].parentNode.removeChild(x[i]);
      }
    }
  }



////////////////////////////////////////////////////////////////////////////
//////////////////////////

  btnBus.addEventListener("click", searchArray); //Quan fan clic en el
botó btnBus (Cerca)

  // Funció per mostra la cerca del medicament en una targeta.
  function searchArray() {
    //console.log("searchArray");
      const res_bus = Array_NomMedica.indexOf(inp.value); //Fiques el
valor de l'input de cerca i record l'array, per trobar si el nom
existeix, si és així torna la posició, si no torna un -1
    //Si a trobat el medicament saltarà el filtre
    if (res_bus != -1) {
       const object = Array_Medicaments[res_bus]; //Traiem l'objecte que
volem aconseguir partir de la posició trobada anteriorment i
l'Array_Medicaments.
      targeta.innerHTML = ""; //Borem la targeta anterior.
      //Coloquem la targeta amb els valor del medicament.
      targeta.innerHTML += `<div class="card" >
      <div class="card-header">
      ${object.name}
    </div>
    <h5 class="card-title"></h5>
    <p class="card-text">${object.des}</p>
    <h6>Preu: ${object.price} </h6>
              <input   type="button"   class="btn   btn-outline-success"
onclick="afegirComanda('${res_bus}');" value="Afegira a la comanda">
              <input   type="button"   class="btn   btn-danger   btn-sm"
onclick="borraBusqueda();" value="Eliminar">
     </div>
  `;
    }
  }
```

```
}

//Funció per borrar la tarjeta de la cerca.
function borraBusqueda() {
  targeta.innerHTML = "";
}

let comanda = [];
let id_rep = [];

//Funció per afegir la comanda a la taula.
function afegirComanda(res_bus) {
   const object = Array_Medicaments[res_bus]; //Donem la posició de
l'Array. "La funció searchArray LINEA 121" Activació "LINEA 134".

  let pre = quan.value; //Agafem el valor de quantitat de medicaments
volem dispensar.

  // console.log(pre);
  //Farem un filtre per sempre que no tingui res l'input afegiríem un
medicament mínim "Valor predeterminat 1"
  if (pre == 0) {
    pre = 1;
  }

  object.quantitat = pre; //Afegim la propietat quantitat a l'objecte
amb el número de medicaments.

  //console.log(object);

   const price = object.price; //traiem el valor del preu en una
constant.
  const total = pre * price; //Calculem el preu total.
  object.total = total; //Afegim la propietat total a l'objecte el preu
total unitari.

   //Crem un filtre per identificar si el medicament que s'intenta
afegir no està afegit amb anterioritat.
  const rep = id_rep.includes(object.id);

   //Si no passa el filtre mostrarem una alerta (No fiquem l'alerta
perquè el programa es bloqueja per alguna raó "està comentat ")
  if (rep == true) {
```

```
    // alert("No pot afegir dos meicaments iguals");
  } else {
     id_rep.push(object.id); //Afegirem l'id en una llista que serà amb
la que es compara en la LINEA 171.
    comanda.push(object); //Afegim l'objecta l'Array comanda.
    // console.log(comanda);
      carregarTaula(comanda); //Executem la funció carregarTaula, per
actualitzar la taula amb el nou objecte dintre del array comanda.
    // console.log(id_rep);
  }
}

//Funció per carregar la taula de comanda.
function carregarTaula(comanda) {
   let html = comanda.map(TaulaCoamandaHtml).join(""); //Recorrem el
array comanda, criden cada cop a la funció TaulaComandaHtml i
l'encadena'm.

   registres.innerHTML = html; //col·loque'm a registres el codi HTML
generat amb la funció anterior LINEA 187.

  function TaulaCoamandaHtml(index) {
    //Funció per genera el HTML.
    return `<tr>
  <td>${index.name}</td>
  <td>${index.quantitat}</td>
  <td>${index.price}€</td>
  <td>${index.total}€</td>
       <td><input    type="button"    class="btn    btn-danger    btn-sm"
onclick="eliminarComanda('${index.id}');"    value="Eliminar    de    la
comanda"></td>
</tr>`;
  }
}

//Funció per eliminar un medicament de la taula de comanda.
function eliminarComanda(index) {
  let pos_id = [];

