
# Desenvolupament d'aplicacions web amb tecnologia Java EE

Projecte Fruit Store

**Nom Estudiant**
GEI
Java EE

**Albert Grau Perisé**
**13 de juny de 2018**

**FITXA DEL TREBALL FINAL**

| | |
|---|---|
| **Títol del treball:** | *FruitStore* |
| **Nom de l'autor:** | *Alejandro Jimenez Nadal* |
| **Nom del consultor/a:** | *Albert Grau Perisé* |
| **Nom del PRA:** | Santi Caballe Llobet |
| **Data de lliurament (mm/aaaa):** | *06/2018* |
| **Titulació o programa:** | *Grau d'Enginyeria Informàtica* |
| **Àrea del Treball Final:** | *Java EE* |
| **Idioma del treball:** | *Català, castellà o anglès* |
| **Paraules clau** | *Spring, Angular, REST* |


**Resum del Treball (màxim 250 paraules):** *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

La finalitat d'aquest projecte amb tecnologies Java EE es adquirí coneixements. Per poder adquirir aquest coneixements s'ha creat una aplicació web basada en Spring i ANgularJS, dues tecnologies en constant evolució, però molt utilitzades.

Com que els tipus de negocis que apareixen avui en dia es basen principalment en la venta a traves de la web, s'ha desenvolupat un aplicació amb un exemple de negoci, en aquest cas una fruiteria.

Per desenvolupar correctament s'ha seguit un cicle de vida estàndard, començant per la planificació del projecte, seguint aquesta planificació es pot ajustar el temps a les fites d'entrega. A continuació un anàlisis funcional i de requisit, per conèixer les necessitats del client, seguit del disseny de l'aplicació.

Un cop acabades les tres primeres fases ja podem començar a desenvolupar e implementar l'aplicació.

I per comprovar el correcte funcionament de l'aplicació se realitza una bateria de probes.

Per concloure es documenta tot el projecte, cada una de les fases.

Durant tot el projecte s'han hagut de fer modificacions als documents d'anàlisi i de requisit, perquè a mesura que es desenvolupa sorgeixen dubtes o prolemes que no s'havien previst, així com canvis en el disseny provocats per dificultats a l'hora de l'implementació.

El resultat obtingut ha estat l'esperat, s'ha aconseguit desenvolupar una aplicació web basada en Java amb Spring i AngularJS.

Com a conclusió puc dir que amb els coneixements obtinguts durant el

desenvolupament, em veig capacitat de desenvolupar aplicacions similars amb major agilitat. Estic content amb el resultat però se que podria ser molt millor.

**Abstract (in English, 250 words or less):**

The purpose of this project with Java EE technologies is to acquire knowledge. In order to acquire this knowledge, a web application based on Spring and ANgularJS has been created, two technologies in constant evolution, but very used.
As the types of businesses that appear today are based mainly on the sale through the web, an application has been developed as an example of business, in this case a greengrocers.
In order to develop correctly, a standard life cycle has been followed, starting with the planning of the project, following this planning, I can adjust the time to delivery goals. Then a functional analysis and requirement, to know the needs of the client, followed by the design of the application.
After the first three phases we can begin to develop and implement the application.
And to verify the correct operation of the application a battery of tests is realized.
To conclude, the entire project is documented, each one of the phases.
Throughout the project, changes have been made to the analysis and requisite documents, because during the develop phase, unexpected doubts and problems arise, as well I have to make changes in design caused by difficulties at the time of the implementation.
The result has been the expected, I was been able to develop a Java-based web application with Spring and AngularJS.
As a conclusion I can say that with the knowledge gained during the development, now i have the ability to develop similar applications with greater agility. I'm happy with the result but I know it could be a lot better.

**Contingut del document**

Aquest treball de final de grau es basa en l'anàlisi, disseny e implementació d'una aplicació web de comerç electrònic. Es tracta d'una aplicació que pot ser aplicada a un comerç real i el seu disseny s'ha desenvolupat amb la intenció de facilitar la reutilització amb la idea de que pugui ser un projecte incremental, del qual s'entrega la primera versió amb les especificacions establertes com a fonamentals per el seu llançament.

El projecte es basa en el paradigma de l'orientació a objectes, en un marc de comunicació en qualsevol tipus de xarxa. Un aspecte interesant ha estat l'implementació de totes les dependències del projecte, les quals per fer-ho de forma automàtica s'ha fet ha traves de Maven, el qual ens ha ajudat a implementar frameworks per la comunicació entre mòduls com Spring d'una forma àgil.

La implementació s'ha basat en el llenguatge Java, s'ha optat per una arquitectura REST basada en el Model Vista Controlador, i per representar la vista s'ha optat per utilitzar Angula bastat en llenguatge JavaScript.
Les tecnologies J2EE ens ha facilitat fer una aplicació robusta, escalable i reutilitzable, característiques molt sol·licitades.

# Index

# 1.Introducció

Durant la realització del grau me vaig interessar en les aplicacions web, tot i que vaig començar l'aplicació fa temps, no tenia coneixements suficients per a acabar de desenvolupar-la.

Per aquest motiu em vaig sentir motivat a formar me ja que aquesta experiència hem pot ajudar molt de cara a un futur, ja que aquí a Mallorca el desenvolupament d'aplicacions web es el mes demandat a les empreses de tecnologies.

L'entorn de l'aplicació esta basat en la relació client-servidor, a on el servidor admet mes d'un client al mateix temps.

En aquest serveis, el client realitza unes peticions al servei per realitzar el conegut com a manteniments, crear, esborrar, modificar les dades de l'aplicació. Per aconseguir això l'aplicació segueix un flux en el qual realitza totes les comprovacions i validacions necessàries per el funcionament de l'aplicació.

Amb aquesta base vaig decidir realitzar una aplicació base la qual es podria aplicar a diversos tipus de negoci. D'una altra banda utilitzar tecnologies que m'ajudin a introduir me al mon laboral amb un poc mes d'experiència com Maven, J2EE, Spring ...

L'abast de les especificacions inicials s'han complert, inclús s'ha aplicat una funcionalitat que comentarem mes endavant. Es tenia intenció d'aplicar tecnologies com JPA, però per mancança de temps ha estat impossible, també es va comentar aplicar geolocalització, s'ha intentat incloure l'api de google per vincular el maps, però aquest api ja no es gratuïta, i vincular un altre implicava un cost en temps massa elevat, el qual faria que no es poguessin complir les fites.

Finalment, estic satisfet amb els coneixements adquirits, però mes amb la capacitat que he adquirit a l'hora de resoldre els problemes que han anat sorgint.

# 2. Objectius del TFG

Com a objectiu principal del TFG m'he proposat familiaritzar-me amb la tecnologia utilitzada (J2EE, Spring, Maven, JavaScript), millorar la meva capacitat de resoldre els problemes que puguin sorgir.

El segon objectiu, ja es mes a nivell d'aplicació, que es realitzar una aplicació de forma àgil, capaç de ser reutilitzada i que sigui escalable.

# 3. Planificació

Per mantenir la planificació estipulada a l'inici del TFG, la feina ha estat constant durant totes les setmanes, per motius de feina, he agut de dedicar mes tems als caps de setmana.

La planificació no s'ha modificat i s'ha seguit el mes estrictament.
La corba de desenvolupament no ha estat lineal, i la planificació estipulada a pics no ha estat acerada, tant per alt com per baix, però al final les hores s'han compensat i s'han complert les fites.

➔TFG ..... 113 Dies – (07-03-2018).

- ➔ PAC 1 ..... 15 Dies – (07-03-2018).
  - ➔Elaboració del pla de treball ..... 14 Dies – (06-03-2018).
  - ➔Entrega PAC 1 ..... 1 Dies – (07-03-2018).
- ➔ PAC 2 ..... 35 Dies – (11-04-2018).
  - ➔ Estudi i analisis ..... 20 Dies – (27 -03-2018).
  - ➔ Disseny ..... 10 Dies – (06 -04-2018).
  - ➔Instalació Entorn ..... 4 Dies – (10-04-2018).
  - ➔Entrega PAC 2 .....1 Dies – (11-04-2018).
- ➔ PAC 3 ..... 49 Dies – (30-05-2018).
  - ➔ Implementació .....40 Dies – (21-05-2018).
  - ➔ Revisió ..... 7 Dies – (28-05-2018).
  - ➔ Entrega PAC 3 ..... 2 Dies – (30-05-2018).
- ➔ Presentació ..... 14 Dies – (13-06-2018).
  - ➔ Preparació Presentació .....13 Dies – (12-06-2018).
  - ➔ Presentació..... 1 Dies – (13-06-2018).

# 4. Producte obtingut

S'ha anomenat FruitStore al producte obtingut. Aquest nom ve de l'utilitat de l'aplicació ja que s'ha enfocat al manteniment del que podria ser una franquícia de fruiteries.

El producte es un aplicació desenvolupada amb tecnologies com J2EE, Tomcat, Spring, Maven i AngularJs.

J2EE es l'especificació Java per a aplicacions de servidor, que proporciona diverses eines per tal de desenvolupar aplicacions orientades cap a client-servidor.

*Tomcat* es un servidor d'aplicacions Java que pot funcionar com servidor web per si mateix.

Spring es un framework de JAVA per el desenvolupament d'aplicacions.

Maven es una eina per gestió i construcció de projectes.

AngularJS es un framework de javaScript que s'utilitza per crear y mantenir aplicacions basades en navegadors.

Els detalls de l'arquitectura i de l'entorn de desenvolupament es comentaran mes endavant.

# 5. Contingut especifica de la memòria

En aquest document hi trobarem altres, els documents que desenvolupen a l'hora de documentar un projecte de desenvolupament de programari.
Aquest documents es creen a mida que avança el projecte i solen ser els següents:

1. Document de Requisits del projecte
2. Document de definició d'actors i casos d'us.
3. Anàlisi dels Requisits de l'aplicació
4. Disseny dels requisits
5. Arquitectura
6. Manual d'usuari

# 6. Especificació i anàlisis dels requeriments.

## 6.1. Introducció.

Aquest apartat recull i documenta els requisits del sistema de programari a implementar. Per tant el seu objectiu defineix detalladament les necessitats d'informació que haurà de resoldre el programari. S'inclou l'anàlisi necessària per a poder traduir els requisits en especificacions més formals (diagrames d'UML) que facilitin posteriorment el desenvolupament de l'etapa de disseny, així com la identificació de les classes fonamentals i l'expressió dels casos d'ús.1

Com es podrà desprendre de la seva lectura, es comença efectuant una aproximació general al sistema mitjançant una anàlisi global d'aquest, per després analitzar el sistema en subsistemes.

## 6.2. Descripció del projecte.

L'empresa Fruits S.A., és una empresa petita, que ha anat creixent molt en poc temps, i per tant desitja actualitzar el seu sistema informàtic per poder adaptar-se a les noves necessitats dels clients i així continuar creixent.

L'empresa esta interessada en una aplicació incremental, capaç de sortir al mercat el mes aviat possible amb les funcionalitats bàsiques i així poder començar a captar clients.

Les primeres necessitats de l'empresa es oferir un servei a traves d'una aplicació web a on qualsevol persona pugui accedir al seus productes. Per començar l'aplicació s'enfoca a una quantitat petita d'usuaris, ja que nomes pot arribar als usuaris propers a les tendes físiques, per tant els costos de computació encara no son el mes important, sinó l'efectivitat i la rapidesa a l'hora d'entregar el producte.

