Creación de un dashboard para usuarios del ticket digital de Mercadona con visualización gráfica de datos: evolución de precios por producto, gastos por categoría de alimentación y ventanas temporales de gastos

Santiago Sánchez Sans

IES Abastos

6 junio 2025

Contenido

1. Introducción
2. Diseño
   - Requisitos
   - Diagramas de sistemas
3. Desarrollo
   - Entornos de desarrollo
   - Despliegue
   - Spring Boot: gestión usuarios
   - FastAPI: parseo de tickets
   - FastAPI: exposición tickets (+ dashboard)
   - Front-end: Vanilla JS
4. Conclusiones

Contenidos

1 Introducción

2 Diseño
- Requisitos
- Diagramas de sistemas

3 Desarrollo
- Entornos de desarrollo
- Despliegue
- Spring Boot: gestión usuarios
- FastAPI: parseo de tickets
- FastAPI: exposición tickets (+ dashboard)
- Front-end: Vanilla JS

4 Conclusiones

1. Introducción

- **Identificación de necesidades**:
  - Usuario del ticket digital → no tiene informes de sus datos.
- **Objetivos**:
  - Proporcionar al usuario del ticket digital una herramienta que muestre en gráficos visuales:
    - **Evolución de precios** (inflación) a lo largo del tiempo en los productos habitualmente obtenidos en el mismo establecimiento[1].
    - **Evolución del gasto** total del usuario a lo largo del tiempo por períodos temporales.

[1] La evolución de precios se mostrará solamente para un mismo centro de Mercadona, dado que distintos centros pueden cambiar los nombres de los productos (por ejemplo, en Cataluña...)

Contenidos

1 Introducción

2 Diseño
- Requisitos
- Diagramas de sistemas

3 Desarrollo
- Entornos de desarrollo
- Despliegue
- Spring Boot: gestión usuarios
- FastAPI: parseo de tickets
- FastAPI: exposición tickets (+ dashboard)
- Front-end: Vanilla JS

4 Conclusiones

Requisitos de los usuarios

Que los usuarios tengan una cuenta de gmail con tickets digitales de Mercadona dentro e, idealmente, tenga decenas de tickets digitales: idealmente con compras estables y productos de adquisición recurrentes. El requisito indispensable es tener un mínimo de dos tickets digitales distintos.

Requisitos funcionales

REQUISITO A: Mostrar **evolución de los precios** de los productos unitarios adquiridos con más frecuencia (visualizable en un gráfico donde en X tendremos el tiempo y en Y el precio en euros).
REQUISITO B: Mostrar **gasto total en distintas ventanas temporales** del usuario: períodos de 1, 3, 6 meses y un año; independientemente del centro de Mercadona en el que se compre (todos juntos).
REQUISITO C: Al lado del gasto total anterior se incluirá un **diagrama de sectores** desglosando porcentaje de dinero gastado en 13 categorias (click para ver categorías)

Requisitos funcionales (cont.)

REQUISITO D[2]**:** Los PDFs descargados del correo del usuario se almacenarán en una carpeta local del ordenador del usuario.
REQUISITO E[3]**:** El sistema front-end y back-end de registro permitirá redirigir a los usuarios rápidamente a un registro de forma inteligente. Nos inspiraremos en el sistema de registro e iniciar sesión de NetFlix. Ver diagrama enrutamiento.

[2] Requisito añadido después de la presentación inicial del proyecto.
[3] Requisito añadido después de la presentación del proyecto.

Requisitos funcionales (RESUMEN)

De los requisitos al diseño (anticipo de lo que será el dashboard):

- evolución de precios por producto → **“inflalyzer”**
- gastos por categoría de alimentación → **“categoryzer”**
- ventanas temporales de gastos → **“intervalizer”**

Requisitos

Figure: Autenticación y autorización por token. Se usa tanto en front-end como en back-end.

Programáticamente, debemos conseguir que estas "vistas públicas" **no sean visualizables** jamás si, en el *local storage*, **existe** un token de acceso que **no haya expirado**[56]. Estas páginas vetadas a los usuarios "logueados" son las siguientes:

- *A)* `pas1_landingSignUp.html` (`index.html`)
- *B)* `pas2A_infoBenvinguda.html`
- *C)* `pas2B_introduirContrasenya.html`
- *D)* `pas2C_login.html`
- *E)* `pas3_crearContrasenya.html`

página accedida	Permisos token	Redirigimos automáticamente a
A), B, C), D) o *E)*	0	pas4_concedirAccesGmail.html
	1||2	dashboard.html

```
<script src="/js/jwt/decodificaJWT.js"></script>
<script src="/js/jwt/restringeixVistesPubliques_USUARI_LOGUEJAT.js"></script>
```

