Ejercicio Evaluación: Super Sneakers

Universidad de Alicante — November 27, 2021

- Daniel Asensi Roch 48776120C

Contents

1 Requerimientos a implementar por la Arquitectura

En este supuesto se nos pedirá diseñar una plataforma de comercio electrónico para la compra de zapatillas deportivas, generando una arquitectura que sea capaz de soportar los servicios y las funcionalidades básicas de una tienda en línea, así como los pagos, el marketing y el envió de productos, contando la misma con su interfaz web y móvil.

1.1 Funcionalidades de Loggin

- Registro de cuenta única que permita el consumo de todos los servicios
- Inicio de sesión centralizado
- Añadir artículos a la cesta
- Editar artículos de la cesta
- Eliminar artículos de la cesta

1.2 Funcionalidades de Cesta

- Añadir artículos a la cesta
- Editar artículos de la cesta
- Eliminar artículos de la cesta

1.3 Funcionalidades de catálogo

- Enumerar artículos del catálogo
- Filtro de artículos por tipo
- Filtro de artículos por marca

1.4 Funcionalidades de Marketing

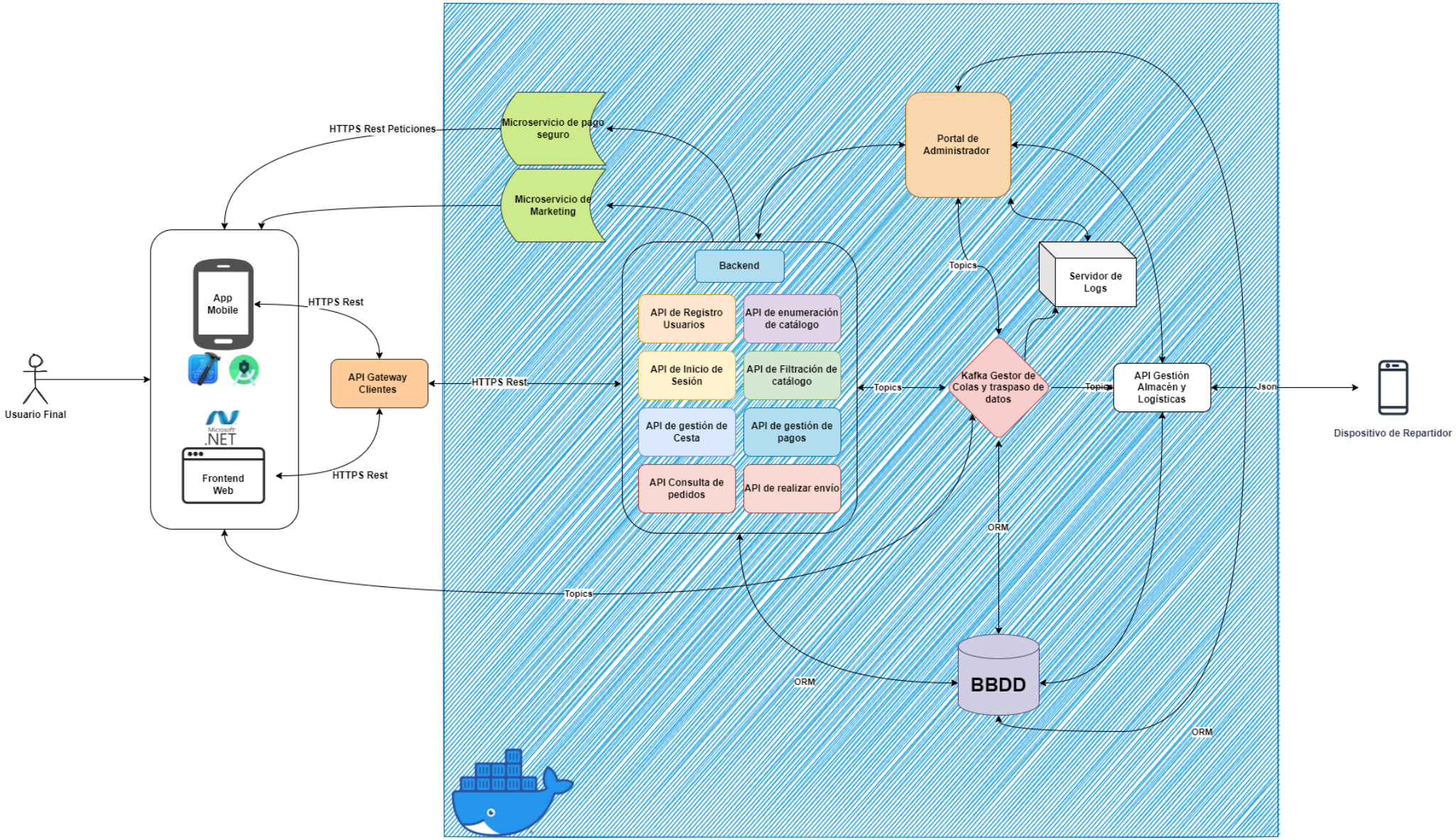
- Obtener información de marketing personalizada

1.5 Requisitos no funcionales

Estos requisitos serán implementados de manera implícita en nuestra arquitectura de manera que queden satisfecho para el usuario final, algunos requisitos no funcionales especificados para este diseño que quedan fuera de los estándares de la fácil escalabilidad y el desarrollo multiplataforma son:

- El acceso de usuarios a los servicios debe estar centralizado.
- Los servicios de gestión de identidades, de catálogo de productos, de pedidos y de la cesta de la compra, deben ser fácilmente reutilizables para futuras tiendas en línea que el cliente desea crear en un futuro.
- Debe incluir una herramienta de marketing que muestre a los usuarios información de productos que puedan ser de su interés. También debe ser fácilmente reutilizable.
- Todos los servicios deben comunicarse entre ellos utilizando mecanismos de comunicación asíncrona.
- Los logs generados por todos los servicios deben estar centralizados y se debe poder acceder a ellos desde el portal de administración.