  //Recorrem la comanda i col·loque'm tots els id en l'array declarat
anteriorment en la LINEA 204.
  comanda.forEach((element) => {
    pos_id.push(element.id);
```

```
  });
  //Busquem quina posició es troba.
  const pos = pos_id.indexOf(index);

  // console.log(pos);
   // Borarem l'objecta en l'array comanda i en l'array id_rep per
deixar ficar un altre cop aquest medicament a la comanda.
  const max = pos + 1;
  comanda.splice(pos, max); //Array --> comanda.
  id_rep.splice(pos, max); // Array --> id_rep. Comentat LINEA 170.

  carregarTaula(comanda); //carege'm la taula amb el nou Array comanda.
}

btn_comanda.addEventListener("click", executarComanda); //Quan fan clic
en el botó btn_comanda (Executar comanda) crida a la funció
executarComanda.

//Funció per executar la comanda.
function executarComanda() {
  let ArrayRow = []; //Array amb les files que s'executaran.
  let ArrayColum = []; //Array amb les columnes que s'executaran.

  // console.log(comanda);

   //Crem let globals per aquesta funció es modificarà el valor més
endavant.
  let amount = "";
  let quantitat = "";
  let id = "";
  let Stock = "";

  comanda.forEach((element) => {
    const row = element.row;
    const colum = element.colum;
    amount = element.amount;
    quantitat = element.quantitat;
    id = element.id;

    Stock = amount - quantitat; //Calculem el Stock que quedarà.

    //Recorrem els array, per poder enviar la comanda.
    for (let index = 0; index < quantitat; index++) {
```

```
      ArrayColum.push(colum);
      ArrayRow.push(row);
    }
  });

  //Activem el spiner de càrrega.
  // on_spiner();

   //Fem una petició http POST a la direcció /comanda, "Enviem els
arrays a la rasberry pi"
  fetch(`${API_DIRECTION}/comanda`, {
    method: "POST",
    body: JSON.stringify({
      row: ArrayRow,
      colum: ArrayColum,
    }),
    headers: {
      "Content-Type": "application/json",
    },
    }).then((data)  =>  console.log(data));  //Mostrem  la  resposta  per
consola.
  //Fem una petició http PUT a la direcció /amount-update, "Enviem els
Stock que quedara"
  fetch("http://localhost:3001/amount-update", {
    method: "PUT",
    body: JSON.stringify({
      id: id,
      amount: Stock,
    }),
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((res) => res.json())
    .then((data) => {
      //Quan arribi la dada farem para el spiner de càrrega

      // off_spiner();
    });
}

//Funció per activar el spiner de càrrega.
function off_spiner() {
```

```
  spiner_.innerHTML += `
  <div>
  </div>
  `;
}

//Funció per activar el spiner de càrrega.
function on_spiner() {
  spiner_.innerHTML += `
  <div class="spinner-border" role="status">
  <span class="visually-hidden">Loading...</span>
  </div>
  `;
}
```

6.2.2 Arxiu register.js

Arxiu per gestionar els medicaments de la base de dades.

```
const API_DEV = "http://localhost:3001"; // Direcció de proves
const API_PRO = "http://192.168.50.52:3001"; // Direcció de producció
(Rasberry Pi - Back-end).

const API_DIRECTION = API_PRO; // Configuració d'on apuntarà el client
per poder-se comunicar amb la base de dades.

const newRegister = document.getElementById("btnCrearRegistro");
//Importació de l'element btnCrearRegistro, boto per crear el
medicament.
const registros = document.getElementById("registros");
const name_ = document.getElementById("name"); //Importació de
l'element name, input del nom del medicament.
const des = document.getElementById("des"); //Importació de l'element
des, input de la descripció del medicament.
const colum = document.getElementById("colum"); //Importació de
l'element colum, input de la columna on es trobarà el medicament.
const row = document.getElementById("row"); //Importació de l'element
row, input de la row on es trobarà el medicament.
const preu = document.getElementById("preu"); //Importació de l'element
preu, input del preu el medicament.
const quantitat = document.getElementById("quantitat"); //Importació de
l'element quantitat, input del STOCK del medicament.
```

```
const modic = document.getElementById("modic"); //Importació de l'element modic, és la targeta per quan modifiquem el medicament.