Figure: Redirecciones automáticas a páginas privadas cuando usuarios logueados (que tienen un token de acceso no caducado) acceden a vistas públicas.

p. privada accedida (vetada)	Permisos	Redirección automática a
dashboard.html	0	pas4_concedirAccesGmail.html
pas4_concedirAccesGmail.html	1||2	dashboard.html

```
<script src="/js/jwt/decodificaJWT.js"></script>
<script src="/js/jwt/restringeixVistesPrivades_USUARI_NO_LOGUEJAT.js">
</script>
```

Figure: Redirecciones automáticas a otras páginas privadas cuando usuarios logueados que ya tienen un token acceden a vistas privadas que no les corresponden.

Diagrama general

registro

"Vanilla" JavaScript
Google API Client
d
e
a
c
l
f
Spring Boot
FastAPI
j
h
b
i
k
g
MySql
(usuaris)
MongoDB
(tickets)
API externa en la nube de google
Front-end
Back-end
Base de datos
Petición HTTP (Cliente)
Respuesta HTTP (Expedida por el servidor)
Guardado a base de datos
Lectura de base de datos

inicio de sesión

Contenidos

1 Introducción

2 Diseño
- Requisitos
- Diagramas de sistemas

3 Desarrollo
- Entornos de desarrollo
- Despliegue
- Spring Boot: gestión usuarios
- FastAPI: parseo de tickets
- FastAPI: exposición tickets (+ dashboard)
- Front-end: Vanilla JS

4 Conclusiones

Entornos de desarrollo

Editor / Herramienta	**Puerto[4])**
IntelliJ IDEA (Java, SpringBoot)	8080
VSCode (HTML, CSS, JS con Live Server)	5500
VSCode (Python, con FastAPI[5])	8000
MySQL Workbench	3306
MongoDB Compass	27017

Table: Entornos de desarrollo y puertos utilizados para despliegue en local

[4] El host es localhost
[5] No depende del editor de código

Despliegue

Se ha automatizado la creación de imágenes e instanciado de contenedores para cada microservicio. PUERTOS: ¡idem!

Despliegue (cont.)

Imagen original	puerto
openjdk:17-alpine	8080
nginx:alpine	5500[6]
Python:alpine (DF)	8000

Table: Imágenes docker base y puertos donde instanciamos su contenedor

base de datos	puerto
MySQL	3306
MongoDB	27017

Table: Bases de datos: no contenerizadas

[6] localhost no sirve; usar 127.0.0.1 en navegador para ver index.html

spring
boot

Java

Spring Boot: gestión usuarios

Spring Boot: gestión usuarios

Spring Boot: gestión usuarios

Spring Boot: gestión usuarios

Spring Boot: gestión usuarios

Spring Boot: gestión usuarios

POST http://localhost:8080/api/login

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"correuElectronic" : "acces@gmail.com", "contrasenya" : "12345678Mm_" }
```

Body Cookies Headers (14) Test Results

{} JSON Preview Visualize

```
1  {
2      "usuari": {
3          "alies": "blackcub3s",
4          "permisos": 1,
5          "idUsuari": 2
6      },
7      "existeixUsuari": true,
8      "AccessToken": "eyJhbGciOiJIUzI1NiJ9.eyJwZXJtaXNvcyI6MSwiaWRVc3Vhcm
9      "teAccesArecursos": true,
10     "contrasenyaCorrecta": true
11 }
```

utils
validacio
Alies
Contrasenya
CorreuElectronic
PlaSuscripcio
ValidacionsUsuari
Usuaris
controlador
UsuariControlador
dto
ActualitzaContrasenyaDTO
CorreuDTO
LoginDTO
RegistreDTO
UsuariDTO

Spring Boot: gestión usuarios

CorreuElectronic.java

```java
//-------VALIDACIONS DTO CORREU ELECTRONIC-------
@Size(min=3, max = TAMANY_MAXIM_EMAIL, message = MISSATGE_ERROR_TAMANY_EMAIL)
@NotBlank(message = MISSATGE_NOT_BLANK_GENERIC)
@Pattern(regexp = REGEX_EMAIL, message = MISSATGE_ERROR_CORREU)
//-------FI VALIDACIONS DTO CORREU ELECTRONIC-------
public @interface CorreuElectronic {
    String message() default "Correo electrónico no válido";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

