

Gestió d’una falla

Projecte final de cicle formatiu



Joel

fuster

BOSCH

|  |  |
| --- | --- |
| **Nom i cognoms** | Joel Fuster Bosch |
| **NIF/NIE** | 020942567D |
| **Curs i CF** | 2n DAM |

|  |  |
| --- | --- |
| **Títol del projecte** | Gestió d’una falla |
| **Nom del tutor individual** | Juan Bautista Talens |
| **Nom del tutor del grup** | Jose Alfredo Murcia |
| **Resum** | *Gestió d’una falla que permet als usuaris fer compres amb les seues polseres de la falla amb tecnologia NFC (Near Field Communication)*. |
| **Abstract** | *Management of a Falla-making committee that provides the users doing shopping with their bracelets of the Falla-making committee with NFC technology (Near Field Communication).* |
| **Mòduls implicats** | * SGE (Sistemes de gestió empresarial) * PMDM (Programació multimèdia i dispositius mòbils) |
| **Data de presentació** | 2 de juny de 2025 |

Contents

[1.Introducció del projecte 4](#_Toc199325710)

[1.1Descripció del projecte 4](#_Toc199325711)

[1.2 Objectius del projecte: 4](#_Toc199325712)

[2.Tecnologies i ferramentes 4](#_Toc199325713)

[2.1 Tecnologies usades per al projecte 4](#_Toc199325714)

[2.1.1 Flutter 5](#_Toc199325715)

[2.1.2 Dart 6](#_Toc199325716)

[2.1.3 VSCode 7](#_Toc199325717)

[2.1.4 Github 7](#_Toc199325718)

[2.1.5 Figma 8](#_Toc199325719)

[2.1.6 Node.js 9](#_Toc199325720)

[2.1.7 PostgreSQL 9](#_Toc199325721)

[2.1.8 Odoo 10](#_Toc199325722)

[2.2 Tecnologia NFC 10](#_Toc199325723)

[2.3 Codi QR 12](#_Toc199325724)

[3. Diseny de la solució 14](#_Toc199325725)

[4. Desenvolupament de la solucio 14](#_Toc199325726)

[4.1 Consideracions prèvies 14](#_Toc199325727)

[4.2 Configuració i desenvolupament del sistema 15](#_Toc199325728)

[4.2.1 Configuració de Flutter i el seu entorn de desenvolupament 15](#_Toc199325729)

[4.2.1.1 Instal·lació de Flutter i les seues dependències 15](#_Toc199325730)

[4.2.1.1 Instal·lar l’entorn de desenvolupament 15](#_Toc199325731)

[4.2.2 Configuració del backend(Servidor de node.js) 15](#_Toc199325732)

[4.2.3 Arquitectura CLEAN 16](#_Toc199325733)

[4.2.3.1 Implementació de l’arquitectura CLEAN 18](#_Toc199325734)

[4.2.3.1.1 Estructura de directoris 18](#_Toc199325735)

[4.2.3.1.2 Capa de Infrastructura 19](#_Toc199325736)

[4.2.3.1.2.1 DataSource 19](#_Toc199325737)

[4.2.3.1.2.2 Implementació del repositori 20](#_Toc199325738)

[4.2.3.1.3 Capa de domini 22](#_Toc199325739)

[4.2.3.1.3.1 Entitats 22](#_Toc199325740)

[4.2.3.1.3.2 Repositori 22](#_Toc199325741)

[4.2.3.1.4 Capa de Presentació 23](#_Toc199325742)

[4.2.3.1.4.1 Pantalles 23](#_Toc199325743)

[4.2.3.1.5 Capa de gestor d’estats 23](#_Toc199325744)

[4.2.3.1.5.1 Provider 23](#_Toc199325745)

[9.Dificultats i millores futures 29](#_Toc199325746)

[9.1 Dificultats 29](#_Toc199325747)

[9.2 Millores futures 29](#_Toc199325748)

[9.3 Que he aprés 29](#_Toc199325749)

[10.Annexos 30](#_Toc199325750)

[10.1 Github 30](#_Toc199325751)

[10.2 Bibliografia 30](#_Toc199325752)

# 1.Introducció del projecte

## 1.1Descripció del projecte

Aquest projecte té com a objectiu unir als fallers en grups anomenats families, els quals es poden unir altres fallers per a una gestió més comoda de la realitat a l’hora de poder unir-se a esdeveniments, comprar menjar/begudes en la parada de la falla...,etc. tot en una sola aplicació per a que puguen dur a terme els esdeveniments de la falla de forma més comoda.