L'empresa disposa d'un servidor a on poder executar i mantenir l'aplicació per tant necessita que aquesta sigui capaç de desplegar-se en aquest servidor i que tant els empleats, com els usuaris han de poder accedir a la mateixa aplicació però amb diferents funcionalitats.

[1] CAMPDERRICH, BENET. *Enginyeria del programari; Anàlisi orientada a objectes* . UOC.

# 7.Requisits Funcionals

L'aplicació permetrà fer el manteniment d'una franquícia de fruiteries fictícia, l'anomenarem FruitStore.

Els usuaris, que pot ser qualsevol persona amb accés a internet mitjançant una interfície web, podran veure les fruites disponibles a la franquícia, el seu nom, una petita descripció i el seu preu, a mes podran veure les tendes de la franquícia, si estan registrats, a més podran veure a quina tenda esta disponible una fruita en concret, per si volguessin anar físicament, o podrien afegir-les al carro, per en un futur poder encarregar-les.

Hi haurà usuaris distribuïdors, aquest seran els propietaris de tendes i/o els seus empleats, els que podran fer el manteniment de les seves tendes, afegint les fruites disponibles en la franquícia a la seva tenda o eliminant les fruites de la seva tenda, també podran afegir noves tendes.

I per últim hi haurà usuaris administrador, encarregats de administrar l'aplicació, que podran afegir noves fruites com disponibles a la franquícia, administrar les tendes afegides per els distribuïdors i els usuaris creats a la aplicació.

Partint d'aquestes indicacions s'han definit els següent requisits.

## Requisits

**Usuari no autentificat**

L'usuari s'ha de poder autentificar.
L'usuari s'ha de poder registrar.
L'usuari ha de poder veure les Fruites.
L'usuari ha de poder veure les Tendes.

**Usuari autentificat**

L'usuari autentificat ha de poder desconnectar-se.

**Usuari autentificat Client.**

L'usuari ha de poder veure les fruites, i veure a quina tenda les pot comprar.
L'usuari ha de poder veure les tendes i seleccionar a quina tenda vol comprar.
L'usuari ha de poder afegir al carro les fruites que vulgui comprar.
L'usuari ha de poder veure el carro.
L'usuari ha de poder descartar el carro.

**Usuari autentificat Distribuïdor.**

L'usuari ha de poder afegir tendes.
L'usuari ha de poder esborrar una tenda afegida per ell.
L'usuari ha de poder veure les tendes afegides per ell.
L'usuari ha de poder editar les tendes creades per ell.
L'usuari ha de poder afegir fruites a les seves tendes.
L'usuari ha de poder eliminar fruites de les seves tendes.

**Usuari autentificat Administrador.**

L'usuari ha de poder fer el mateix que l'usuari distribuïdor però amb totes les tendes del sistema.
L'usuari ha de poder veure tots els usuaris.
L'usuari ha de poder editar els usuaris.
L'usuari ha de poder eliminar els usuaris.

# 8. Definició d'actors i casos d'us.

A continuació s'especifiquen el diferents actors que intervenen en el projecte, així com les seves funcionalitats dins el sistema.

Els usuaris tenen uns rols assignats que els hi donaran accés a diferents funcionalitats.

Es contemplen quatre tipus d'usuaris diferents

| **Actors** | |
|---|---|
| Nom | Funcionalitats |
| Usuari no registrat | Es la persona que mitjançant l'aplicació pot veure les fruites i les tendes de la franquícia |
| Client | Es la persona que mitjançant l'aplicació pot veure les fruites, les tendes i pot afegir fruites al carro per comprar-les en un futur. |
| Distribuïdor | Es la persona que mitjançant l'aplicació pot veure les fruites, crear noves tendes i administrar-les. |
| Administrador | Es la persona que administra l'aplicació, administra les fruites, les tendes i els usuaris. |

Es presenten els diagrames de casos d'us detallats de cada actor amb una descripció a continuació.

