Creación de un dashboard para usuarios del ticket digital de Mercadona: que permita consultara evolución de productos habitualmente adquiridos, el coste de sus compras por períodos temporales y el monto de gastos por áreas de producto

Santiago Sánchez Sans

Ciclo formativo en desarrollo de aplicaciones web

Memoria del Proyecto de DAW

IES Abastos. Curso 2024/25. Grupo 7X. XX de Junio de 2025

Tutor Individual: Carlos Furones

Índice general

Introducción

1.1. ¿Qué es el ticket digital de Mercadona?

Mercadona implementa un sistema de tickets digitales que vinculan la tarjeta de débito a un correo electrónico. Cualquier usuario del supermercado que quiera utilizar el ticket digital solamente deberá facilitar estos dos datos y el supermercado le enviará por correo electrónico los tickets de las posteriores compras hechas en cualquier establecimiento de Mercadona.

Las ventajas para el usuario son evidentes: no se pierden los tickets de cara a devoluciones y el cliente del supermercado no debe esperar a la impresión del ticket.

1.2. Identificación de necesidades

Los tickets de cada usuario se acumulan de forma recurrente en el correo electrónico y con un formato estructurado (los asuntos son predecibles e incluyen las fechas) y dentro de cada correo de un ticket digital se encuentra un PDF con el desglose de la compra (producto, unidades vendidas, establecimiento, etc).

Esta información se acumula en el correo del usuario pero a pesar de ser una información estructurada su acceso para el usuario no es simple: no puede visualizar lo que ha gastado, ni el precio de los productos y de su evolución, ni los supermercados en los que ha comprado, ni las veces que lo ha hecho, etc.

1.3. Objetivos del proyecto

Este proyecto quiere responder a estas necesidades. Para ello se plantea la Creación de un dashboard o çuadro de mando"front-end para que un usuario del ticket digital de Mercadona pueda visualizar la evolución de precios de los productos adquiridos, el coste promedio de sus compras por períodos temporales y sus distribuciones de gastos a partir de los tickets digitales guardados en una base de datos.

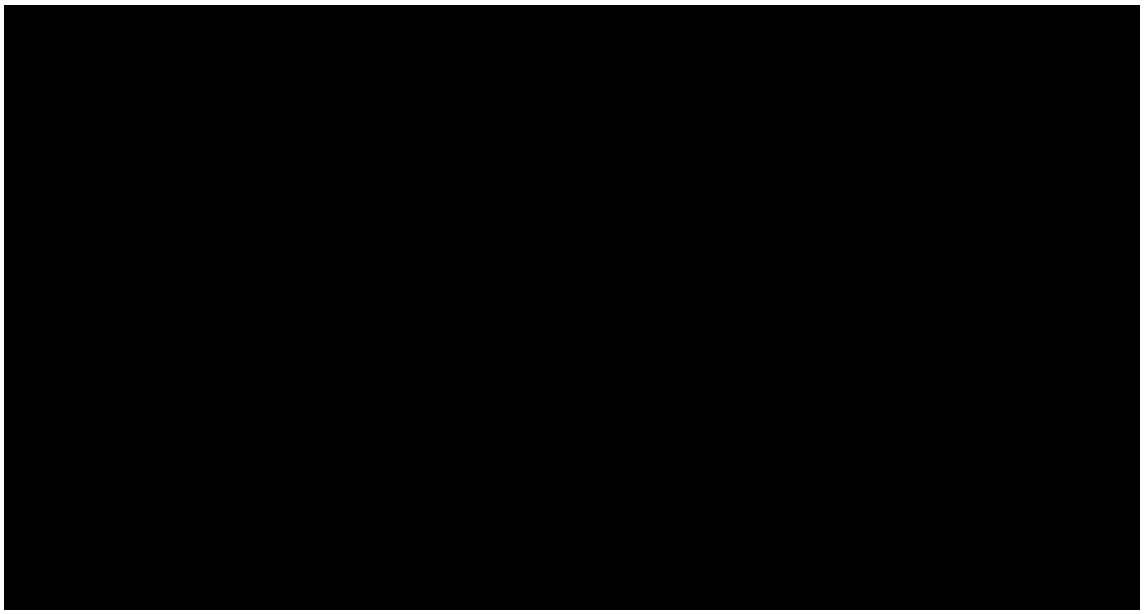
Los **Objetivos principales** del proyecto son mostrar al usuario:

- **Evolución de precios** (inflación) a lo largo del tiempo en los productos

habitualmente comprados en el mismo establecimiento[1].

- **Evolución del gasto** total del usuario a lo largo del tiempo.

Figura 1.1: nuestra "NavBar": fondo negro, letra blanca y un solo desplegable.



[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...).

Diseño del proyecto

Para implementar los objetivos principales de los que hemos hablado en la sección 1.3 hemos proyectado una serie de requisitos funcionales de la aplicación.

2.1. Requisitos Funcionales

2.1.1. Requisitos de la aplicación

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). Para los productos de precios muy variables (productos a granel, como frutas, etc.), se mostrará la evolución del precio por kg a lo largo del tiempo.

REQUISITO B: Coste total de la cesta de la compra del usuario a lo largo de distintas ventanas temporales (por meses, períodos de 3, 6 meses y un año), independientemente del centro de Mercadona en el que se compre (todos juntos).