## 1.2 Objectius del projecte:

Gestionar tots els esdeveniments, productes, tickets...,etc gràcies amb les polseres que funcionen amb la tecnologia NFC(Near-Field-Comunication) i alternativament amb ús de QR(Quick-Response-Code) per als dispositius que no conten amb la tecnologia NFC en una sola aplicació.

# 2.Tecnologies i ferramentes

## 2.1 Tecnologies usades per al projecte

Primerament vaig a fer un resum de les específicacions dels meus dispositius que han sigut involucrats en aquest projecte:

El projecte ha sigut desenvolupat en el meu ordinador personal, un portàtil amb les següents específicacions:

-Processador: Intel Core I5-11400H @ 2.70GHz, 2668MHz, 6 processadors principals, 12 proccessadors lògics

-Sistema operatiu: Microsoft Windows 11 Pro

-Memòria RAM:16GB

I ha sigut provat en un dispositiu mòbil Android de les següents específicacions:

-Versió de Android:14

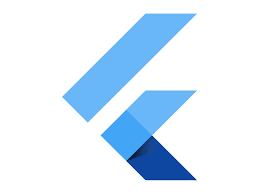
-Processador: MediaTek Dimensity 700 8 nuclis

-Espai d’enmagatzenament: 128GB

-Memòria RAM: 4GB

Ara esmentare les eines que he usat per a que aquest projecte haja sigut possible.

### 2.1.1 Flutter



Framework multiplataforma desenvolupat per Google, tenia com a proposit l’ús de software de la comanyia, però amb el temps s’ha transformat en una plataforma de codi obert per al desenvolupament multiplataforma.

Permet el desenvolupament en les paltaformes de:

-Android

-iOS

-Windows

-Linux

-macOS

-Web

En aquest projecte s’ha desenvolupat únicament per a plataformes Android, degut a que no dispose d’un sistema iOS per a fer proves i volia centrar-me en el ecosistema de Android que és el sistema operatiu mòbil més usat en Espanya un 78,8% en 2023 segons CNMC(Comisión Nacional de los Mercados y la Competencia), teniu un link en la bibliografia del article de CNMC.

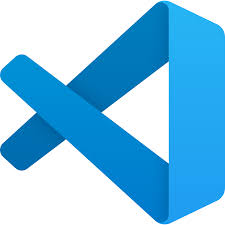
### 2.1.2 Dart



Dart es un llenguatge de programació de codi obert desenvolupat per Google, va ser revelat en la conferènci *goto* el 10 d’octubre de 2011. L’objectiu de Dart és oferir una alternativa més moderna a JavaScript per al desenvolupament web, es pot veure reflexat en les declaracions de Lars Bak, ingenier de software en Google que defineix a Dart com un *“llenguatge estructurat però flexible per a programació web”*.

He utilitzat Dart degut a que el framework de Flutter usa el llenguatge de programació de Dart m’he vist en l’obligació de programar en ell.

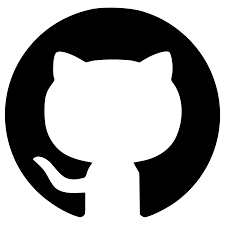
### 2.1.3 VSCode



Visual Studio Code llançat el 18 de novembre de 2015 és un editor de text pla desenvolupat per Microsoft de codi obert basat en Electron, un framework que s’utilitza per a implementar Chronium i NodeJS com a aplicacions d’escriptori, que s’executa en el motor de disseny Blink.

He usat Visual Studio Code com el meu IDE perquè és el IDE que més conec i més gaste actualment, però tens que tindre Android Studio instal·lat per les dependencies de Android que proveeix aquest últim junt amb les extensions de Flutter, i aquest últim té per defecte les llibreries de Git, així que m’ha ajudat molt per al control de versions Github.

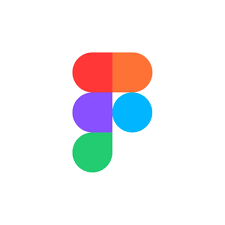
### 2.1.4 Github



Github, publicat en 2008 i desenvolupat amb Ruby on Rails és una forja(paltaforma de desenvolupament col·laboratiu) per a allotjar projectes utilitzant el sistema de control de versions de Git. S’usa principalment per a la creació de codi font de programes d’ordinador.

He fet servir GitHub com a repositori públic, perquè qualsevol persona puga veure el codi font del projecte, descarregar-lo o clonar-lo. Està tot organitzat per carpetes i podeu trobar tant el frontend com la resta del projecte. No he treballat amb branques ni commits complexos, però l’he utilitzat com a lloc de referència i accés al codi.