Contrasenya.java

```java
[...]
//-------VALIDACIONS DTO Contrasenya-------
@Size(min = TAMANY_MINIM_CONTRASENYA, max = TAMANY_MAXIM_CONTRASENYA, message = MISSATGE_ERROR_TAMANY)
@NotBlank(message = MISSATGE_NOT_BLANK_GENERIC)
@Pattern(regexp = REGEX_CONTRASENYA, message = MISSATGE_ERROR_CONTRASENYA)
//-------FI VALIDACIONS DTO Contrasenya-------
public @interface Contrasenya {
    String message() default "Contrasenya no vàlida";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

ValidacionsUsuari.java

```java
//VALIDO ELS CAMPS DE LA TAULA USUARI
public class ValidacionsUsuari {


    public static final String REGEX_CONTRASENYA =  "^(?=[a-zA-Z0-9àáéè(òóú_.ÀÉÉÍÓÚÜÑÇ]+$)(?=.*[a-zç])(?=.*[A-ZÇ])(?=.*\\d)(?!.*\\s).{8,25}$";
    public static final String MISSATGE_ERROR_CONTRASENYA = "¡La contraseña debe contener al menos una minúscula, una mayúscula, un número y no puede
contener espacios! Se admiten puntos y barras bajas pero no otros caracteres";
    public static final int TAMANY_MINIM_CONTRASENYA = 8;  //per a @Size
    public static final int TAMANY_MAXIM_CONTRASENYA = 25; //per a @Size
    public static final String MISSATGE_ERROR_TAMANY = "¡La contraseña debe tener entre 8 y 25 caracteres!"; //per a @Size

        [...]

}
```

LoginDTO.java

```java
@Getter @Setter
public class LoginDTO {

    @CorreuElectronic
    private String correuElectronic;

    @Contrasenya
    private String contrasenya;
}
```

UsuariControlador.java

```java
@RestController
@RequestMapping("/api")
public class UsuariControlador {

    //   [...]

    @PostMapping("/login")
    public ResponseEntity<HashMap<String, Object>> login(@RequestBody @Valid LoginDTO dto) {
        HashMap<String, Object> mapJSONlike = serveiUPP.generaBodyLogin (
                                                dto.getCorreuElectronic(),
                                                dto.getContrasenya()

        );
        return new ResponseEntity<>(mapJSONlike, HttpStatus.OK);
    }
```

Figure: Función de servicio en Spring Boot para expedir un acceso con el id de usuario proporcionado por FastAPI (solicitud cliente cuando FastAPI ya tiene tickets persistidos). **Nótese que en 3 líneas de código tenemos la persistencia en mySQL hecha. ¿Función save?**.

```
@Repository
public interface UsuariRepositori extends
JpaRepository<Usuari, Integer> {
```

Figure: Con JpaRepository tenemos funciones para persistir en base de datos sin tener que definir consultas (la función save de la diapositiva anterior). Al hacer nuestra interface para operaciones de persistencia solamente debememos extender de JpaRepository e indicar la clase con la que hacemos el ORM y su clave primaria

usuari
id_usuari INT
correu_electronic VARCHAR(70)
hash_contrasenya VARCHAR(25...
alies VARCHAR(30)
permisos TINYINT
data_registre DATETIME
Indexes
usuari_ampliat
id_usuari INT
nom VARCHAR(25)
primer_cognom VARCHAR(2...
segon_cognom VARCHAR(25)
Indexes
PRIMARY

FastAPI: parseo de tickets

FastAPI: parseo de tickets

FastAPI: parseo de tickets

FastAPI: parseo de tickets

Figure: Añadir usuario de prueba

Figure: Configurar un origen autorizado

FastAPI: parseo de tickets

LEYENDA

Delimitadores

ROIs

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95
		TOTAL (€)	**33,36**
		TARJETA BANCARIA	**33,36**

IVA	BASE IMPONIBLE (€)	CUOTA (€)
4%	15,32	0,61
10%	15,57	1,56
21%	0,25	0,05
TOTAL	**31,14**	**2,22**

TARJ BANCARIA: **** **** **** 6633
N.C: 098100738 AUT: 241212
AID: A0000000031010 ARC: 00
Verificado por dispositivo

Importe: 33,36 Â€ Visa Debit

MERCADONA 800 500 220

SE ADMITEN DEVOLUCIONES CON TICKET

FastAPI: parseo de tickets

LEYENDA

Delimitador

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95
		TOTAL (€)	**33,36**
		TARJETA BANCARIA	**33,36**

IVA	BASE IMPONIBLE (€)	CUOTA (€)
4%	15,32	0,61
10%	15,57	1,56
21%	0,25	0,05
TOTAL	**31,14**	**2,22**

TARJ BANCARIA: **** **** **** 6633
N.C: 098100738 AUT: 241212
AID: A0000000031010 ARC: 00
Verificado por dispositivo

Importe: 33,36 Â€ Visa Debit

MERCADONA 800 500 220

SE ADMITEN DEVOLUCIONES CON TICKET