## 8.1. Diagrames casos d'us

**Diagrama cas d'us usuari no autenticat:**

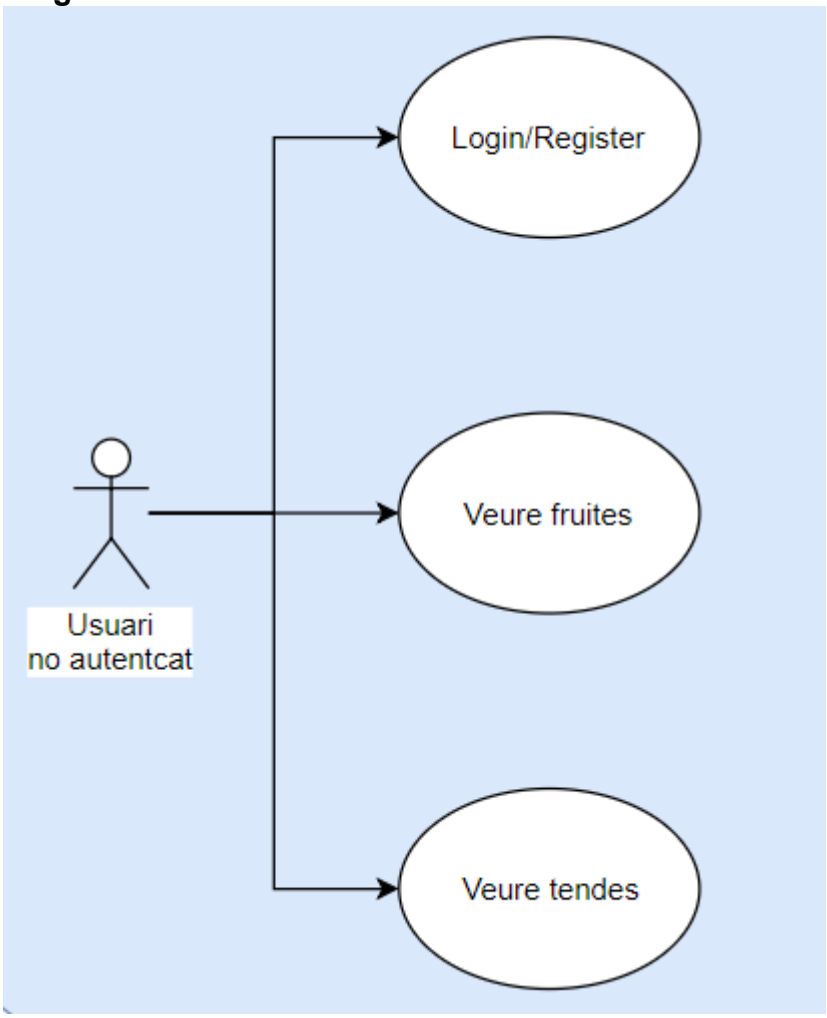


**Diagrama cas d'us usuari autenticat. Client.**

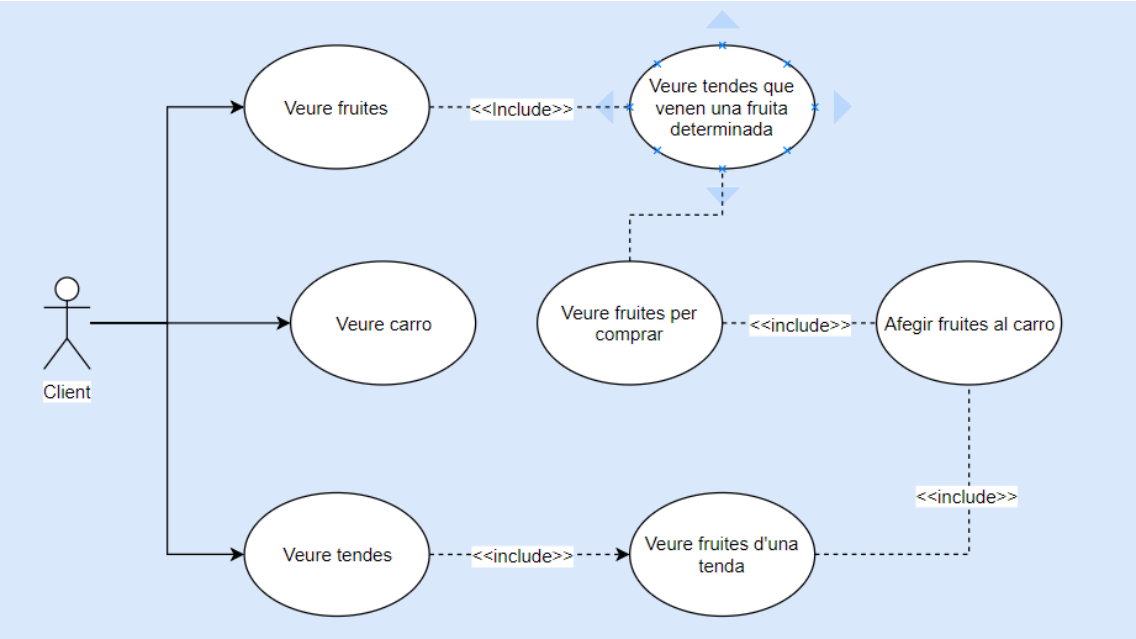



**Diagrama cas d'us usuari autentificat. Distribuïdor.**

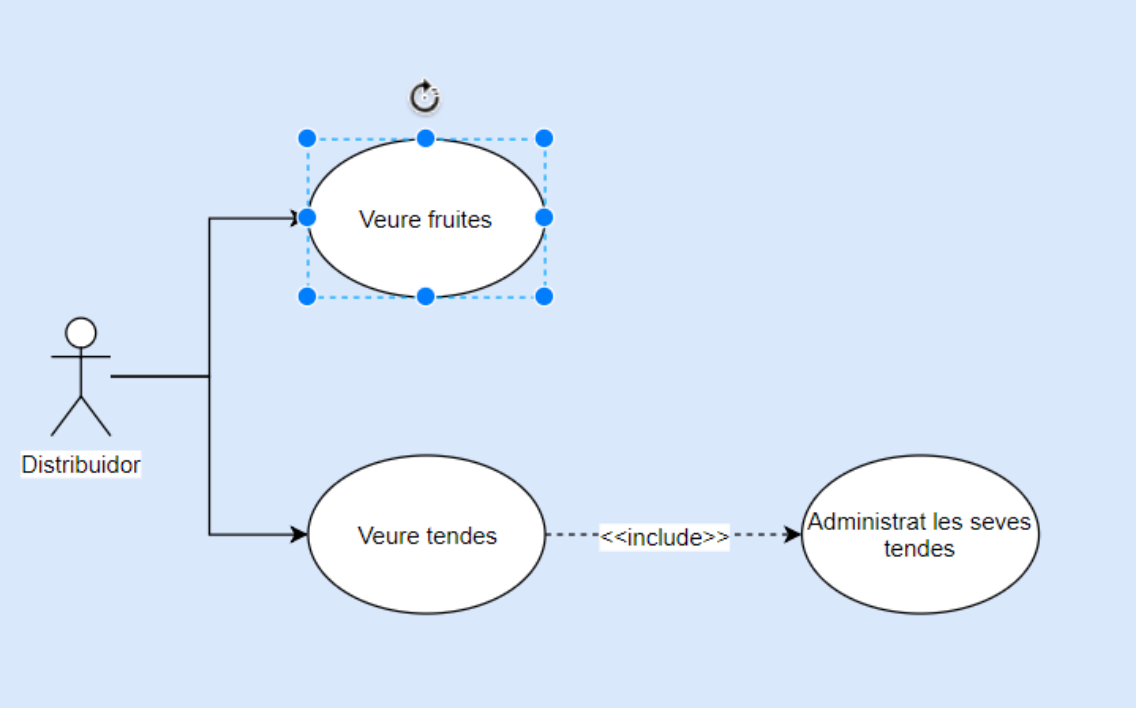
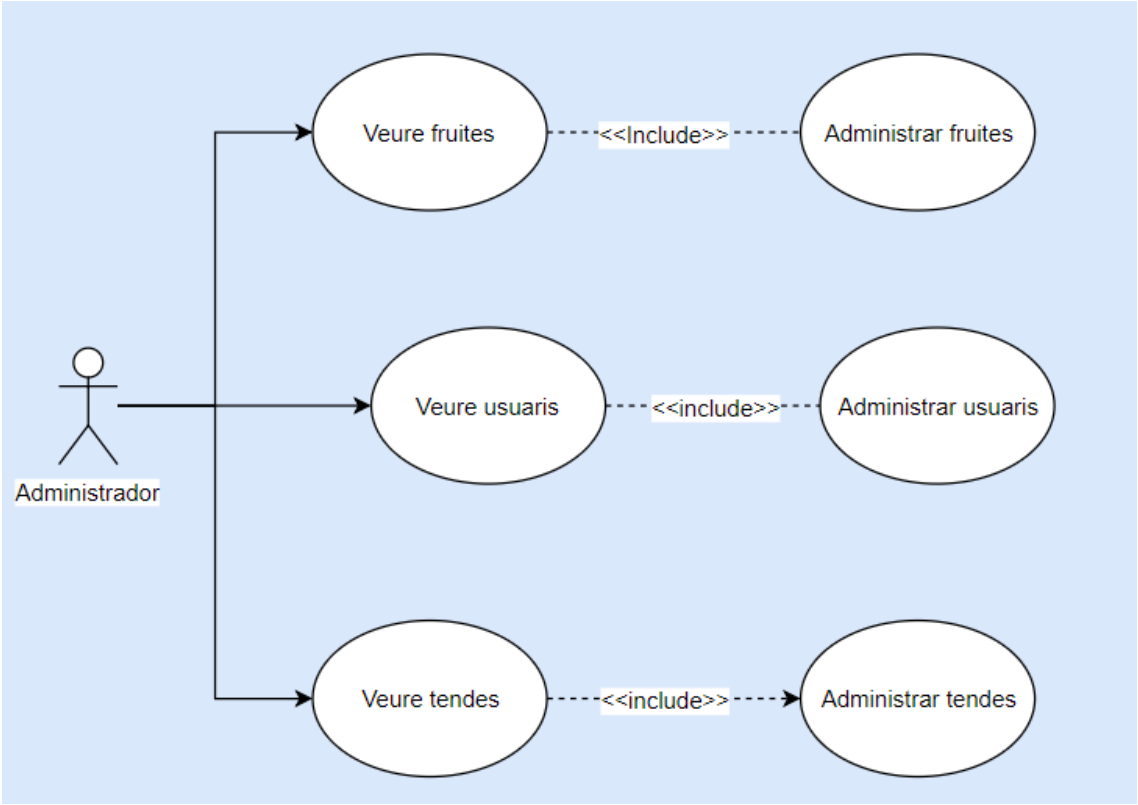


**Diagrama cas d'us usuari autentificat. Administrador.**



## 8.2. Especificació casos d'us.

En aquest apartat es mostrarà un anàlisi mes detallat dels casos d'us separats en tres:
Partim de la base que l'usuari administrador vendrà definit per defecte i aquest es l'únic que pot crear nou usuaris administradors o distribuïdors.

### 8.2.1. Gestió d'usuaris.

**Alta d'usuari**
Cas d'ús: "Register"
Actors: Usuari anònim.
Descripció: L'usuari accedeix al sistema per tal de crear un compte personal a l'aplicació. Fent això haurà d'especificar diverses dades personals.
Condició prèvia: Els usuaris de tipus Administrador i distribuïdor es crearan per un procés diferent.
Cas alternatiu: El sistema detecta que les dades que ha introduït l'usuari són incorrectes i mostra un missatge d'error a l'usuari.

**Modificació dades personals**
Cas d'ús: "Editar datos Usuarios"
Actors: Administrador
Descripció: L'usuari accedeix a l'àrea d'usuaris a on els pot crear, editar, esborrar, veure les seves tendes o afegir-ne.
Condició prèvia: Cal haver fet login a l'aplicació

### 8.2.2. Gestió de Fruites.

**Administrar Fruites**
Cas d'ús: "Administrar Frutas"
Actors: Usuari administrador.
Descripció: L'usuari accedeix a l'area de fruites per tal de afegir,editar o esborrar fruites.
Condició prèvia: L'usuari ha de tenir permisos d'administrador
Cas alternatiu: El sistema detecta que les dades que ha introduït l'usuari són incorrectes i mostra un missatge d'error a l'usuari.

### 8.2.3. Gestió de tendes.

**Administrar Tendes**
Cas d'ús: “Administrar Tiendas“
Actors: Usuari administrador.
Descripció: L'usuari accedeix a l’àrea de tendes per tal de afegir, editar o esborrar tendes, a mes pot afegir fruites que pot tenir la tenda seleccionada.
Condició prèvia: L’usuari ha de tenir permisos d’administrador
Cas alternatiu: El sistema detecta que les dades que ha introduït l'usuari són incorrectes i mostra un missatge d'error a l'usuari.

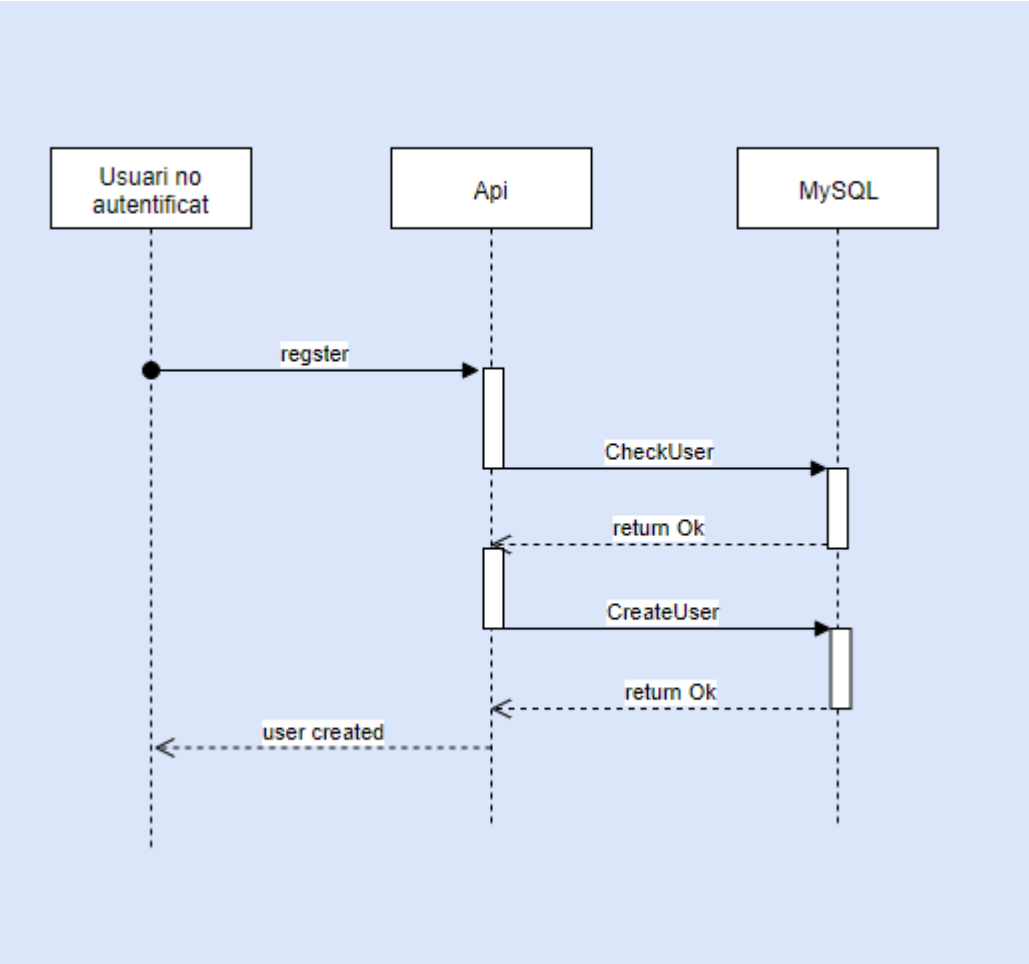
**Administrar Tendes**
Cas d'ús: “Administrar Tiendas“
Actors: Usuari distribuïdor.
Descripció: L'usuari accedeix a l’àrea de tendes per tal de afegir, editar o esborrar tendes, a mes pot afegir fruites que pot tenir la tenda seleccionada, l’usuari nomes veure les tendes que te assignades.
Condició prèvia: L’usuari ha de tenir tendes assignades i permisos de distribuïdor.
Cas alternatiu: El sistema detecta que les dades que ha introduït l'usuari són incorrectes i mostra un missatge d'error a l'usuari.

## 8.3. Diagrames de seqüencies.

En aquest document es recullen dos exemples de diagrames de seqüencia, amb aquest diagrames es pot comprovar que el sistema esta basat en MVC.

Concretament tenim la vista, que es a on l'usuari farà les cridades, aquestes seran enviades a la aplicació, que s'encarregarà de enviar la petició correcta a las base de dades, un cop aquesta torni les dades demanades per l'aplicació, aquesta s'encarregarà de tractar les dades i tornar-les a l'usuari a traves de la vista.

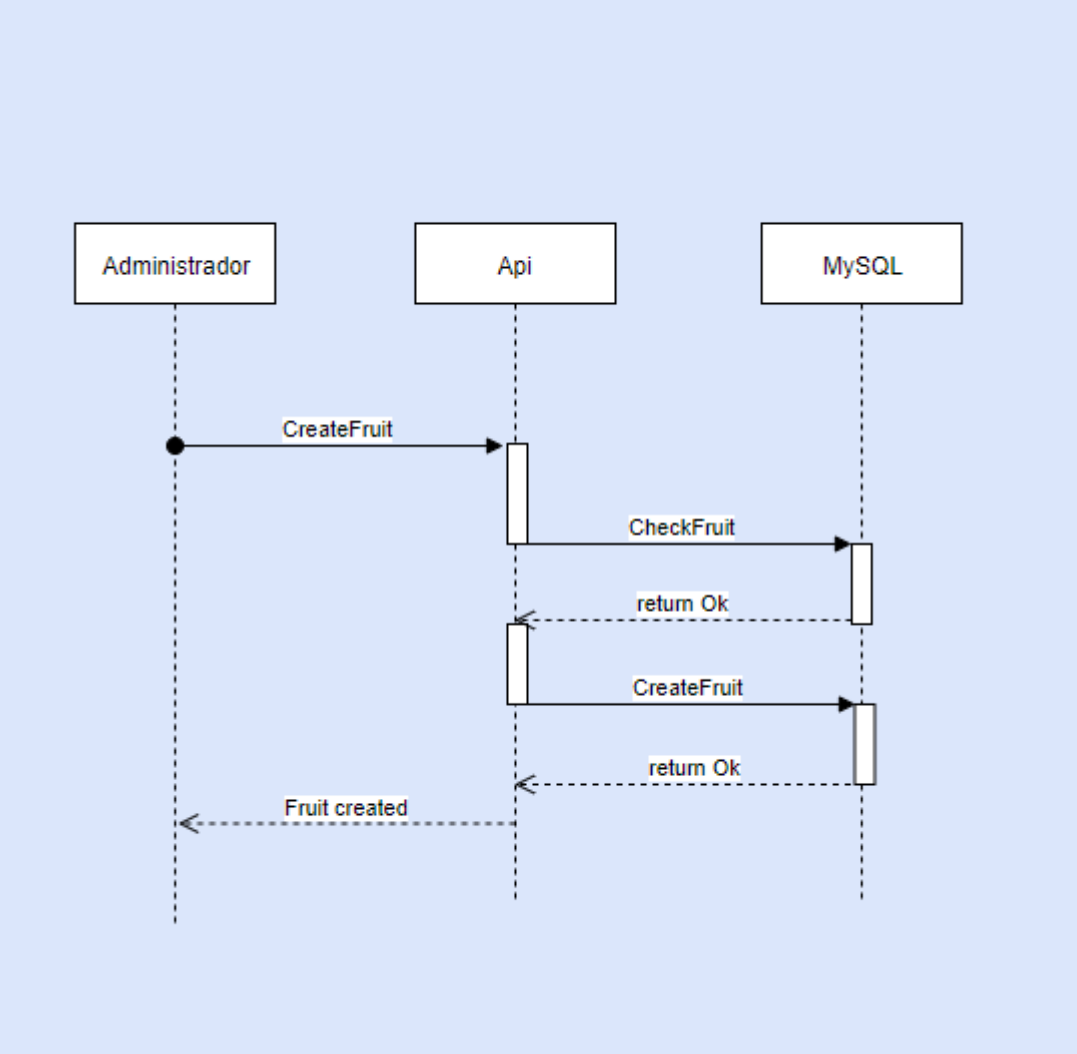
Com que quasi totes les peticions funcionen d'una manera similar, s'ha considerat que amb dos exemples es pot entendre el funcionament del fluxe sense gaire complicacions.