REQUISITO C: Al lado de este mismo coste total mostrado en B), se incluirá un diagrama de queso (o de sectores) desglosando qué porcentaje del dinero se ha destinado a cada una de las siguientes categorías: verdura y hortalizas, frutas, huevos y lácteos, agua y bebidas, aceite y especias, carnes, pescado, hogar e higiene personal. Para ello, dado que no tenemos categorizados todos los productos de Mercadona ni podríamos hacerlo por falta de una lista exhaustiva y de tiempo, se usará un modelo predictivo con word embeddings (módulo Spacy) y cosine similarity (sklearn) para encontrar distancias pequeñas entre las descripciones de los tickets y las categorías, facilitando así la clasificación.

REQUISITO D: Un botón "Actualizar" permitirá al usuario refrescar los datos desde el servidor cuando haya añadido nuevas compras. También podríamos permitir que los PDFs descargados en el servidor se almacenen en una carpeta local del usuario para que pueda verificar la extracción de los datos.

REQUISITO E: Hacer un sistema front-end y back-end que permitan redirigir a los usuarios rápidamente a un registro de forma inteligente. Nos inspiraremos en el sistema de registro e iniciar sesión de NetFlix **(POSAR DIAGRAMA A L'ANEX SOBRE EL SISTEMA DEL NETFLIX I EL NOSTRE)**

2.1.2. Requisitos de los usuarios

El correo electrónico y la contraseña de la cuenta de Gmail de alguien que sea usuario del ticket digital de Mercadona y tenga decenas de tickets digitales por analizar, con compras estables y productos recurrentes.

Nota: En la demo se proporcionarán ya muchos tickets digitales (tickets míos, que cederé para mostrar la utilidad de la aplicación). No será necesario recurrir a la extracción de datos de otro usuario de ticket digital. Se mostrarán un mínimo de tickets digitales en un mismo centro de Mercadona para poder evidenciar la evolución de precios y gastos.

2.2. Stack tecnológico

Hemos escogido un stack tecnológico que permite que seamos fieles a los requisitos funcionales que nos hemos marcado para la aplicación:

2.2.1. Front-End: HTML, CSS y Javascript

Se ha usado HTML, CSS y JavaScript. - Para la visualización de gráficos se usará una librerría javascript: https://www.chartjs.org/. Entre otras cosas se utilizará para hacer los gráficos de la evolución de precios por producto.

2.2.2. back-end: Java (SpringBoot) y Python

- Back-end con Java (springboot para el login y la autenticación de usuarios: con este framework guardaremos datos en la BBDD mySQL).

- Python dentro de un contenedor docker (o python a secas, para descargar los pdfs del correo) y parsear el contenido de los mismos: con sklearn, numpy y spacy que luego se podrán pasarlos a la BBDD mongoDB.

2.2.3. BBDD: MySQL y MongoDB

Para guardar los datos de los usuarios se debe usar un sistema de gestión de base de datos relacional. Hemos escogido MySQL dado que es el que hemos visto en el grado superior y estamos bien versados en ello.

Sin embargo, los productos de Mercadona no los conocemos de antemano ni tenemos una lista exhaustiva de los mismos. Además, el número de productos que

se pueden encontrar en un ticket varía en cada compra, por lo que no podemos usar una base de datos relacional tradicional como MySQL o PostgreSQL por que se trata de información no estructurada. En su lugar, usaremos MongoDB, una BBDD NoSQL que almacena datos en formato JSON y permite, además, búsquedas eficientes.

Para optimizar el backend, intentaremos que un usuario pueda consultar repetidamente sus compras sin sobrecargar el servidor. La primera vez que consulte sus datos, estos se descargarán y almacenarán en localStorage del cliente. En consultas posteriores, los datos se obtendrán directamente de localStorage sin necesidad de hacer peticiones al servidor. Evaluaremos la viabilidad de este sistema durante el desarrollo; Esto es la fase de diseño y como tal, **PUEDE QUE EN LA FASE DE DESARROLLO CAMBIE**, en caso de no ser factible, las consultas se harán directamente en MongoDB.

2.3. Diagramas de la aplicación

Fer un diagrama guapo de tots els components de l'aplicació.

Desarrollo del proyecto

3.1. GitHub del proyecto

Para desarrollar este proyecto se ha trabajado con GitHub y git. Dado que no ha habido trabajo en equipo no se han utilizado pull requests a la rama main sino simplemente se ha seguido la estrategia de crear ramas de característica y, una vez son satisfactorias, hacer un merge en la rama main en local.

Un flujo de trabajo habitual es mediante ramas de característica (puede verse anexo 5.1). También puede verse el GitHub del proyecto a continuación. Dentro del readme del proyecto encontraréis instrucciones para su descarga, clonado y "despliegue" de sus componentes en vuestro ordenador personal si así lo deseáis.

Link al repositorio	https://github.com/blackcub3s/mercApp
Página desplegada	TO DO

Cuadro 3.1: Enlaces importantes del proyecto.

3.2. Entornos de desarrollo

Java SPRINGBOOT con IntelliJ Idea community edition Frontend y python con VScode

3.3. desarrollo del front-end

3.4. desarrollo back-end

Evaluación y Conclusiones Finales

ANEXO

5.1. Flujo de trabajo habitual en git

```
# trabajamos con el proyecto y se introduce
# en el staging area
git add -A

# creamos rama para aglutinar los cambios
git branch backEnd

# cambiamos a la rama que acabamos de crear
git checkout backEnd

# guardamos los cambios como nodos dentro de
# la rama con la que desarrollamos.
git commit -m "commit 1"
git commit -m "commit 2"
# [...]
git commit -m "commit n"

#cambiamos a rama main local y luego integramos cambios
git checkout main
git merge backEnd

#Subimos los cambios al repo remoto
git push origin main
```

5.2. CODIGO 2

5.3. CODIGO 3

5.4. CODIGO 4

5.5. CODIGO 5

Bibliografía