// const p = document.getElementById("btnNEW");

newRegister.addEventListener("click", NewMedicines); //Quan fan clic en newRegistre (Crear noumedicament) Executem NewMedicines.

let Array_Medicamentos = []; // Array que s'emplena amb tots els medicaments de la base de dades.
let Array_id = []; // Array amb només els id dels medicaments.

GetMedicines(); //Cada cop que carregem la pagina executem GetMedicines.

//Funció per crear un nou medicament.
function NewMedicines() {
  //console.log("NewMedicines");

  //Petició POST http per crear els medicaments, aquesta ruta apunta a la API /new-medicine del back-end.
  fetch(`${API_DIRECTION}/new-medicine`, {
    method: "POST",
    body: JSON.stringify({
      name: name_.value,
      des: des.value,
      row: row.value,
      colum: colum.value,
      amount: quantitat.value,
      price: preu.value,
    }),
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((res) => res.json())
    .then((data) => {
      console.log(data);
    });

  Array_Medicamentos = []; //Fiquem el Array estigui completament vuit.
  GetMedicines(); //Executem la funció GetMedicines.
```

```
    carregarTaula(Array_Medicamentos);  //Carregem la taula amb el nou
medicament.
}

// Funció GetMedicaments per portar els medicaments de base de dades.
function GetMedicines() {
  // console.log("GetMedicaments");

  //Petició GET http per obtenir els medicaments, aquesta ruta apunta a
la API /get-medicines del back-end.
  fetch(`${API_DIRECTION}/get-medicines`, {
    method: "GET",
    headers: {
      "Content-Type": "application/json",
    },
  }) // Quan obtenim els medicaments, recorrem l'array i col·loque'm el
medicament en els Arrays.
    .then((res) => res.json())
    .then((data) => {
      data.forEach((element) => {
        Array_Medicamentos.push(element);
        Array_id.push(element.id);
      });
        carregarTaula(Array_Medicamentos); //Carregem la taula amb els
medicaments de la base de dades.
    });
}

//Funció per eliminar una medicina de la base de dades a partir de l'id
del medicament.
function eliminarMedicament(id) {
  //console.log(`eliminarMedicament + ${id}`);

  let Array_id = [];
  //Petició DELETE http per borrar el medicament, aquesta ruta apunta a
la API /delete-medicines del back-end.
  fetch(`${API_DIRECTION}/delete-medicines`, {
    method: "DELETE",
    body: JSON.stringify({
      id: id,
    }),
    headers: {
      "Content-Type": "application/json",
```

```
    },
  })
    .then((res) => res.json())
    .then((data) => console.log(data));
  Array_Medicamentos.forEach((element) => {
    Array_id.push(element.id);
  });

  const pos = Array_id.indexOf(id); //Identifica quina posició es troba
l'element que volem eliminar.

  //console.log(pos);
  // Borarem l'objecta en l'Array_Medicamentos i en l'array Array_id.
  const max = pos + 1;
  Array_Medicamentos.splice(pos, max);
  Array_id.splice(pos, max);

  carregarTaula(Array_Medicamentos); //Executem la funció carregarTaula
i carregem la nova taula amb les noves dades.
}

//console.log(Array_Medicamentos);

//Funció per editar el medicament seleccionat aparti de l'id.
function edit(id) {
  const pos = Array_id.indexOf(id); //Busquem la posició d'on es troba.
   const ob = Array_Medicamentos[pos]; //Trobem l'objecte on es troba
tota la informació del medicament.
  const mod_name = ob.name; //Nom del medicament

  modic.innerHTML = ""; //Borem la targeta de modificar el medicament.