Aquest diagrama de seqüencia representa un usuari que s'enregistra correctament.

Però com s'ha comentat avanç, canviant les cridades i els actors també podria representar qualsevol tipus de creació, modificació o eliminació d'alguna dada del sistema (usuaris, tendes, fruites).

Per exemple la creació d'una fruita:



# 9.Disseny tècnic del sistema

## 9.1. Entitats derivades de l'anàlisi.

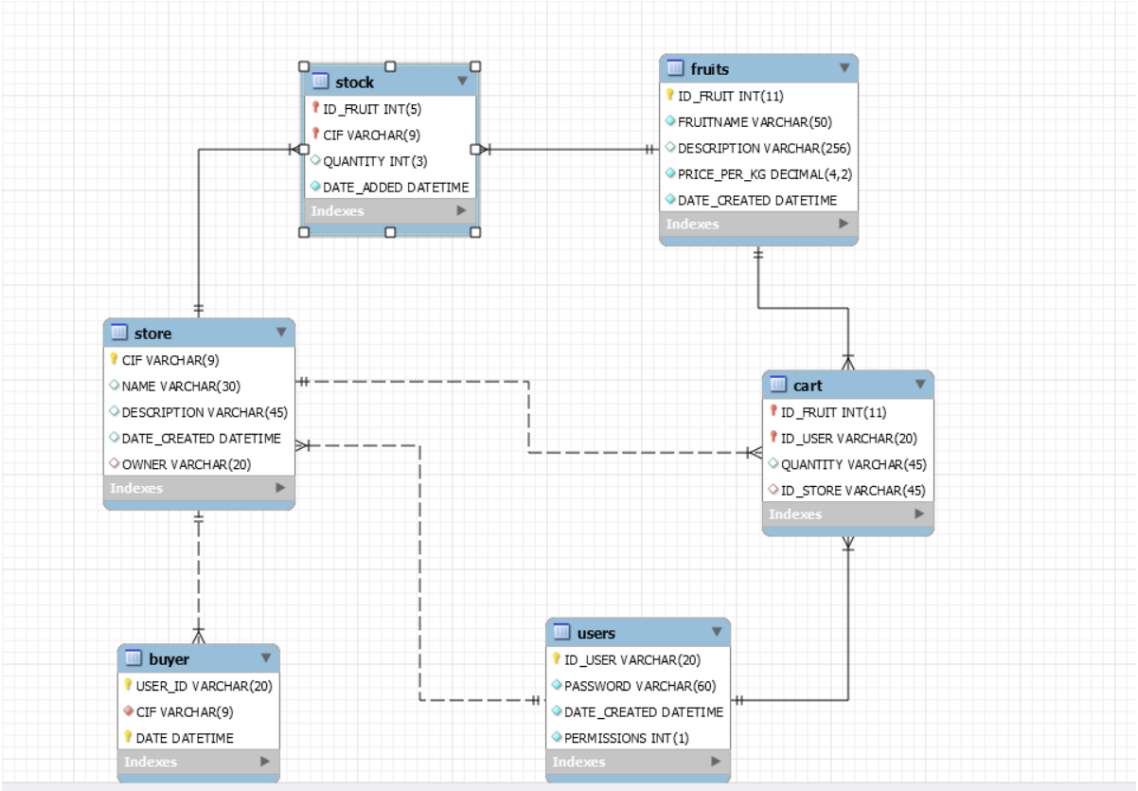
A continuació es dona la llista d'entitats detectades en l'analisi de requisits amb una petita descripció.

- User: Modela un usuari del sistema.
- Fruit: Modela les fruites del sistema.
- Store: Modela les tendes del sistema.
- Stock: Conte les fruites que hi ha a una tenda determinada.
- Cart: Conte les fruites que un usuari vol comprar, i la tenda en la qual vol fer la compra.
- Buyer: Conte un històric de compradors de una tenda.

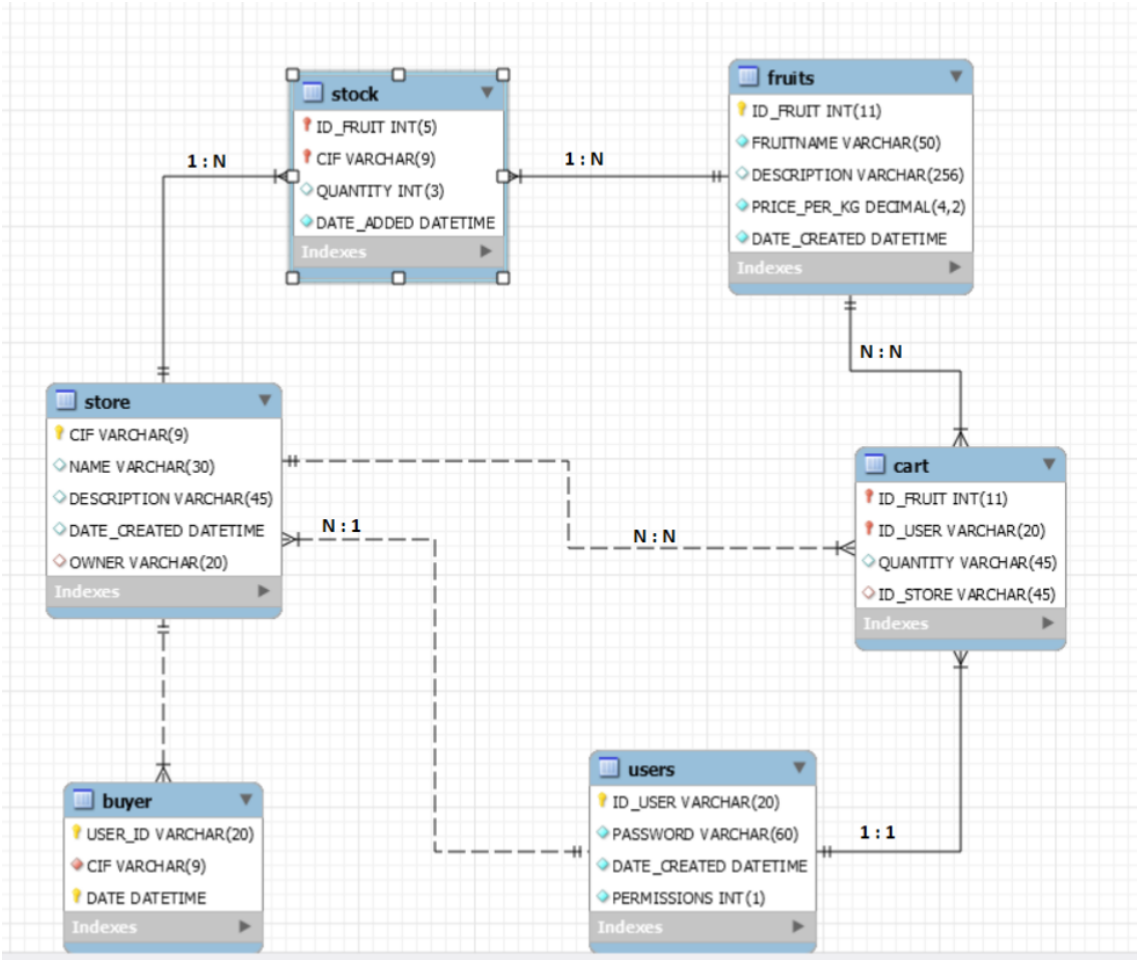
Partint d'aquestes entitats podem definir es model.

## 9.2. Diagrama de Classes.

En funció de les entitats declarades avanç s'ha pogut crear el següent diagrama UML per definir el model.



En el qual podem inserir les relacions les quals també s'han definit en funció dels requisits.



## 9.3. Arquitectura

Un dels patrons més utilitzats en el desenvolupament de aplicacions web es el patró MVC.

Aquest patró va ser originat en el mon SmallTalk i defineix la disposició de les classes, de les seves responsabilitats i les seves relacions com a solució reutilizable.

Com el seu nom indica està dividit en tres tipus de components en funció de la seva responsabilitat en el sistema:

- El Model: Es el component de dades i lògica de negoci. Aquest model de dades pot tenir diverses vistes (interfícies).

- La Vista: Es el component de presentació de dades a l'usuari. Una mateixa vista pot donar diferents presentacions de dades als usuaris. Rep dades del model que mostra a l'usuari.

- El Controlador: Es el component que respon a les entrades de l'usuari, tradueix events d'interfície d'usuari en crides al Model i defineix com reacciona la interfície d'usuari amb aquest canvis.

Fent servir aquest patró s'obtenen grans avantatges:

- Les aplicacions son més flexibles davant futurs canvis. De fet, permet modificar completament el disseny d'una aplicació web, sense que això afecti al Model. O fins i tot, realitzar canvis d'implementació a la lògica i al Model, sense que sigui necessari modificar el disseny de la interfície d'usuari.

- Els programadors i els dissenyadors gràfics, o els maquetadors poden treballar independentment, tan sols respectant un mínim contracte.

- La separació de la lògica d'empresa i la de presentació facilita en gran part la documentació del codi.

- La lògica de negoci no haurà de dependre mai de components de comunicació a l'usuari o a tercers actors. Per exemple, no tindrà mai necessitats de coneixement de protocol HTTP, o de qualsevol altre, permetent la seva execució des de qualsevol entorn que implementi el model de Vista i Controlador.

L’arquitectura del projecte a nivell de controlador, està basada en l’estil REpresentational State Transfer (REST), que recolza totalment en l’estàndard HTTP.

Això ens permetrà crear l’aplicació que podrà ser utilitzada per qualsevol dispositiu que pugui fer feina amb HTTP, es a dir qualsevol dispositiu amb un navegador web.