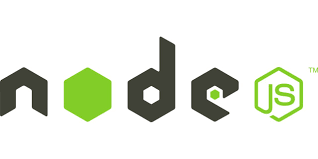
### 2.1.5 Figma



Figma, desenvolupat per Dylan Field i Evan Wallace començaren a treballar en Figma en 2012 i el van llançar el 27 de septembre de 2016 és un editor de gràfics vectorial i una ferramenta de generació de prototips, principalment basada en la web, amb caracteristiques off-line adicionals habilitades per aplicacions d’escriptori en macOS i Windows.

He utilitzat Figma únicament per a fer el prototip d’alta fidelitat del frontend. Em va servir per planificar el disseny visual abans de començar a programar l’aplicació, i assegurar-me que l’estructura i els estils quedaven clars.

### 2.1.6 Node.js



Node.js, publicat el 27 de maig de 2009 és un entorn de desenvolupament de codi obert per a la capa de servidor basat en el llenguatge de JavaScript, asincron amb E/S(periferic d’entrada/eixida) amb dades en una arquitectura basada en esdeveniments i basat en el motor V8 de Google.

He usat node.js com a backend i com a API, en Flutter faig peticions en el servidor de Node, que aquesta és connectara a la base de dades i torna la petició amb les dades que és mostraran en pantalla, però originalment anava a usar Odoo, després ho explicaré.

### 2.1.7 PostgreSQL



PostgreSQL també conegut com a Postgres, és un sistema de gestió de bases de dades relacional orientat a objectes i de codi obert publicat en 1996 baix la llicencia PostrgreSQL similar a la BSD o MIT

He usat PostgreSQL com a base de dades que està connectada al servidor amb NodeJS, aquesta guarda tota la informació sobre usuaris, families, productes...,etc i el servidor de Node.js arreplegara aquesta informació i la passara a l’aplicació en Flutter.

### 2.1.8 Odoo



Odoo, abans conegut com OpenERP i més enrrere conegut com TinyERP, va ser publicat el 4 de juliol de 2004 es un software de ERP integrat. Compta amb una versió “Community” de codi obert baix la llicencia LGPLv3 i una versió empresarial baix llicencia comercial que complementa la versió comunitaria amb característiques i serveis comercials i desenvolupada per la companyia belga Odoo S.A.

Es un software empresarial tot en un que inclou CRM, lloc web i comerç electrònic, facturació, contabilitat, fabricació gestió de magatzems i projectes i inventari...,etc.

Originalment el backend del meu projecte abans de que fera el servidor de Node, és paregut el que fa el servidor de Node i la base de dades PostgreSQL, però amb una sola plataforma, va ser substituïda pel meu tutor per a implementar el servidor de Node, el qual està en eta presentació, però per falta de temps no va ser possible implementar les funcions de Node a Odoo.

## 2.2 Tecnologia NFC

NFC o per les seues sigles (Near Field Communication) funciona a través d’ones electromagnètiques que conseguiran comunicar un dispositiu a un altre fins a una distancia de 10 a 15 cm a una velocitat de 424Kb/s, ara esmentare alguns exemples.

Exemples de NFC:

-Tarjetes tant de crèdit com d’accés:

-Polseres amb la tecnología abans esmentada (aquesta serà en la que treballarem):



-Alguns rellotges intel·ligents:



-Etiquetes:



-Algunes figuretes col·leccionables (per exemple els Amiibo de Nintendo per als sistemes de Nintendo 3DS, Nintendo Switch i possiblement també en Nintendo Switch 2):



Els dispositius que conten amb la tecnologia NFC tenen 2 formes per a comunicar-se:

-Pasiva: Sols un dels dispositus està actiu i l’altre intercambiara les dades.

-Activa: Els 2 dispositius generen un camp electromagnètic i intercambien dades.

## 2.3 Codi QR

També conegut com (Quick-Respnse-Code), va ser inventat en 1994 pel ingenier japones **Masahiro Hara**, va ser desenvolupat per la empresa **Denso Corporation** la qual és una subsidiaria de Toyota, el codi QR és una millora al codi de barres tradicional, degut a que és un mètode per a enmagatzemar informació mitjançant constractes de color de forma bidimensional(2D), tenint com a objectiu una letura ràpida del contingut del codi, per això el seu nom de **Q**uick-**R**esponse-Code, és pot usar de diverses fromes, com per exemple:

-Productes en general:



-Cartes de restaurant:



-Correus electrònics:



-Anuncis:



-Accés ràpid a documents gubernamentals:



-Alguns videojocs (per a conseguir algunes reconpenses o et du a la pàgina oficial del videojoc):