  //Mostrem la targeta de modificar el medicament.
  modic.innerHTML += `
  <div>
   <h1>Estas modifican el medicament ${mod_name}</h1>
           <input    type="button"    class="btn    btn-danger    btn-sm"
onclick="cancelaEdit();" value="Cancelar modificar el medicament">
           <input    type="button"    class="btn    btn-danger    btn-sm"
onclick="update('${id}');" value="Modificar el medicament">
  </div>
`;
}
```

```
//Funció per actualitzar el medicament a la base de dades apartir del
id.
function update(id) {
  // console.log("update");

  //Petició PUT http per modficar el medicament, aquesta ruta apunta a
la API /update-doc del back-end.
  fetch(`${API_DIRECTION}/update-doc`, {
    method: "PUT",
    body: JSON.stringify({
      name: name_.value,
      des: des.value,
      row: row.value,
      colum: colum.value,
      amount: quantitat.value,
      price: preu.value,
      id: id,
    }),
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((res) => res.json())
    .then((data) => console.log(data));

  Array_Medicamentos = []; //Fiquem el Array estigui completament vuit.
  GetMedicines(); //Executem la funció GetMedicines.
   carregarTaula(Array_Medicamentos); //Carregem la taula amb el nou
medicament.
}

//Cancel·lem editar el medicament.
function cancelaEdit() {
  //console.log("cancelaEdit");
  modic.innerHTML = "";
}

//Funció per carregar la taula amb els medicaments
function carregarTaula(m) {
   let html = m.map(TaulaComandaHtml).join(""); //Recorrem el array de
medicaments, criden cada cop a la funció TaulaComandaHtml i
l'encadena'm.
```

```
   registros.innerHTML = html; //col·loque'm a html LINEA 164 el codi
HTML generat amb la funció anterior

  function TaulaComandaHtml(index) {
    // console.log(index);

    return `<tr>
  <td>${index.name}</td>
  <td>${index.des}</td>
  <td>${index.colum}</td>
  <td>${index.row}</td>
  <td>${index.amount}</td>
  <td>${index.price} €</td>
       <td><input    type="button"    class="btn    btn-danger    btn-sm"
onclick="edit('${index.id}');" value="Editar medicament"></td>
       <td><input    type="button"    class="btn    btn-danger    btn-sm"
onclick="eliminarMedicament('${index.id}');" value="Eliminar de base de
dades"></td>
</tr>`;
  }
}
```

6.2.4 Arxiu state.js

Arxiu per saber l'estat de la màquina.

```
const API_DEV = "http://localhost:3001"; // Direcció de proves
const  API_PRO  =  "http://192.168.50.52:3001";  // Direcció de producció
(Rasberry Pi - Back-end).

const API_DIRECTION = API_PRO; // Configuració d'on apuntarà el client
per poder-se comunicar amb la base de dades.

const  est  =  document.getElementById("est");  //Importació de l'element
targeta, on sortirà el medicament buscat.

let  Array_Medicamentos  =  [];  // Array  que  s'emplena  amb  tots  els
medicaments de la base de dades.
let correct = false;

GetMedicines();   //Cada   cop   que   carregem   la   pagina   executem
GetMedicines.
```

```
function GetMedicines() {
  // console.log("GetMedicaments");

  //Petició GET http per obtenir els medicaments, aquesta ruta apunta a
la API /get-medicines del back-end.
  Array_Medicamentos = [];
  fetch(`${API_DIRECTION}/get-medicines`, {
    method: "GET",
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((res) => res.json())
    .then((data) => {
      data.forEach((element) => {
              Array_Medicamentos.push(element);  //col·loque'm  tot  els
medicaments obtinguts en l'array.
        // console.log(Array_Medicamentos.length);
        //Filtre: Si el array te mes posicions que 0
        if (Array_Medicamentos.length !== 0) {
          state(); //Executem la fucnió satate
          correct = true; //Coloquem a la varibale correct "true".
        }
      });
    });
}
//console.log(Array_Medicamentos);

//Filtre  per  veure  si  tenim  connexió  amb  el  back-end  i  la  base  de
dades.
if (correct == false) {
  not_oper(); //Executar funció not_oper.
}

//Funció per mostrar que la connexió està establerta.
function state() {
  est.innerHTML = ""; //Esborrem el que té la pantalla

  est.innerHTML += `
  <div class="row">
       <div  class="col-5"><h3>La  Base  de  dades  i  el  back-end  estan
operatius</h3></div>
```