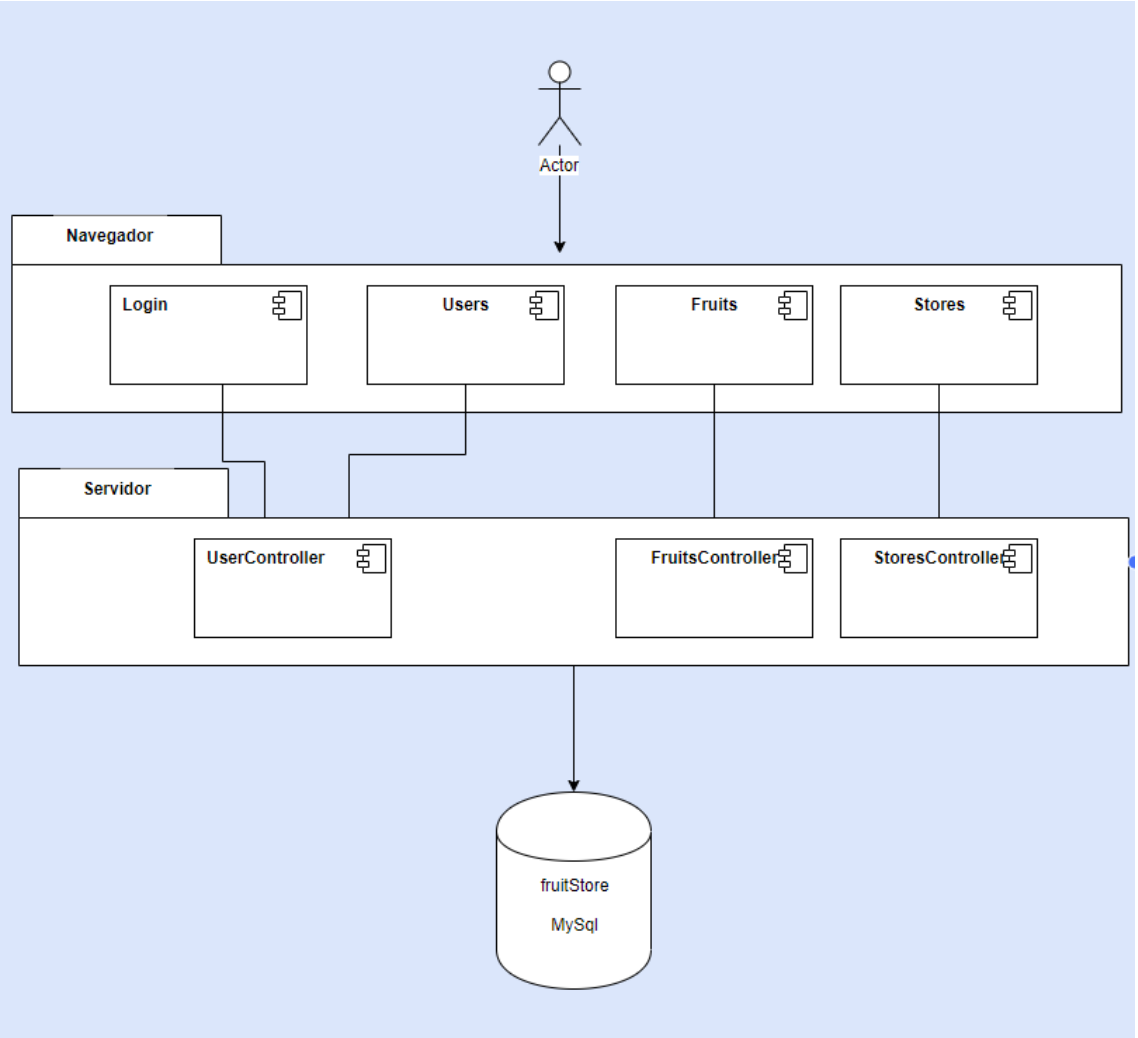
Utilitzarem un framework anomenat Spring, el qual s’encarregarà de la comunicació entre les diverses capes de l’aplicació i facilita l’estandardització.

La implementació de la persistència s’ha fet amb java, que te llibreries de connexió a base de dades.

Per definir el model i emmagatzemar les dades farem servir el servidor de base de dades MySQL, el qual podem connectar a l’aplicació mitjançant un driver propi de aquest servidor, es una base de dades relacional gratuïta que qualsevol pot utilitzar, com el seu nom ens indica s’utilitza amb llenguatge SQL, com quasi totes les bases de dades.

Per acabar la vista, aquesta ha estat definida amb AngularJS que es un framwork de JavaScript, aquest framework també utilitza una arquitectura MVC, es a dir te uns controlador que son els que s’encarreguen del tractament de dades, tenen una vista basada en HTML, i un model, que en aquest cas, el model seria la nostra REST.

El següent diagrama es un petit resum de les especificacions mencionades:



## 9.4. J2EE, Maven i Tomcat

A continuació se presenta una llista amb tecnologies utilitzades per desenvolupar el sistema que no s'han descrit.

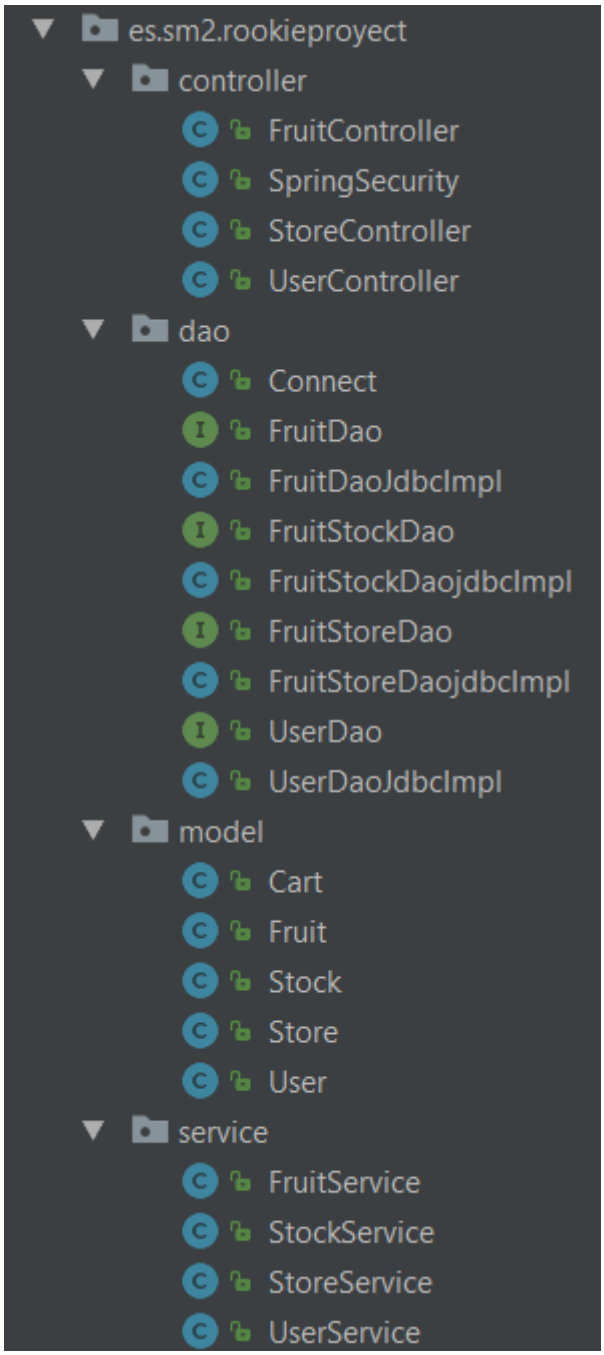
La primera elecció ha estat J2EE com a llenguatge i plataforma de programació, Java es un llenguatge obert, en constant evolució i molt actualitzat, es uns dels llenguatges mes utilitzats actualment i ens proporciona molta documentació accessible ,portabilitat i robustesa.

A l'hora de desplegar l'aplicació necesitam un bon contenidor, en aquest cas hem elegit Tomcat ja que es gratuït, a mes esta basat en llenguatge Java, per el que es totalment compatible amb la nostra aplicació i es pot fer servir en qualsevol maquina amb una maquina virtual Java.

A més, a l'hora de desenvolupar necessitem diverses dependències, com les de Spring, per això les hem d'importar al projecte, si hem de seleccionar totes les dendéncies, descarregar-les e importar-les al projecte podem perdre un temps valuós, a mes cada pic que vulguem desplegar el projecte en una altra maquina haurem de copiar aquestes dependències, el que pot augmentar el pes del projecte, per gestionar la construcció del projecte utilitzarem l'eina Maven, la qual s'encarrega de gestionar les dependències i/o els pluguins necessaris per executar el projecte, nomes necessitem connexió a internet per que pugui descarregar tot el que necessiti el projecte.

## 9.5. Estructura del projecte

El projecte te en gran mesura javaDoc, que es l'estàndard de comentaris dins el codi de l'aplicació, això facilita la feina a terceres persones. Si una persona que no coneix la funcionalitat del projecte, intenta editar o modificar el codi, li resultarà mes fàcil ja que els mètodes estan comentats explicant molt breument la funcionalitat.
L'estructura de la REST es aquesta:




Es pot comprovar que es basa en l'arquitectura MVC.

Tenim un controlador que s'encarrega de les peticions, el model que conte les entitats, un servei que s'encarrega de tractar les dades, el dao, que conte la connexió amb la base de dades. Ens faltaria la vista, però aquesta se comenta un poc mes endavant.

A continuació se mostren uns exemples de classes java a on podem veure que conte anotacions Spring per al seu correcte funcionament.

Un exemple de controlador:

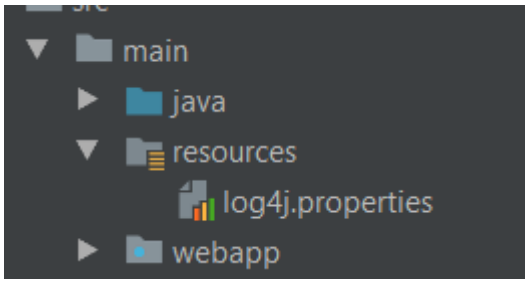
```java
@Component
@Path("/logRest")
public class UserController {

    @Autowired
    private UserService userService;


    @Path("/log")
    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    public Boolean loginUser(@RequestBody User user) { return userService.login(user); }


    /**
     * Method that get all the users in the table
     *
     * @return json, json with all the users in the table
     */
    @Path("/users")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public String getTable() {

        String json = new Gson().toJson(userService.getList());
        return json;

    }
```

Exemple de entitat:

```java
public class User {

    private String name;
    private String password;
    private String dateTime;
    private int permissions = 3;

    public User() {
    }

    public User(String user, String password) {
        this.name = user;
        this.password = password;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public void setDateTime(Date date, Time time) { this.dateTime = time.toString() + " " + date.toString(); }

    public int getPermissions() { return permissions; }

    public void setPermissions(int permissions) { this.permissions = permissions; }
```