```python
def pdf_to_text(file_path):
    try:
        text = ""
        with open(file_path, "rb") as file:
            reader = PyPDF2.PdfReader(file)
            for page_num in range(len(reader.pages)):
                page = reader.pages[page_num]
                text += page.extract_text() + "\n"
        return text
    except PyPDF2.errors.PdfReadError:
        return "errorPdfAtext"
```

```python
textPDF = pdf_to_text(doc)
```

```python
separacioTicketPerTOTAL = textPDF.split("TOTAL (€)")
```

FastAPI: parseo de tickets

LEYENDA

Delimitador

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

TOTAL (€) 33,36
TARJETA BANCARIA 33,36

IVA	BASE IMPONIBLE (€)	CUOTA (€)
4%	15,32	0,61
10%	15,57	1,56
21%	0,25	0,05
TOTAL	**31,14**	**2,22**

TARJ BANCARIA: **** **** **** 6633
N.C: 098100738 AUT: 241212
AID: A0000000031010 ARC: 00
Verificado por dispositivo

Importe: 33,36 Â€ Visa Debit

MERCADONA 800 500 220

SE ADMITEN DEVOLUCIONES CON TICKET

```
separacioTicketPerTOTAL = textPDF.split("TOTAL (€)")
```

```
separacioTicketPerTOTAL[0]
```

```
separacioTicketPerTOTAL[1]
```

FastAPI: parseo de tickets

LEYENDA

Delimitador

FastAPI: parseo de tickets

LEYENDA

- **Delimitador**
- **ROI**

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95
		TOTAL (€)	**33,36**
		TARJETA BANCARIA	**33,36**

IVA	BASE IMPONIBLE (€)	CUOTA (€)
4%	15,32	0,61
10%	15,57	1,56
21%	0,25	0,05
TOTAL	**31,14**	**2,22**

TARJ BANCARIA: **** **** **** 6633
N.C: 098100738 AUT: 241212
AID: A0000000031010 ARC: 00
Verificado por dispositivo

Importe: 33,36 Â€ Visa Debit

MERCADONA 800 500 220

SE ADMITEN DEVOLUCIONES CON TICKET

```
separacioTicketPerTOTAL = textPDF.split("TOTAL (€)")
```

```
separacioTicketPerTOTAL[0]
```

```
separacioTicketPerTOTAL[1]
```

```
preuTotalTicket = float(separacioTicketPerTOTAL[1]
                        .index("\n")].strip().replace(",",".")
```

FastAPI: parseo de tickets

LEYENDA

Delimitador

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

TOTAL (€) 33,36
TARJETA BANCARIA 33,36

IVA	BASE IMPONIBLE (€)	CUOTA (€)
4%	15,32	0,61
10%	15,57	1,56
21%	0,25	0,05
TOTAL	**31,14**	**2,22**

TARJ BANCARIA: **** **** **** 6633
N.C: 098100738 AUT: 241212
AID: A0000000031010 ARC: 00
Verificado por dispositivo

Importe: 33,36 Â€ Visa Debit

MERCADONA 800 500 220

SE ADMITEN DEVOLUCIONES CON TICKET

```
separacioTicketPerTOTAL = textPDF.split("TOTAL (€)")
```

```
separacioTicketPerTOTAL[0]
```

```
separacioTicketPerTOTAL[1]
```

String de una sola línea con la parte verde del ticket digital dentro

FastAPI: parseo de tickets

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

```
ll_liniesTicket = separacioTicketPerTOTAL[0].split("\n")
for i in range(len(ll_liniesTicket)):
    print(ll_liniesTicket[i])
```

```
MERCADONA, S.A.   A-46103834
C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14  OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893
Descripción P. Unit Importe
2BOLSA PLASTICO 0,15 0,30
2LECHE SEMI CALCIO 1,03 2,06
1SPAGHETTI 1,20
1QUESO CHEDDAR 3,30
1CINTAS DE BACON 2,30
1JAMON PAVO FINAS LON 1,85
1LOMO DE PAVO 3,20
1FILETE PECHUGA CERT 4,28
1NATA 15% SIN LACTOSA 2,40
112 HUEVOS SUPER GRAN 3,90
1ROJA DULCE BOLSA 4,16
1FILETE PECHUGA CERT 4,28
1NATA 15% SIN LACTOSA 2,40
112 HUEVOS SUPER GRAN 3,90
1ROJA DULCE BOLSA 4,16
1AQUAREL 1,5L 0,62
1BRONCHALES 6L 1,28
1KIWI VERDE
0,494 kg 3,16 €/kg 1,56
1CALABACIN BLANCO
0,454 kg 2,10 €/kg 0,95
```

```
separacioTicketPerTOTAL[0]
```

FastAPI: parseo de tickets

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELEFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

```python
ll_liniesTicket = separacioTicketPerTOTAL[0].split("\n")
for i in range(len(ll_liniesTicket)):
    print(ll_liniesTicket[i])
```

```
MERCADONA, S.A.   A-46103834
C/ SENYERA 24
46006 VALENCIA
```

[...]