```
    <div class="col-5 card bg-success">
    </div>
  </div>
`;
}

//Funció per mostrar que la connexió no esta establerta.
function not_oper() {
  est.innerHTML = ""; //Esborrem el que té la pantalla

  est.innerHTML += `
  <div class="row">
      <div  class="col-5"><h3>La  Base  de  dades  i  el  back-end  estan
operatius</h3></div>
   <div class="col-5 card bg-danger">
  </div>
  </div>
`;
}
```

6.2.5 Arxiu Index.html

Aquest arxiu va relacionat amb l'arxiu index.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
     <meta name="viewport" content="width=device-width, initial-scale=1"
/>

    <!-- Bootstrap CSS -->
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boots
trap.min.css"
      rel="stylesheet"

integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ck
xlpbzKgwra6"
      crossorigin="anonymous"
```

```
    />

    <link rel ="stylesheet" href="css/New_att.css">

    <title>Opthi-Pharm</title>
    <nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark">
      <div class="container-fluid">
        <img src="assets/logo.png" alt="" width="100" height="50" />
        <button
          class="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarNavAltMarkup"
          aria-controls="navbarNavAltMarkup"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
          <a class="nav-link text-light" href="index.html">Comanda</a>
            <a
              class="nav-link active text-light"
              aria-current="page"
              href="html/register.html"
              >Gestio de la DB</a
            >
             <a class="nav-link text-light" href="html/state.html">Estat
de la màquina</a>
          </div>
        </div>
      </div>
    </nav>
  </head>
  <body>

  <div class="container-fluid">
    <div class="row">
    <!--columna esquerra-->
    <div class="col-12 col-lg-6">

      <div class="row justify-content-center">
```

```
    <!--Part Esquerra-->
      <div class="col-10 col-lg-5">
                  <form  autocomplete="off"  action="/action_page.php"
class="d-flex">
            <div class="autocompletado" style="width: 300px">
               <div class="form-outline">
                 <input
                  class="form-control w-100"
                  id="marcas"
                  type="text"
                  name="Marcas"
                  placeholder="Buscar un medicaments"
                  aria-label="Search"
                 />
               </div>
             </div>
        </form>
      </div>
      <!--Part Dreta-->
      <div class="col-2 col-lg-7">
                <button  id="btn-bus"  class="btn  btn-outline-success"
type="submit">
         Buscar
       </button>
     </div>

   <div class="row justify-content-center">
      <div class="col-md-6 col-ld-6 col-6" id="targeta">
       <div class="card">
         <h5 class="card-title">Buscar un medicament per poder afegir a
la comanda</h5>
       </div>
     </div>

   <div class="col-3 col-lg-6">
     <input
          type="text"
          id="quantitat"
          class="form-control mb-4"
          placeholder="Quantitat predetirminada 1"
        />
        </div>
  </div>
```

```
</div>
</div>

 <!--columna dreta-->
 <div class="col-12 col-lg-6 p-3">

        <table class="table align-middle" style="width: 100%">
          <tr>
            <th>Producte</th>
            <th>Quantitat</th>
            <th>Preu</th>
            <th>Total</th>
            <th>Eliminar</th>
          </tr>
          <tbody id="registres"></tbody>
        </table>
          <div class="row">
          <div class="col-12 col-lg-3 p-3">
          <input
            id="btn-comanda"
            type="button"
            class="btn btn-outline-success"
            value="Realitzar comanda"
          />
          </div>
          <div class="col-12 col-lg-6 p-3">
            <div class="" id="spiner">
            </div>
          </div>

          </div>

      </div>



  <script type="text/javascript" src="js/index.js"></script>

  <!-- Optional JavaScript; choose one of the two! -->

  <!-- Option 1: Bootstrap Bundle with Popper -->
  <script
```

```
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"
      crossorigin="anonymous"
    ></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
                                                                <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js"
integrity="sha384-SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1RDvt3wT4gWFG" crossorigin="anonymous"></script>
                                                                <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js"
integrity="sha384-j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQqNXNE6DZiVc" crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

6.2.6 Arxiu register.html

Aquest arxiu va relacionat amb l'arxiu register.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
     <meta name="viewport" content="width=device-width, initial-scale=1" />

    <!-- Bootstrap CSS -->
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"
      rel="stylesheet"
```

```
integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ck
xlpbzKgwra6"
      crossorigin="anonymous"
    />

    <title>Opthi-Pharm</title>
    <nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark">
      <div class="container-fluid">
        <img src="../assets/logo.png" alt="" width="100" height="50" />
        <button
          class="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarNavAltMarkup"