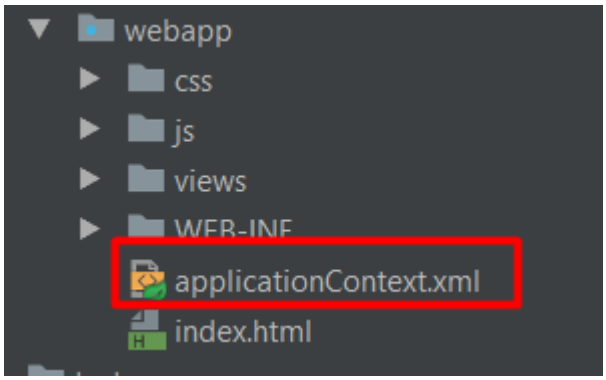
Exemple de servei:

```java
@Component
public class UserService {

    @Autowired
    @Qualifier("UserDaoImpl")
    UserDao userDao;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public User user;

    public boolean login(User user) {
        String hashPassword = userDao.get(user.getName()).getPassword();
        String password = user.getPassword();
        this.user = getUser(user.getName());
        boolean userTrue = this.user.getName().equals(user.getName());
        boolean passwordTrue = passwordEncoder.matches(password, hashPassword);
        if ((userTrue) && (passwordTrue)) {
            return true;
        } else return false;
    }
```



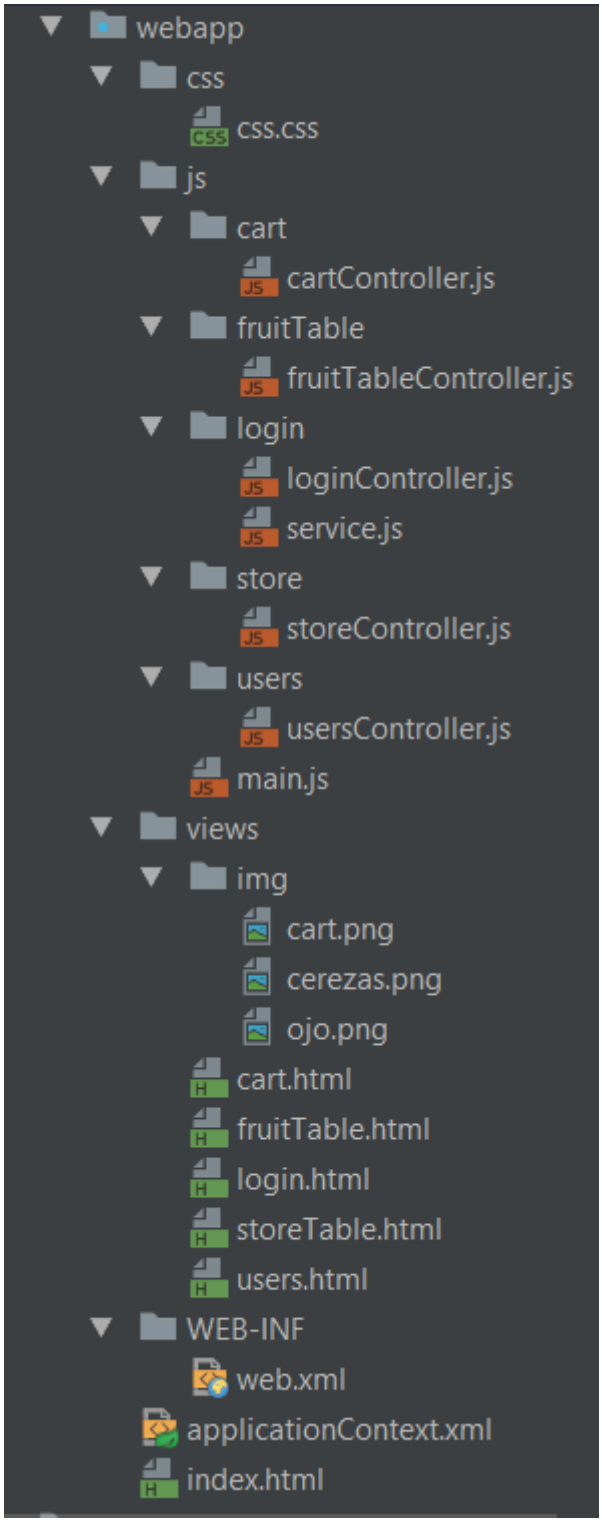

A la carpeta de resource, s'inclouen els fitxers de properties, que s'utilitzen per la configuracions de l'aplicació, en aquest cas per el systema de logs de l'aplicació, que s'ha implementat amb log4j.

Per que funcionin les anotacions d'Spring, s'ha de crear els beans en un document xml anomenat applicationContext.xml, de la següent manera:




```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"
       xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context">

    <context:component-scan base-package="es.sm2.rookieproyect"/>

</beans>
```

L’estructura de la vista es aquesta.




Un altre vegada tenim present el model vista controlador.
En aquest cas, tenim les vistes, i els controladors que ens ajuden a tractar les dades, el model ens vendra definit per la REST

A continuació s'expossen exples de la vista.

Exemple d'un controlador de la vista:

```
app.controller("usersController", function ($scope, $http, $window, $location, userSelectedService, userService, storeService )
    var url = "http://localhost:9090/rest/logRest";

    $scope.store = storeService.store;

    $scope.getUsers = function () {
        $scope.userLoged = userService.user;
        $http.get(url+"/users" ).success(function(data){
            $scope.users=data;
        })
    };


    /**
     * Method that redirects to update, insert or remove
     * @param method, the param that indicates what we have to do
     * @param user, the user we want to update, delete or insert
     */
    $scope.updateUser = function (method, user) {
        $scope.edit = true;
        if (method == 'new') {
            $scope.edit = false;
            $scope.name = '';
            $scope.password='';
            $scope.dateTime = '';
            $scope.permissions = ''
        } else {
            if (method == 'delete') {
                $scope.user = user;
                $scope.edit = false;
            } else {
                if (method == 'deleteAll') {
                    $scope.user = user;
                    $scope.edit = true;
```

El servei utilitzat a la vista:

```
app.factory("userService", function() {

        return {
            user: {}
        };

});

app.factory("storeService", function() {

    return {
        store: {}
    };

});


app.factory("userSelectedService", function() {

    return {
        user: {}
    };
});

app.factory("cartService", function () {

    return {
        cart: {}
    };

});
```

Un exemple d'un html:

```html
<!DOCTYPE html>

<html>
<title>AngularJS Routing</title>

<body>

<div ng-app="myFruitsList" ng-controller="mainController">
    <nav class="w3-sidenav w3-light-grey w3-card-2" style="...">
        <div class="w3-container w3-teal">
            <h4>Menu</h4>
        </div>
        <a href=#login" ng-click="unSelectUser()">Login</a>
        <a href="#fruitTable" ng-click="unSelectUser()">Fruits</a>
        <a href="#storeTable" ng-click="unSelectUser()">Stores</a>
        <a ng-show="userLoged.permissions == 1" href="#users">Users</a>
        <a href="#cart" ng-show="userLoged.permissions == 3" ng-click="unSelectUser()">Cart</a>

    </nav>

    <div ng-view>
    </div>

    <link rel="stylesheet" href="http://www.w3schools.com/lib/w3.css">
    <link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css">
    <link rel="stylesheet" href=css/css.css>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/ng-password-strength.min.js"></script>


    <script src="js/main.js"></script>
```

I a continuació, el fitxer que utilitza Maven per importar les dependències:

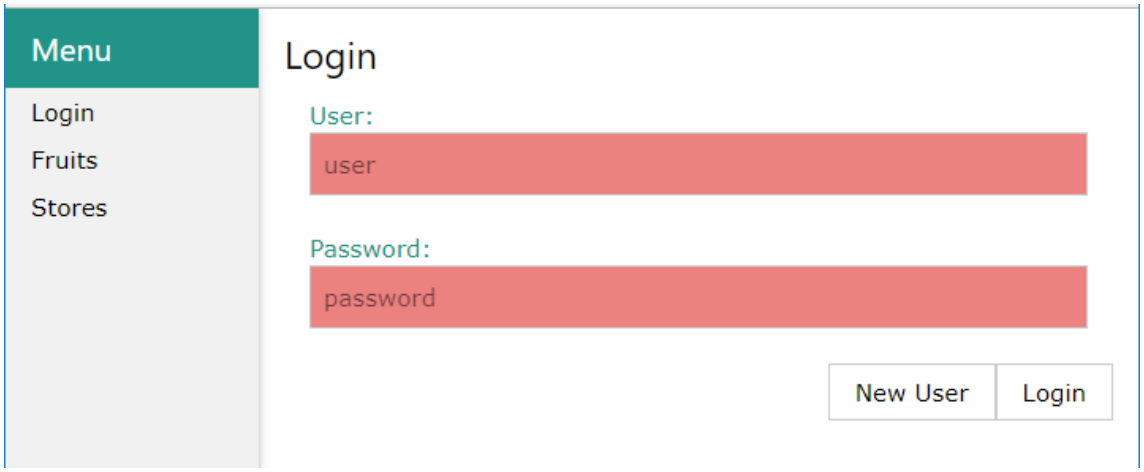
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>RestAngular</groupId>
    <artifactId>RestAngular</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>


    <dependencies>
        <!-- The spring-web module provides basic web-oriented integration features
              such as multipart file upload functionality and the initialization of the
              IoC container using Servlet listeners and a web-oriented application context -->