```python
carrer, CP_ciutat = ll_liniesTicket[1], ll_liniesTicket[2]
direccioSuper = carrer + " " + CP_ciutat
```

```
separacioTicketPerTOTAL[0]
```

FastAPI: parseo de tickets

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

```
ll_liniesTicket = separacioTicketPerTOTAL[0].split("\n")
for i in range(len(ll_liniesTicket)):
    print(ll_liniesTicket[i])
```

```
MERCADONA, S.A.   A-46103834
C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14  OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893
Descripción P. Unit Importe
```

```
dataOperacio = ll_liniesTicket[4]
liniaFactura = ll_liniesTicket[5]
```

```
separacioTicketPerTOTAL[0]
```

FastAPI: parseo de tickets

FastAPI: parseo de tickets

MERCADONA, S.A. A-46103834

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
30/04/2025 21:14 OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893

	Descripción	P. Unit	Importe
2	BOLSA PLASTICO	0,15	0,30
2	LECHE SEMI CALCIO	1,03	2,06
1	SPAGHETTI		1,20
1	QUESO CHEDDAR		3,30
1	CINTAS DE BACON		2,30
1	JAMON PAVO FINAS LON		1,85
1	LOMO DE PAVO		3,20
1	FILETE PECHUGA CERT		4,28
1	NATA 15% SIN LACTOSA		2,40
1	12 HUEVOS SUPER GRAN		3,90
1	ROJA DULCE BOLSA		4,16
1	AQUAREL 1,5L		0,62
1	BRONCHALES 6L		1,28
1	KIWI VERDE		
	0,494 kg	3,16 €/kg	1,56
1	CALABACIN BLANCO		
	0,454 kg	2,10 €/kg	0,95

ROIs

Delimitador

```
30/04/2025 21:14  OP: 4083409
FACTURA SIMPLIFICADA: 2423-026-567893
```

```
dataOperacio = ll_liniesTicket[4]
liniaFactura = ll_liniesTicket[5]
```

```
dataHora, nreOP = dataOperacio.split("OP:")
```

```
# CREEM UNA CLAU PRIMÀRIA PER TICKET QUE NO S'HAURIA DE REPETIR MAI
nreFactura = liniaFactura.split("FACTURA SIMPLIFICADA:")[1].strip()
nreOP = nreOP.strip()
clauPrimaria = nreFactura + "_OP" + nreOP
```

2423-026-567893 **4083409**

[2423-026-567893_OP4083409] → **Clave Primaria ticket**

FastAPI: parseo de tickets

Diccionario

```python
jsonTicket = {
    "_id": clauPrimaria,
    "idUsuari": idUsuari_enToken,
    "productesAdquirits": diccProductes,
    "totalTicket": preuTotalTicket,
    "direccioSuper": direccioSuper,
    "data": data_ISO8601,
    "hora" : hora
}
print(json.dumps(jsonTicket, indent=4, ensure_ascii=False))
```

```json
{
    "_id": "2423-026-567893_OP4083409",
    "idUsuari": 3,
    "productesAdquirits": {},
    "totalTicket": 33.36,
    "direccioSuper": "C/ SENYERA 24 46006 VALENCIA",
    "data": "2025-04-30",
    "hora": "21:14"
}
```

FastAPI: parseo de tickets

Diccionario

```
jsonTicket = {
    "_id": clauPrimaria,
    "idUsuari": idUsuari_enToken,
    "productesAdquirits": diccProductes,
    "totalTicket": preuTotalTicket,
    "direccioSuper": direccioSuper,
    "data": data_ISO8601,
    "hora" : hora
}
print(json.dumps(jsonTicket, indent=4, ensure_ascii=False))
```

JSON