# 3. Diseny de la solució

# 4. Desenvolupament de la solucio

## 4.1 Consideracions prèvies

## 4.2 Configuració i desenvolupament del sistema

### 4.2.1 Configuració de Flutter i el seu entorn de desenvolupament

#### 4.2.1.1 Instal·lació de Flutter i les seues dependències

#### 4.2.1.1 Instal·lar l’entorn de desenvolupament

### 4.2.2 Configuració del backend(Servidor de node.js)

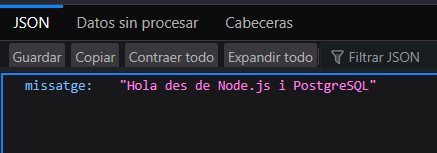
La única cosa que tens que fer per a que el servidor de node.js i la base de dades funcionen per a que l’aplicació reba totes les dades es obrir la carpeta de fakeApiOdoo que està en el projecte de GitHub i en un terminal poses “node app.js” i l’aplicació estara ja funcionant perque en el terminal posara el següent:

**Servidor escoltant en** [**http://0.0.0.0:3000**](http://0.0.0.0:3000)

**IMPORTANT:**

Com el servidor de node tens que arrancar-lo tu mateix això significa que depen d’on estigues i del dispositiu el servidor tindra una IP o una altra, com en el meu cas el servidor està en una màquina virtual en NAT per a connectar-te des del mateix dispositiu sera sempre la mateixa 127.0.0.1, però per a l’app de flutter és una altra història, tindràs que posar la ip del teu dispositiu real 192.168.x.x per a que l’app de flutter es connecte al sevidor de Node.js.

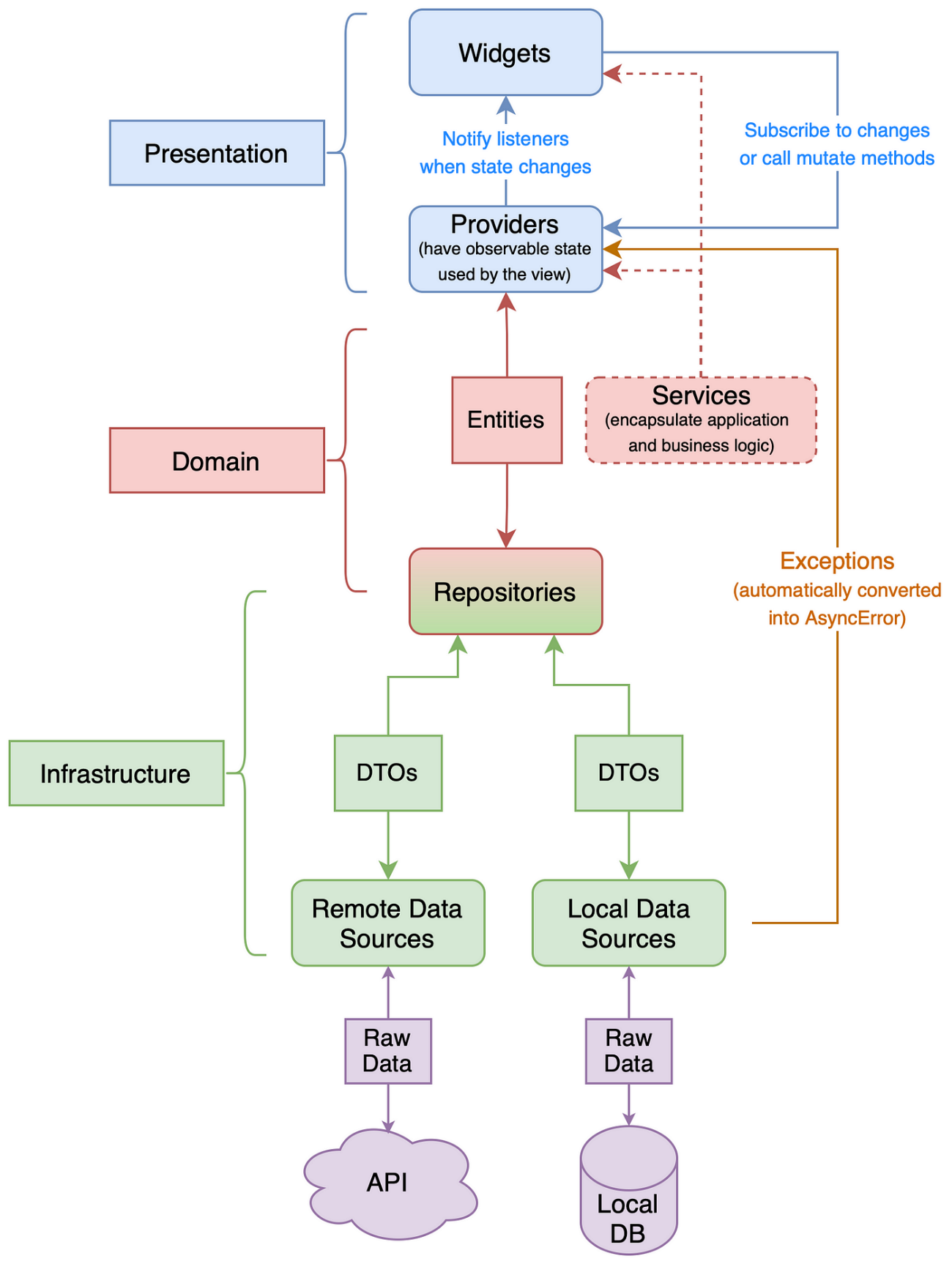
Ara ignorant el problema si aneu al vostre navegador i poseu localhost:3000 o ipVostra:3000 podeu vore el següent:



Això significa que el servidor ja està funcionant i esperant a rebre peticions.

### 4.2.3 Arquitectura CLEAN

L’arquitectura CLEAN proposada per **Robert C. Martin (Uncle Bob)** és unacrònim de que prové de “**Clean Architecture**” es una forma per a organitzar el codi de forma que separe clarament les responsabilitats, millore la escabalitat, mantenibilitat i “testabilitat” en una aplicació, encara que l’arquitetura CLEAN no és exclusiva de Flutter es pot aplicar perfectament en projectes de Flutter/Dart per a estructurar de forma correcta una app, l’arquitectura CLEAN es pot resumir en la següent imatge:



Explicació:

Capa Presentation:

-UI: Interficie de l’aplicació (el que és mostrara en la app).

Capa Domain:

-Repository(implementació): implementa les funcions del DataSource, hereta del Repository(sense implementació) i l’usa la intreficie(UI).

-Entities: Els objectes de l’aplicació que rebran els canvis.

Capa Infrastructure:

-Repository: Capa intermitja que connecta i implementa el DataSource i heretara a la implementació del Repositori en la capa de “Domain”

-DataSource: Capa que fa peticions a APIs o a altres llocs per a arreplegar la informació i després mostrar-la en la interficie(UI).

Gestor d’estats:

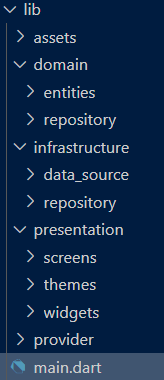
-Provider: Gestor d’estat per a saber l’estat de les peticions.

Ara que ja em dit la teoria anem a la pràctica

#### 4.2.3.1 Implementació de l’arquitectura CLEAN

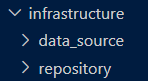
##### 4.2.3.1.1 Estructura de directoris

En el vostre projecte de Flutter per a seguir amb l’arquitectura CLEAN heu de seguir la següent cadena de directoris.



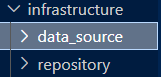
##### 4.2.3.1.2 Capa de Infrastructura

Primerament creem la següent cadena de carpetes si no les teniu creades



###### 4.2.3.1.2.1 DataSource

Anem a la carpeta de “data\_source” i creem un nou fitxer



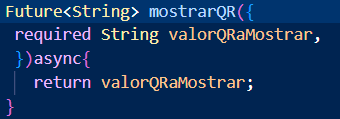


Ara en el fitxer inicialitzarem la classe si volem afegir alguna variable si volem, en este cas com no fa falta ho deixem buit.



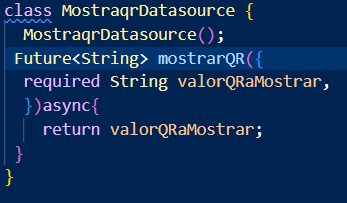
Inicialització de la classe

Quan ja hem inicialitzat el DataSource farem les funcions que fagen falta, com en este cas sols és una i és molt sencilla sols torna el valor de forma asincrona per a poder mostrar un QR que està en la pantalla, sols ens interesa el valor que tindra el QR, així que la implementem:



Funcions del DataSource

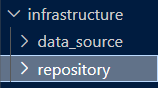
Quan implementem la funció deuria de quedar de la següent forma:



Classe completa del DataSource

###### 4.2.3.1.2.2 Implementació del repositori

Anem a la carpeta del repositori i creem un nou fitxer



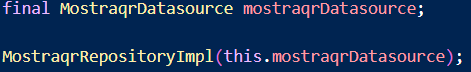


Quan creem la classe és important que implemente al repositori(sense implementació) per a que herete totes les funcions del repositori(sense implementació) i les pugues implementar:



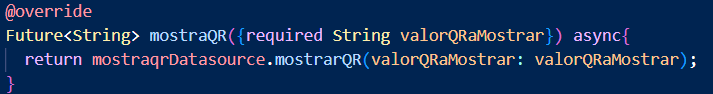
Implementació del repositori(sense implementació)

Per a fer les funcions del repositori és important que s’inicialitze el DataSource, perque gastarem aquest últim per a fer les funcions i estalviar-mo’n codi:



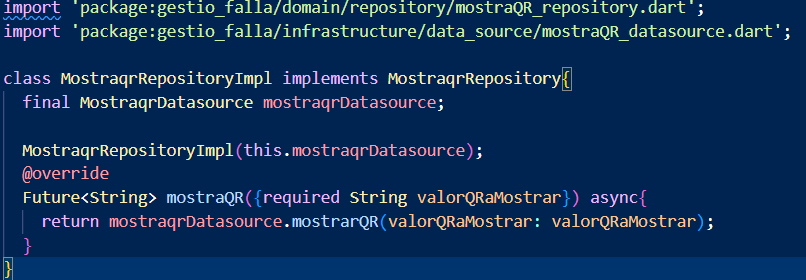
Inicialització del DataSource

Com he dit abans les funcions seran exactament les mateixes que el repositori(sense implementació) així que és important que dalt de la funció poseu “@override” perque sobrescriura la funció del repostitori(sense implementació) i cridarem a la instancia de DataSource que haviem creat abans:



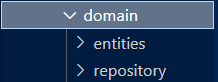
Funcions del repositori(implementacio) amb @override del repositori(sense implementació)

Al final deuria de quedar de la següent forma:



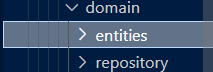
##### 4.2.3.1.3 Capa de domini

Primerament creem la següent cadena de carpetes si no les teniu creades



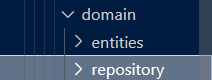
###### 4.2.3.1.3.1 Entitats

En este cas la entitat no forma part directament en esta funció, pero com en la pantalla que implementem la arqur¡itectura parlaré un poc d’aquesta, així que anem a la carpeta de entitats:



###### 4.2.3.1.3.2 Repositori

Anem a la carpeta de repositoris i creem un nou fitxer:





Ara per a crear la classe la tenim que fer abstracta perque serveixen per definir una **estructura comuna o un contracte** que altres classes han de seguir, però sense implementar completament els detalls, aleshores implentem la classe abstracta:

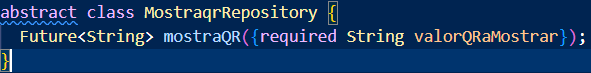


Implementació de la classe abstracta del repositori

I per últim fem les funcions que tenen que ser iguals al DataSource per a que el repositori amb implementació puga usar el DataSource:

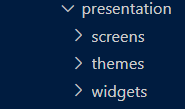


Al final deuria de quedar de la següent forma:



##### 4.2.3.1.4 Capa de Presentació

Creeu la següent cadena de carpetes si no les teniu creades:



Sols tinc les pantalles perque no me s’ha ocurrit algun widget personalitzat o un tema en específic.

###### 4.2.3.1.4.1 Pantalles

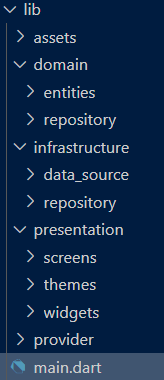
Ací és on ocorrira la magia de que apararega en la pantalla, no n’hi ha molt més misteri, sols que ací podràs aplicar tant els DataSources, repositoris i/o gestors d’estats(BLOC,Provider, Riverpod...,etc).