        <!-- Spring dependencies -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>4.3.1.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.3.1.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>4.3.1.RELEASE</version>
        </dependency>
```

A mes amb aquest document, s'adjuntara un fitcher anomenat Script.sql el qual executarem a la nostra base de dades per generar les taules.

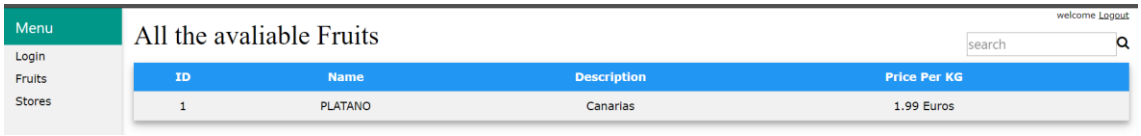
## 9.6. Pantalles

### 9.6.1. Usuari no autenticat:

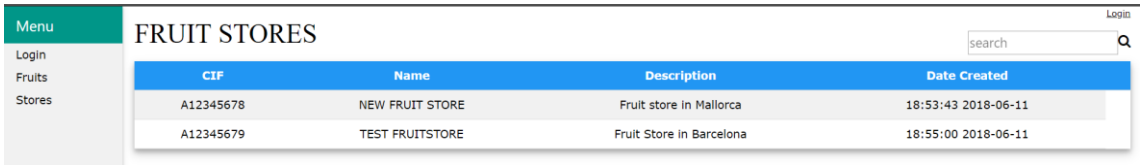
**Login:**



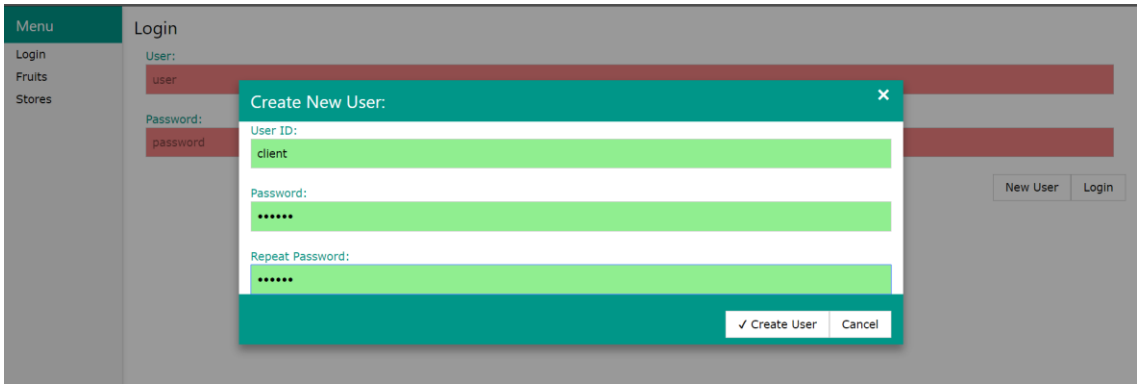

**Fruits:**




**Stores:**

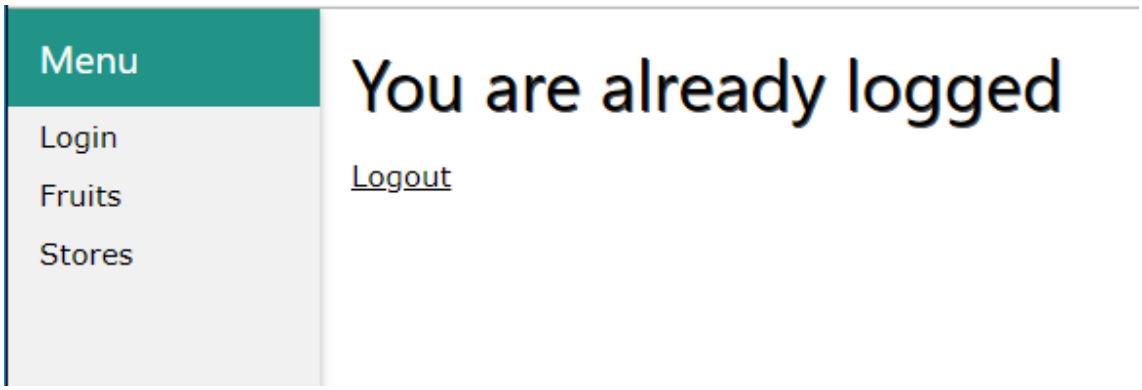



**Create User:**

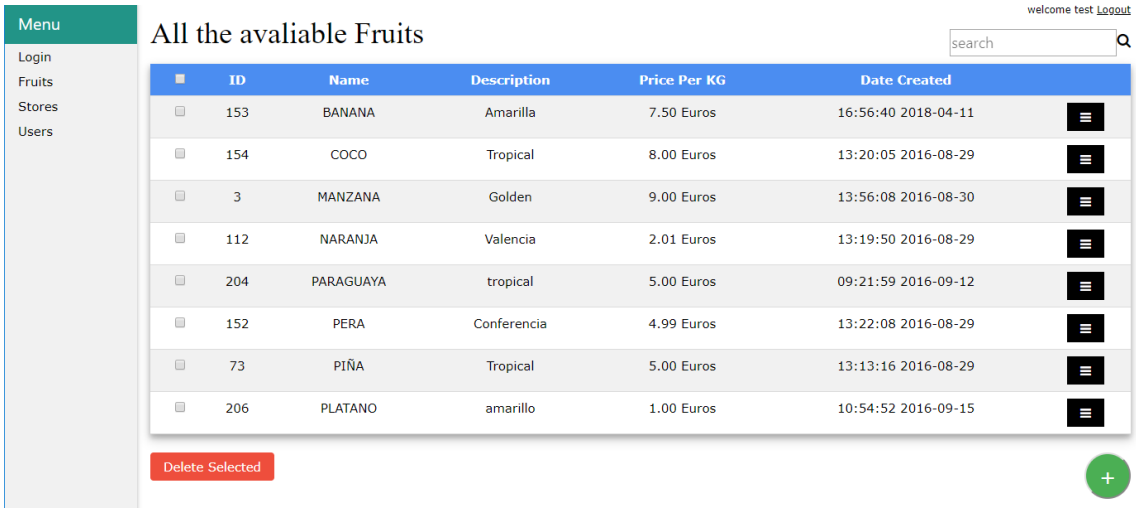


### 9.6.2. Usuari Administrador

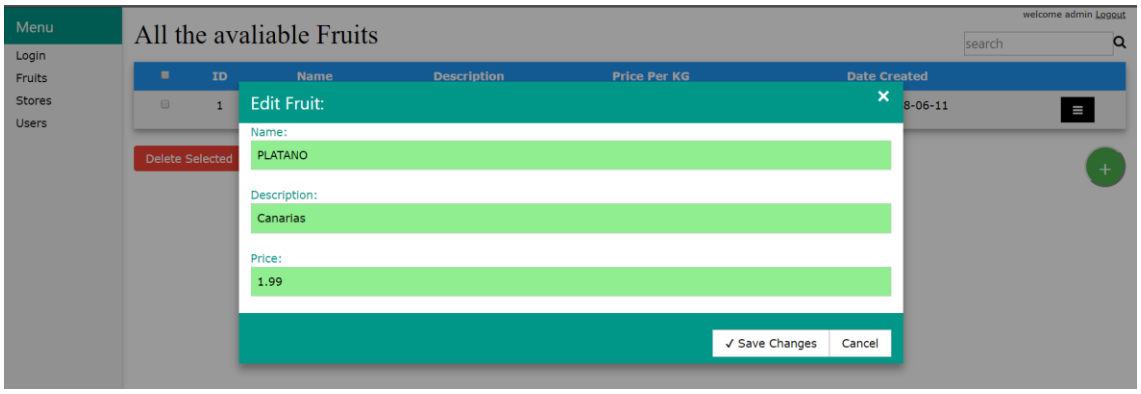
**Login:**



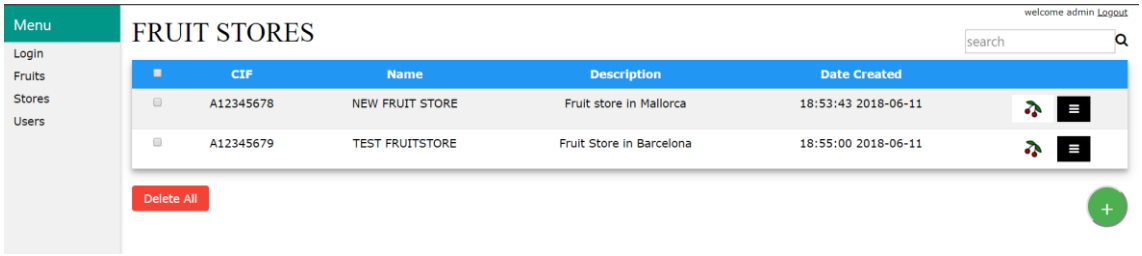

**Fruits:**




**Edit/Add Fruit:**

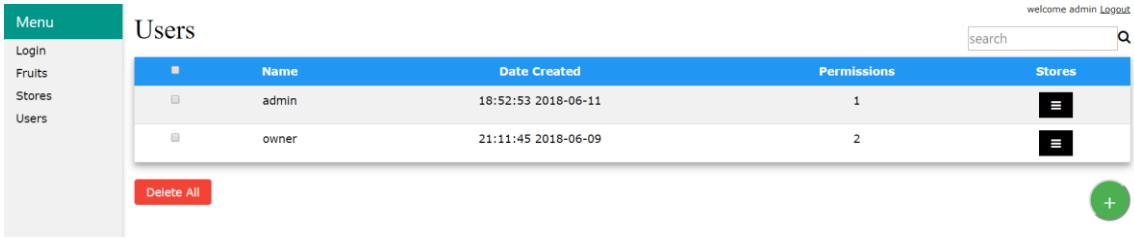


## Stores:

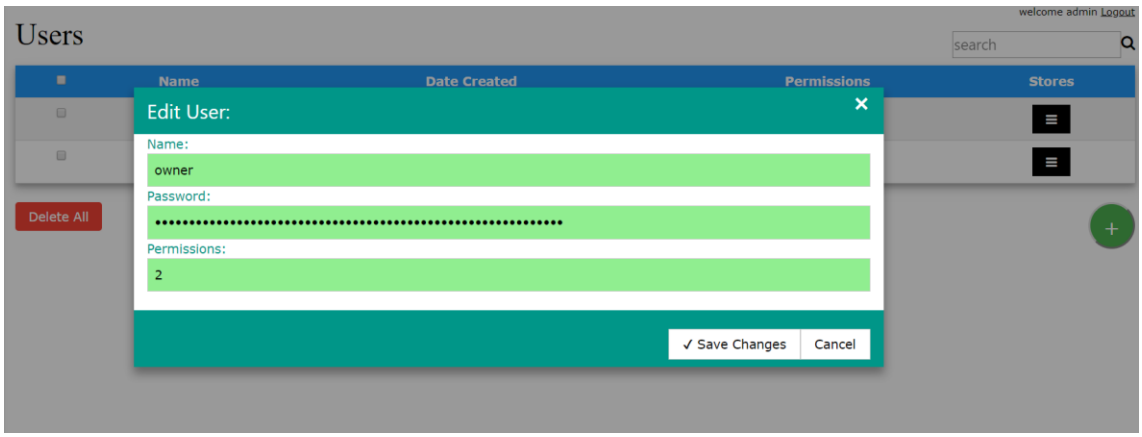



## Edit/Add Store:


## Users:




## Edit User:



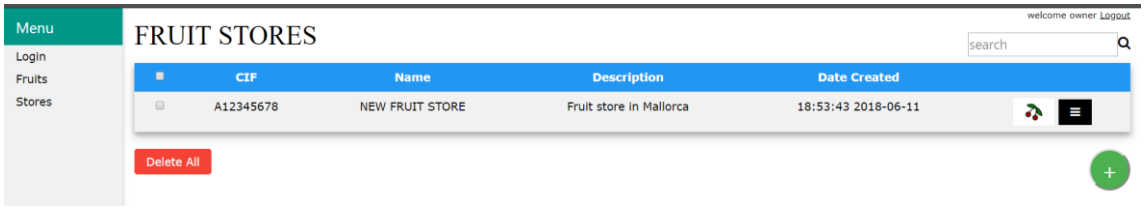
### 9.6.3. Usuari Distribuïdor

**Add fruits to Store:**


**Fruits in Store:**

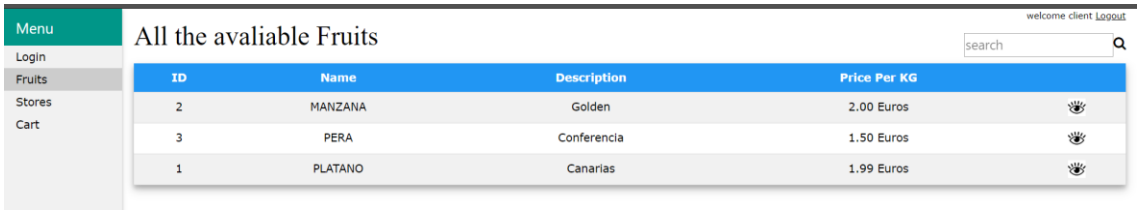

**Edit/Add Store:**


**Stores owned:**

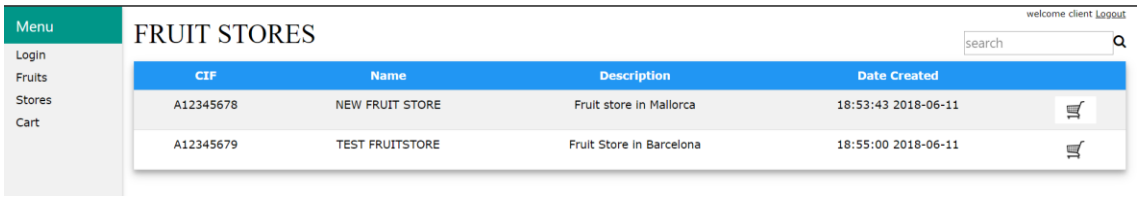


### 9.6.4. Usuari Client.

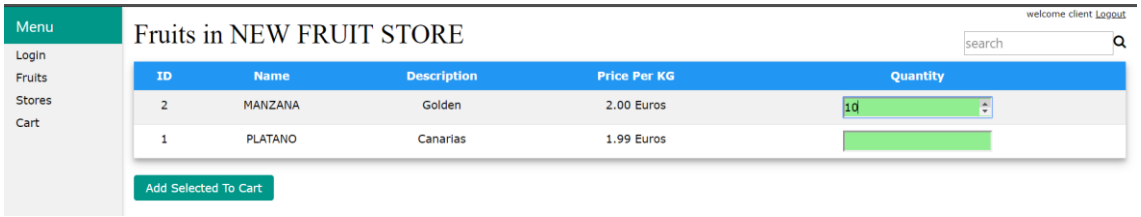
**Fruits:**



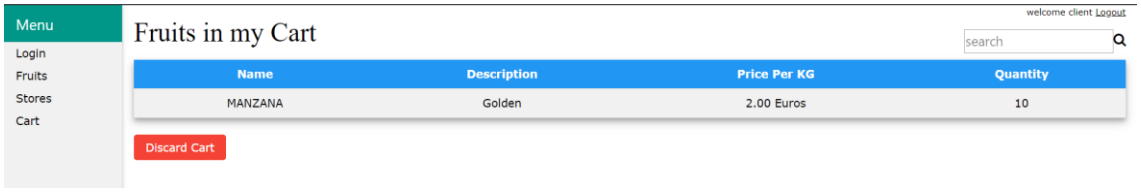

**Stores:**




**Fruits of store:**

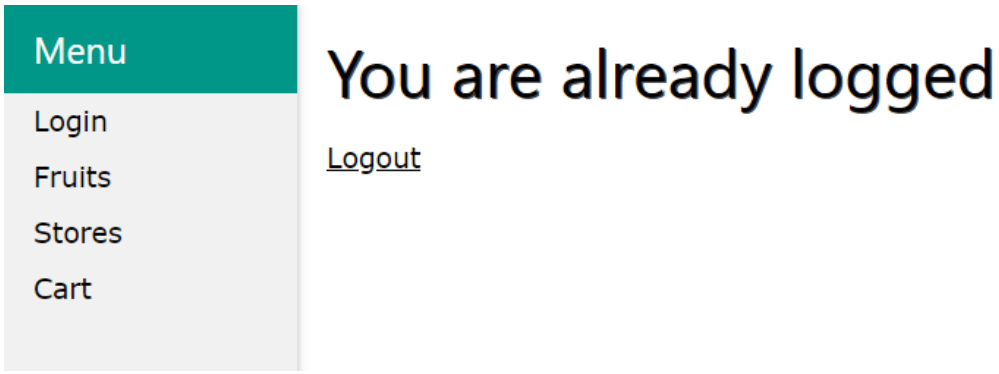



**Cart:**



### 9.6.5. Usuari autenticat

**Login:**



# 10. Entorn de feina.

Per desenvolupar l'aplicació se ha utilitzat el següent entorn:

- Windows 10.
- Java 8.
- Angular 1.4.3.
- IntelliJ IDEA: Ide per realitzar la codificació.
- MySQL Workbench 6.3: per gestionar la base de dades.
- Apache Maven 3.3.9: S'utilitza per construir el projecte.
- Draw.io: per realitzar els diagrames
- Icecream Screen Recorder i Aimersoft Video Editor: per grabar el vídeo i editar-ho.

# 11. Millores futures.

Com que hi ha un temps limitat per la realització del projecte s'han implementat les funcionalitats esmentades al principi, pero això no vol dir que l'aplicació estigui al 100%.

L'aplicació permet moltes millores, tant funcionals com de rendiment, si en un futur, aquesta empresa creix molt, necessitarà adaptar l'aplicació al nivell d'usuaris. Això se pot fer tractant millor les dades, i paginant les peticions a base de dades.

A mes la funcionalitat de compra no esta finalitzada, s'ha creat un carro, però es podria incloure una passarel·la de pagament.