```
{
    "_id": "2423-026-567893_OP4083409",
    "idUsuari": 3,
    "productesAdquirits": {},
    "totalTicket": 33.36,
    "direccioSuper": "C/ SENYERA 24 46006 VALENCIA",
    "data": "2025-04-30",
    "hora": "21:14"
}
```

Delimitamos la tabla de productos (gracias a la cabecera)

Figure: Este proceso depende de encontrar la cabecera en la línea siete (funciona para catalán y castellano). Lanzamos excepcion si falla.

1a Detección productos envasados (No granel)

Figure: Procedimiento: primera aproximación a la detección de productos que no son a granel mediante su importe.

1a Detección productos a granel

Figure: Procedimiento: primera aproximación a la detección de productos a granel a partir de la falta de importe en su primera línea.

1a Detección productos envasados (continuación)

$\forall$ producto envasado $\exists$ "d,dd" al final de línea (importe).

- Si se compra $\rightarrow$ **una unidad**, *entonces*:
  - **No** Existe patrón "d,dd" a su izquierda (columna P.Unit)?
- Si se compran $\rightarrow$ **2 o más** unidades, *entonces*:
  - **Sí** Existe patrón "d,dd" a su izquierda (columna P.Unit)?

07/01/2025 19:21 OP: 3322341
FACTURA SIMPLIFICADA: 3960-022-157614

	Descripció	P. Unit	Import
2	12 OUS GRANS L	2,28	4,56
1	BOSSA PLÀSTIC		0,15
1	TOMÀQ. PERA TERRINA		2,20
1	PEBROT FREGIR		
	0,184 kg	2,39 €/kg	0,44
		TOTAL (€)	**7,35**
		TARGETA BANCÀRIA	**7,35**

Se pierde el espacio

```
[ 07/01/2025
212 OUS GRANS L 2,28 4,56
1BOSSA PLÀSTIC 0,15
1TOMÀQ. PERA TERRINA 2,20
1PEBROT FREGIR
0,184 kg 2,39 €/kg 0,44 ]
```

Figure: producto conflictivo envasado: número de unidades queda mezclado con el inicio de la descripción o nombre de un producto imposibilitando segmentar ambos datos mediante espacio (*split()*)

Figure: Solución al conflicto: se calcula qué parte de los dígitos pertenecen al número de unidades adquiridas y qué parte al nombre o descripción del mismo mediante coociente Importe/precioUnitario

FastAPI: parseo de tickets

MERCADONA, S.A. A-46103834

AVDA. ALCALDE ROVIRA ROURE 87
25198 LLEIDA
TELÈFON: 973305750
18/12/2024 12:58 OP: 4003395
FACTURA SIMPLIFICADA: 4037-010-778810

	Descripció	P. Unit	Import
1	PATATA 3 KG		5,00
1	CORS CABDELLS		1,89
1	ESPINACS		1,22
1	BOSSA PLÀSTIC		0,15
1	FILM TRANSPARENT		1,85
1	PIT FAM.		5,52
1	12 OUS PAGÈS		2,67
1	KIWI GROC SAFETA		3,51
1	PORRO TALLAT		2,86
1	PERA ROCHA		
	0,514 kg	1,99 €/kg	1,02
1	CARBASSÓ VERD		
	0,668 kg	1,80 €/kg	1,20
1	ALVOCAT		
	0,460 kg	5,50 €/kg	2,53
1	PARQUING		0,00
	ENTRADA 12:37	SORTIDA 12:58	
		TOTAL (€)	**29,42**

!!!!!

Al no encontrar patrón {d,dd} en la segunda línea, se confunde con la primera línea de un producto a granel.

Por lo tanto se tratará de ver la siguiente línea para encontrar kg, €/kg, etc...

Dado que no hay más líneas aquí dará IndexError

Figure: producto conflictivo: Sale un parking que podemos confundir por un producto envasado (primera línea) y uno a granel (2a línea) que no tendría la línea que lo suele seguir con los datos a extraer.

FastAPI: parseo de tickets

Figure: Solución al conflicto: Saltamos la línea que contiene "PARKING" y la siguiente sin llegar a procesar nada de su contenido: no es de interés.

FastAPI: parseo de tickets

25/06/2024 12:01 OP: 547954
FACTURA SIMPLIFICADA: 2423-016-211701

	Descripción	P. Unit	Importe
1	BRONCHALES 6L		1,26
1	BRONCHALES 1,5L		0,39
1	LECHE SEMI CALCIO		0,96
1	BOLSA PLASTICO		0,15
1	PISTACHO TOSTADO SAL		3,35
1	12 HUEVOS SUPER GRAN		3,25
1	CORAZONES COGOLLO		1,98
1	FRESA		3,34
	PESCADO		
	DORADA		
	0,460 kg	8,75 €/kg	4,03
1	BANANA		
	0,366 kg	1,45 €/kg	0,53
		TOTAL (€)	**19,24**

Figure: producto conflictivo a granel: El producto ocupa tres líneas en vez de dos. El conflicto viene por partida doble: se añade una línea por encima con la categoria y esta primera línea -y la segunda- NO tiene un número “1” de unidades como en el resto de productos a granel.

FastAPI: parseo de tickets

25/06/2024 12:01 OP: 547954
FACTURA SIMPLIFICADA: 2423-016-211701

	Descripción	P. Unit	Importe
1	BRONCHALES 6L		1,26
1	BRONCHALES 1,5L		0,39
1	LECHE SEMI CALCIO		0,96
1	BOLSA PLASTICO		0,15
1	PISTACHO TOSTADO SAL		3,35
1	12 HUEVOS SUPER GRAN		3,25
1	CORAZONES COGOLLO		1,98
1	FRESA		3,34
	PESCADO		
	DORADA		
	0,460 kg	8,75 €/kg	4,03
1	BANANA		
	0,366 kg	1,45 €/kg	0,53
		TOTAL (€)	**19,24**

Figure: Solución al conflicto: No la vamos a implementar por ahora, porque queremos forzar que salgan errores, tal y como sería para la aplicación en producción (tickets no vistos previamente, casos imposibles de preveer sin un enfoque empírico)

FastAPI: parseo de tickets

Figure: Así mostraremos los tickets cuyo parseo ha fallado al usuario y los guardaremos a parte concatenando el id de usuario y el ticket

FastAPI: parseo de tickets

C/ SENYERA 24
46006 VALENCIA
TELÉFONO: 963341702
07/02/2025 12:45 OP: 468149
FACTURA SIMPLIFICADA: 2423-022-563148

	Descripción	P. Unit	Importe
1	PATATA 3 KG		5,10
2	12 HUEVOS GRANDES-L	2,28	4,56
1	POLLO ENTERO LIMPIO		6,44
1	AGUACATE BANDEJA		3,49
1	TOMATE PERA TARRINA		2,37
1	MANDARINA		
	0,264 kg	2,19 €/kg	0,58
1	BANANA		
	0,310 kg	1,52 €/kg	0,47
		TOTAL (€)	**23,01**