          aria-controls="navbarNavAltMarkup"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
                                      <a   class="nav-link   text-light"
href="../index.html">Comanda</a>
            <a
              class="nav-link active text-light"
              aria-current="page"
              href="register.html"
              >Gestio de la DB</a
            >
            <a class="nav-link text-light" href="state.html"
              >Estat de la màquina</a
            >
          </div>
        </div>
      </div>
    </nav>
  </head>
  <body>

  <div class="container-fluid">
```

```
    <div class="col-md-6" id="modic"></div>

    <form id="new" class="border border-light p-4">
      <p class="h3 mb-4 text-left">Nou Medicament</p>

      <input
        type="text"
        id="name"
        class="form-control mb-4"
        placeholder="Nom del medicament"
        Required
      />

      <input
        type="text"
        id="des"
        class="form-control mb-4"
        placeholder="Descripció del medicament"
        Required
      />

      <input
        type="text"
        id="colum"
        class="form-control mb-4"
        placeholder="Columna del medicament a la màquina"
        Required
      />

      <input
        type="text"
        id="row"
        class="form-control mb-4"
        placeholder="Fila del medicament a la màquina"

      />

         <input
        type="text"
        id="preu"
        class="form-control mb-4"
        placeholder="Preu"
        Required
```

```
      />

          <input
        type="text"
        id="quantitat"
        class="form-control mb-4"
        placeholder="Quantitat"
        Required
      />

        <button
        id="btnCrearRegistro"
        class="btn btn-success btn-block my-4"
        type="Fila"
      >
        Crear un nou medicament
      </button>
    </form>


     <table class="table table-striped p-4">
            <thead>
              <tr>
                <th>Nom</th>
                <th>Descripció</th>
                <th>Columna</th>
                <th>Fila</th>
                <th>Quantitat</th>
                <th>Preu</th>
                <th>Editar</th>
                <th>Eliminar</th>
              </tr>
            </thead>
                              <tbody   class="border   border-light   p-4"
id="registros"></tbody>
          </table>
        </div>

    </div>


                                      <script     type="text/javascript"
src="../js/register.js"></script>
```

```
    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"
      crossorigin="anonymous"
    ></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
                                                                <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js"
integrity="sha384-SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1RDvt3wT4gWFG" crossorigin="anonymous"></script>
                                                                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js"
integrity="sha384-j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQqNXNE6DZiVc" crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

6.2.7 Arxiu state.html

Aquest arxiu va relacionat amb l'arxiu state.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
     <meta name="viewport" content="width=device-width, initial-scale=1" />

    <!-- Bootstrap CSS -->
```

```
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boots
trap.min.css"
      rel="stylesheet"

integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ck
xlpbzKgwra6"
      crossorigin="anonymous"
    />
    <link
      rel="stylesheet"
      type="text/css"
      media="screen"
      href="../css/css_code.css"
    />

    <title>Opthi-Pharm</title>
    <nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark">
      <div class="container-fluid">
        <img src="../assets/logo.png" alt="" width="100" height="50" />
        <button
          class="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarNavAltMarkup"
          aria-controls="navbarNavAltMarkup"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
                                      <a   class="nav-link   text-light"
href="../index.html">Comanda</a>
            <a
              class="nav-link active text-light"
              aria-current="page"
              href="register.html"
              >Gestio de la DB</a
            >
            <a class="nav-link text-light" href="state.html"
```

```
              >Estat de la màquina</a
            >
          </div>
        </div>
      </div>
    </nav>
  </head>
  <body>
    <div class="container-fluid">
      <div class="col-md-6" id="est"></div>
    </div>

    <script type="text/javascript" src="../js/state.js"></script>
    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"
      crossorigin="anonymous"
    ></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
                                                                <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js"
integrity="sha384-SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1RDvt3wT4gWFG" crossorigin="anonymous"></script>
                                                                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js"
integrity="sha384-j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQqNXNE6DZiVc" crossorigin="anonymous"></script>
    -->
  </body>
</html>
```