També es pot millorar la seguretat, ara mateix te una contrasenya xifrada amb Spring, però nomes es bloquetja l'aplicació en la vista, si s'accedeix a l'aplicació des d'altres llocs que consumeixin apis amb HTTP i es coneixen les peticions no hi ha cap tipus de seguretat.

# 12. Conclusions.

Realitzar el TFG m'ha ajudat a adquirir molts coneixements nous. Vaig elegir Java EE com a base del meu projecte ja que es un tecnologia molt utilitzada a Mallorca, i volia adquirir les bases i el coneixement suficient per desenvolupar una aplicació des de zero.

L'elecció de les tecnologies pot ser podria estar mes encertada, ja que tant AngularJS com Spring 4 estan un poc des actualitzats, encara que per la finalitat del projecte eren suficients, ja que Angular esta basat en JavaScript, que es un dels llenguatges mes utilitzat de cara a representació de les vistes, i Spring encara que en versions mes actuals, es un dels frameworks mes utilitzats per java avui en dia.

He trobat la part de la documentació molt complicada, i extensa, si que es molt necessària a l'hora de desenvolupar un programari, però la part que suposa un repte es la de desenvolupar i per tant, la que mes sol agradar.

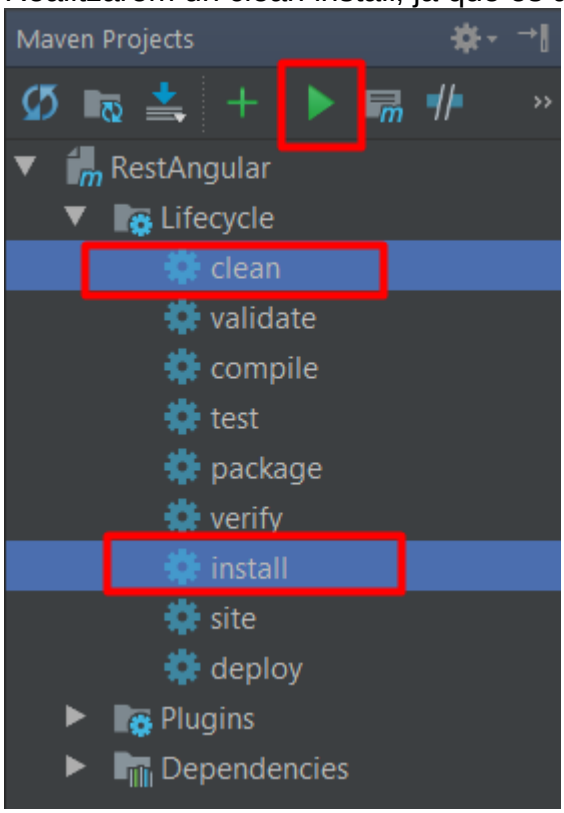
Desenvolupar tot sol també ha suposat un repte, ja que tots els problemes que han anat sorgint els he agut de resoldre, amb la documentació que he pogut trobar, i si ha estat necessari amb el consultor.

En general estic satisfet amb l'experiència, pot ser l'aplicació es molt senzilla, però per jo ha suposat un esforç bastant elevat per la manca de coneixements. Encara que per a futurs desenvolupaments, aquesta ha estat una experiència molt valuosa i que necessitaré.

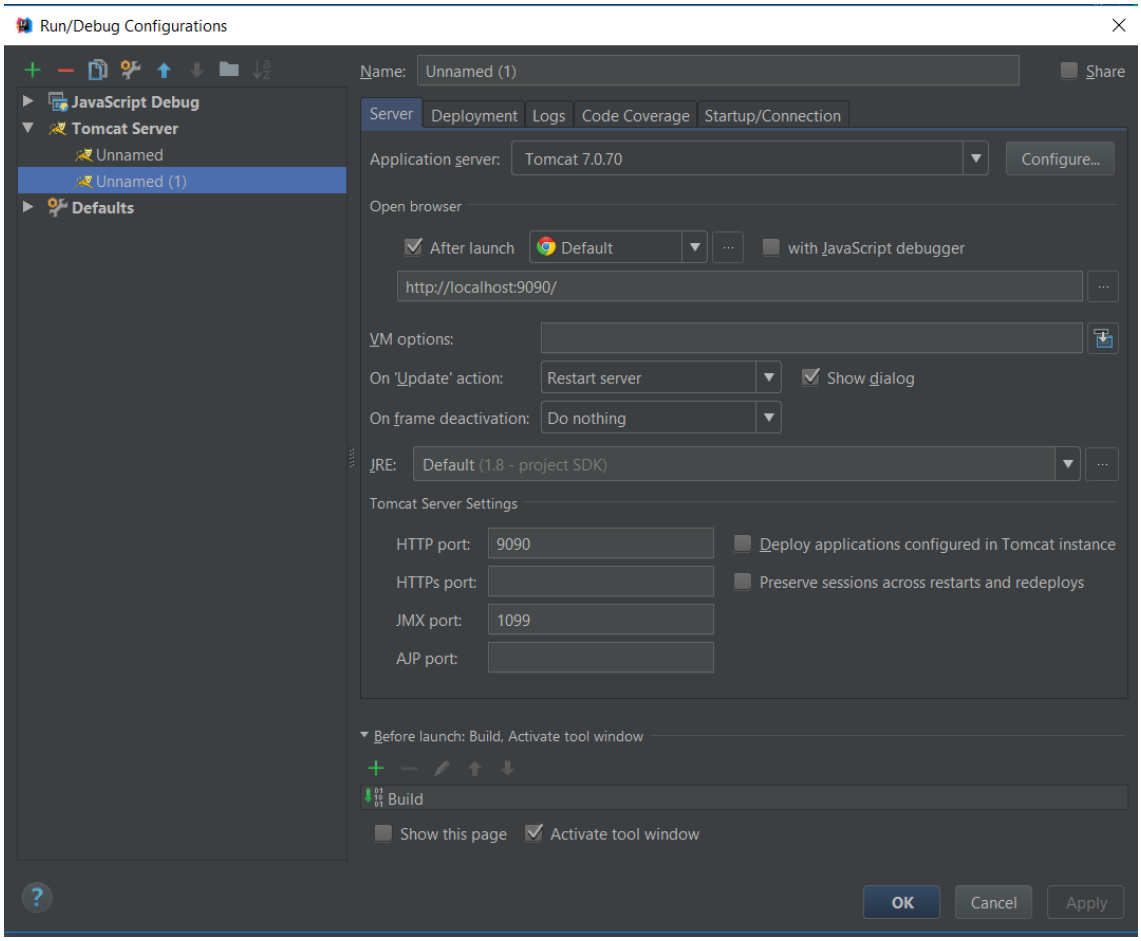
# 13. Manual d'aplicació

En aquest punt s'explica com executar l'aplicació ja que les funcionalitats es mostren a les imatges anteriors amb les pantalles.

1. El primer que hem de fer es obrir el projecte amb un entorn de desenvolupament (IDE), en aquest cas IntelliJ IDEA
2. A continuació editar la configuració, si no ve per defecte, aplicar la versió de java i de maven al projecte.
3. Realitzarem un clean install, ja que es un projecte maven.




4. Crearem una configuració per executar l'aplicació basada en Tomcat server i ja podrem executar el projecte.




L'aplicació es desplegara en la ruta http://localhost:9090/#/login .

# 14. Bibliografia.


- Intellij IDEA - https://www.jetbrains.com/idea/
- Spring - https://spring.io/
- AngularJS - https://angularjs.org/
- Apache Tomcat - http://tomcat.apache.org/
- Draw.io - https://www.draw.io/
- MySQL - https://www.mysql.com/
- Apache Maven - https://maven.apache.org/