```
{
    "_id": "2423-022-563148_OP468149",
    "idUsuari": 3,
    "productesAdquirits": {
        "PATATA 3 KG": {
            "esGranel": false,
            "preuUnitari": 5.1,
            "quantitat": 1,
            "categoria": 13,
            "import": 5.1
        },
        "12 HUEVOS GRANDES-L": {
            "esGranel": false,
            "preuUnitari": 2.28,
            "quantitat": 2,
            "categoria": 13,
            "import": 4.56
        },
        "POLLO ENTERO LIMPIO": {
            "esGranel": false,
            "preuUnitari": 6.44,
            "quantitat": 1,
            "categoria": 13,
            "import": 6.44
        },
        "AGUACATE BANDEJA": {
            "esGranel": false,
            "preuUnitari": 3.49,
            "quantitat": 1,
            "categoria": 13,
            "import": 3.49
        },
        "TOMATE PERA TARRINA": {
            "esGranel": false,
            "preuUnitari": 2.37,
            "quantitat": 1,
            "categoria": 13,
            "import": 2.37
        },
        "MANDARINA": {
            "esGranel": true,
            "preuUnitari": 2.19,
            "quantitat": 0.264,
            "categoria": 13,
            "import": 0.58
        },
        "BANANA": {
            "esGranel": true,
            "preuUnitari": 1.52,
            "quantitat": 0.31,
            "categoria": 13,
            "import": 0.47
        }
    },
    "totalTicket": 23.01,
    "direccioSuper": "C/ SENYERA 24 46006 VALENCIA",
    "data": "2025-02-07",
    "hora": "12:45"
}
```


Figure: Ticket parseado correctamente a su formato JSON (persistible)

Persistencia | FastAPI → mongoDB

MongoDB (NoSQL)	**MySQL (SQL)**
Colección (tickets)	Tabla
Documento JSON → BSON (ticket)	Fila
Campos anidados: esquema flexible	Campos fijos: esquema rígido

Table: Comparación entre MongoDB y MySQL

Persistencia | FastAPI → mongoDB

Se ha escogido MongoDB porque es una base de datos noSQL, ideal en el caso que aquí nos ocupa:

- No existe un número **fijo** de productos en un ticket.
- **No tenemos** un listado exhaustivo de todos los productos del supermercado.
- En el *"inflalyzer"* queremos hacer **filtros por productos que todavía no conocemos**.

Figure: Guardar los tickets en formato BSON tiene una implicación evidente más allá de la posibilidad de hacer búsquedas.

FastAPI: exposición tickets (+ dashboard)

Figure: Extracción de datos de un endpoint de fastAPI desde su salida del repositoriTickets.py (donde hacemos la consulta en MongoDB) hasta su llamada a controlador.py a través de POSTMAN mediante token de acceso (permisos a 1)

FastAPI: exposición tickets (+ dashboard)