##### 4.2.3.1.5 Capa de gestor d’estats

###### 4.2.3.1.5.1 Provider

Provider és un gestro d’estat més avanzat i més sencill que BLOC, un altre gestor de l’estat, ara dire com podeu implementar provider en els vostres projectes:

1.Tindre tant la capa de serveis tant la de repositori ja feta



Fixeu-se en les carpetes de:

domain -> repository

infrastructure -> repository i datasource

2.Crear la capa de provider en la carpeta anomenada “lib”



3.Crear el fitxer “nomdelafuncioProvider.dart” i al crear la classe que implemente “ChangeNotifier”

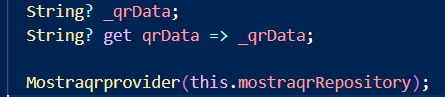




4.Inicialitzar el provider amb el repository



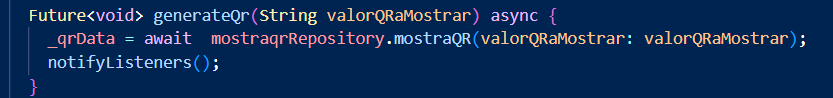
5.Completar el provider amb les funcions del repositori



Funcions bàsiques del provider que és voran en la pantalla

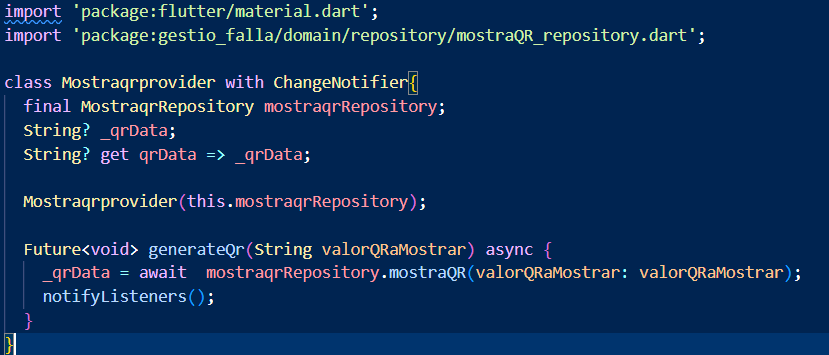
+

Inicialització del repository al provider



Funció que implementa el repositori i mostrara guarda en una variable de provider per a que es puga vore per pantalla

El fitxer provider deuria de quedar de la següent forma:

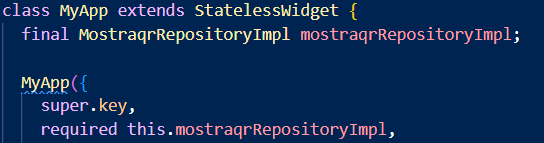


6.Inicialitzar al fitxer “main.dart”

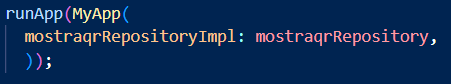




Inicialització del datasource i el repositori(la implementació)

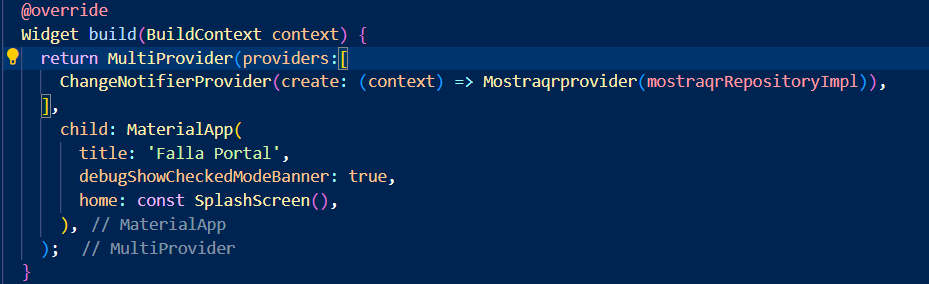


Implememtem el repositori i fem que el fitxer main el requerisca per a funcionar



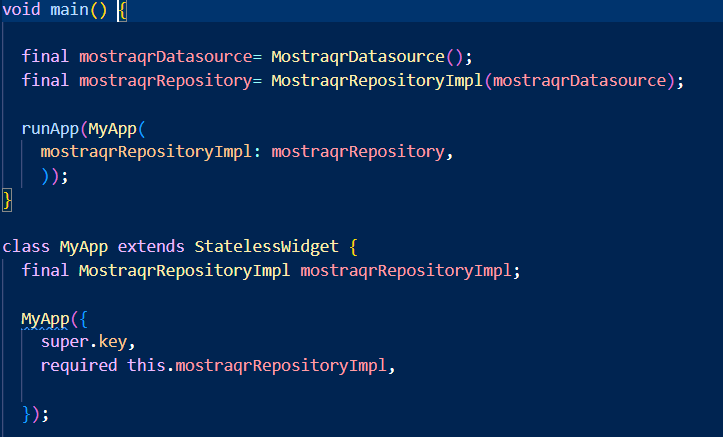
Per últim posem el repositori que haviem inicialitzat al principi