```javascript
//NOMBRE PRODUCTES DIFERENTS ADQUIRITS
fetch('http://localhost:8000/api/frequenciesProductes', {
    method: "GET",
    headers: {
        'Authorization': "Bearer " + localStorage.getItem("AccessToken"),
        'Accept': 'application/json'
    }
})
.then(response => {
    if (!response.ok) {
        throw new Error('La solicitud devolvió código de error en paso3 : ' + response.sta
    }
    return response.json();
})
.then(diccDades => {  //estil --> {'BOLSA PLASTICO': 132, 'BANANA': 53, 'BRONCHALES 6L': 5
    const dom_nreProdDiferents = document.getElementById("nreProductesDiferentsAdquirits")
    dom_nreProdDiferents.innerHTML = Object.entries(diccDades).length;
})
.catch(error => {
    console.error('Error en paso3:', error);
});
```

Figure: En lugar de postman ahora lo sacamos hacia el front con la función Fetch de JavaScript. Sacamos números de pares clave:valor del JSON (Nótese: Python (*dict*) → JSON → JavaScript(*Object*))

FastAPI: exposición tickets (+ dashboard)

```
//NOMBRE PRODUCTES DIFERENTS ADQUIRITS
fetch('http://localhost:8000/api/frequenciesProductes', {
    method: "GET",
    headers: {
        'Authorization': "Bearer " + localStorage.getItem("AccessToken"),
        'Accept': 'application/json'
    }
})
.then(response => {
    if (!response.ok) {
        throw new Error('La solicitud devolvió código de error en paso3 : ' + response.sta
    }
    return response.json();
})
.then(diccDades => {  //estil --> {'BOLSA PLASTICO': 132, 'BANANA': 53, 'BRONCHALES 6L': 5
    const dom_nreProdDiferents = document.getElementById("nreProductesDiferentsAdquirits")
    dom_nreProdDiferents.innerHTML = Object.entries(diccDades).length;
    localStorage.setItem("frequenciesProductes", Object.entries(diccDades)); //guardem le
})
.catch(error => {
    console.error('Error en paso3:', error);
```

Figure: Persistimos los datos en el localStorage del cliente. Volveremos a usarlos después en el "inflalyzer" y así ahorramos sucesivas llamadas a BBDD para unos datos que ya se extrayeron una vez

FastAPI: exposición tickets (+ dashboard)

POST http://localhost:8000/api/graficDataPreuProducte

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"nomProducte": "POLLO ENTERO LIMPIO"}
```

Body Cookies Headers (4) Test Results

{} JSON Preview Visualize

```
1 [
2     {
3         "x": "2024-05-30",
4         "y": 5.81
5     },
6     {
7         "x": "2024-07-17",
8         "y": 5.97
9     },
```

Figure: Endpoint que permite hacer el gráfico de inflación de un producto, obteniendo sus puntos X (fecha aaaa-mm-dd) e Y (€/kg o €/unidad)

FastAPI: exposición tickets (+ dashboard)

Figure: Endpoint utilizado para obtener los gastos totales por cada una de las 13 categorías de alimentación.

Front-end: Vanilla JS

Front-end: Vanilla JS

Front-end: Vanilla JS

Front-end: Vanilla JS

Front-end: Vanilla JS

Front-end: Vanilla JS

Demostración!

Contenidos

1 Introducción

2 Diseño
- Requisitos
- Diagramas de sistemas

3 Desarrollo
- Entornos de desarrollo
- Despliegue
- Spring Boot: gestión usuarios
- FastAPI: parseo de tickets
- FastAPI: exposición tickets (+ dashboard)
- Front-end: Vanilla JS

4 Conclusiones

Conclusiones

- Definir y desarrollar dos back-ends desde cero, uno para usuarios y otro para tickets digitales, integrándolos entre sí.
- Aprender y manejar bases de datos NoSQL (MongoDB) y relacionales (MySQL), junto con el mapeo de objetos Java a bases relacionales usando JPA.
- Entender y programar la gestión de autenticación y autorización con JWT en Java, creando clases personalizadas para generar y validar tokens.
- Programar el front-end con JavaScript puro, controlando vistas según tokens de acceso y aplicar CSS desde cero, incluyendo media queries sin frameworks.
- Contenerizar tanto back-end como front-end y automatizar despliegues. Configurar CORS.

¡Gracias Por vuestra atención!
¿Preguntas?