7. Fer que retorne “Multiprovider” per a tindre una llista de providers



Fem que en lloc de retornar MaterialApp retorne el Multiprovider i el MaterialApp sera un fill de Multiprovider per a implementar la resta del main amb normalitat

Al final el fitxer main deuria de quedar de la següent forma:



8.En la pantalla que vullgues inicialitza la funció amb “Provider.of<nomdelafuncioProvider>”



Inicialitzem el provider i li posem el valor de la funcio que demanava (en este cas és un objecte de tipus String, però no n’hi haura problema a l’hora de mostrar-lo)

Ara en la funció inicialitzada la implementem en la pantalla:



En el widget Consumer posem el nom de la classe de Provider i en el seu constructor (builder) posem el provider i ja podem usar la funció de provider dins de la pantalla

I amb això ja pots usar la teua provider en qualsevol lloc de l’aplicació.



En aquest exemple com es un codi QR no es veu molt bé el resultat, però com el codi és generat per un String es pot vore de forma clara com s’ha dibuixat en la pantalla

# 9.Dificultats i millores futures

## 9.1 Dificultats

Adaptar la pantalla del mòbil per a que no es sobreisca en horizontal era un problema que tenia en les primeres versions de l’aplicació, vaig estar batallant amb eixe tema unes setmanes fins que vaig poder fer que no s’isca i poder implemntar un SingleChildScrollView per a que si se n’ix es puga tirar cap al costat per on se n’ix conseguint un resultat que estic molt content.

Fer que el servidor de node estiga en el nuvol el qual vaig voler implementar per a no tindre que anar cambiant en la app de Flutter la IP cada 2x3, però al final no va ser possible per falta de temps i em vaig estar atascat durant setmanes mirant com solucionar-ho sense èxit.

## 9.2 Millores futures

Passar del servidor de Node a Odoo mitjançant jsonrpc, com he dit abans vaig tindre que deixar Odoo de costat per a poder fer que funcione en Node, però per falta de temps no vaig poder cambiar-ho altra vegada a Odoo, tenia unes funcions molt bàsiques, però res comparades al servidor de Node.js

Compatibilitat en iOS, encara que jo no tinga un deispositiu de Apple o un dispositiu amb el sistema operatiu de iOS m’agradaria poder fer una versió completament funcional en aquest sistema operatiu per a no deixar als usuaris de Apple sense l’aplicació.

## 9.3 Que he aprés

Amb la realització d’aquest projecte he descobert que és la tecnologia NFC i per a que serveix, junt amb els codi QR, que ja sabia que existien, però no sabia com funcionaven.

# 10.Annexos

## 10.1 Github

El repositori del projecte en Github: <https://github.com/JoelFusterBosch/Projecte_Final_DAM_Gestio-_d_una_falla>

## 10.2 Bibliografia

2.Tecnologies i ferramentes

2.1.1 Flutter

Sistemes mòbils més usats CNMC(Comisión Nacional de los Mercados y la Competencia):

[https://www.cnmc.es/prensa/panel-hogares-usos-internet-20231103#:~:text=Un%2078%2C8%20%%20de%20los%20espa%C3%B1oles%20con,Internet%20y%20los%20servicios%20OTT%20en%20Espa%C3%B1a.](https://www.cnmc.es/prensa/panel-hogares-usos-internet-20231103%23:~:text=Un%2078%2C8%20%25%20de%20los%20espa%C3%B1oles%20con,Internet%20y%20los%20servicios%20OTT%20en%20Espa%C3%B1a.)

Pàgina oficial de Flutter:

<https://flutter.dev/>

2.1.2 Dart

Pàgina oficial de Dart:

<https://dart.dev/>

2.1.3 Visual Studio Code

Pàgina oficial de Visual Studio Code:

<https://code.visualstudio.com/>

2.1.4 Github

Pàgina oficial de github:

<https://github.com/>

2.1.5 Figma

Pàgina oficial de Figma:

<https://www.figma.com>

2.1.6 Node.js

Pàgina oficial de Node.js

<https://nodejs.org/es>

2.1.7 PostgreSQL

Pàgina oficial de PostgreSQL

<https://www.postgresql.org/>

2.1.8 Odoo

Pàgina oficial de Odoo

<https://www.odoo.com/es_ES>

4.Desenvolupament de la solució

4.2Configuració i desenvolupment del sistema

4.2.3 Arquiteectura CLEAN

Més informació sobre l’arquitectuura CLEAN i altres arquitectures

<https://blog.burkharts.net/practical-flutter-architecture>

10.Annexos

El meu repositori de github del projecte:

<https://github.com/JoelFusterBosch/Projecte_Final_DAM_Gestio-_d_una_falla>