2 Implementación de la arquitectura.


HTTPS Rest Peticiones
Microservicio de pago seguro
Microservicio de Marketing
Portal de Administrador
Backend
API de Registro Usuarios
API de enumeración de catálogo
API de Inicio de Sesión
API de Filtración de catálogo
API de gestión de Cesta
API de gestión de pagos
API Consulta de pedidos
API de realizar envío
Topics
Servidor de Logs
App Mobile
HTTPS Rest
Usuario Final
API Gateway Clientes
HTTPS Rest
Kafka Gestor de Colas y traspaso de datos
Topics
API Gestión Almacén y Logísticas
Json
Dispositivo de Repartidor
Frontend Web
HTTPS Rest
Topics
ORM
ORM
BBDD
ORM

3 Explicación de la implementación

3.1 Especificación de Tecnologías Utilizadas

Para el dasarrollo de la aplicación y su posterior puesta en marcha para el cliente utilizaríamos el entorno de Visual Studio proporcionado por Microsoft además de la herramienta de control de versiones Github con Docker para su posterior lanzamiento a producción y para salvaguardar errores de dependencias y malas configuraciones en los entornos del servidor de los futuros clientes.

3.2 Base de datos

El tipo de base de datos utilizada sería MySql con su gestor MariaDB, ya que Microsoft nos proporciona una cómoda gestión desde el entorno de desarrollo de Visual Studio.

3.3 Entornos de desarrollo

El entorno de ejecución será el de .NET FrameWork proporcionado por Microsoft.

.NET Framework ya que este se infinidad de herramientas para desarrolladores tanto del backend como el frontend, lo que quiere decir que será un entorno para desarrolladores fullStack, esto facilitará la implementación y la integración para los desarrolladores web.

El código del backend será realizado en C# y conectado un ORM siendo este un modelo de programación que nos permitirá mapear las estructuras de una base de datos relacional (MySQL,MariaDB), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones, para ello utilizaremos Dapper ORM.

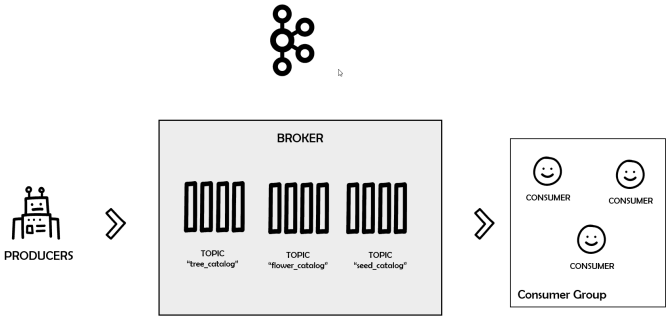
Para el desarrollo del frontend utilizaremos Razor Pages en nuestra WebApplication. Para el desarrollo de la aplicación móvil utilizaremos Android Studio para la aplicación que se ejecutará en los dispositivos Android, y para el desarrollo en dispositivos IOS utilizaremos SwiftUI en el entorno de desarrollo de XCode lo que permitirá un rápido desarrollo.

3.4 Despliegue de servicios

Para desplegar los servicios de nuestra aplicación utilizaremos imágenes Docker teniendo estos encapsulados, con las configuraciones necesarias y permitiendo una alta escalabilidad, de esta manera podremos desarrollar aplicaciones que puedan correr en diferentes buscadores, siendo estos por ejemplo Google-Chrome, Mozilla Firefox o Opera, lo que a su vez permitirá compatibilidad con diferentes sistemas operativos como Windows, Gnu/Linux o Mac-Os.

3.5 Mensajería y gestión de colas

Usaremos Kafka para la mensajería y la gestión de colas eso nos dejará una total libertad en la creación de Topics para el envio y recibo de mensaje, todos los mensajes serán escritos en nuestros Logs.



Esquema de uso de Kafka

3.6 Seguridad de la información

Para mayor seguridad en nuestra Web compraremos el dominio de <dominiocomprado> y utilizaremos HTTPS en la comunicación con el frontend y el backend.

3.7 Host y Logs

El host que utilizaremos para nuestra aplicación será Amazon Web Services para nuestro servidor en la nube, y usando su ventana Elastic para nuestra para nuestros micro servicios, además para los micro servicios utilizaremos el cloud sin servidor de SAM.

Los logs serán almacenados mediante los topics que se generarán en nuestro gestor de colas (kafka), todos ellos serán almacenados para que los administradores puedan acceder a ellos y comprobar que ha pasado en la web.

3.8 Documentación de API's

Se utilizará la aplicación de Postman para realizar la documentación de nuestras API's ya que cuenta con una intuitiva interfaz gráfica y la capacidad de ver los archivos JSON que recibiremos desde las mismas.

4 FrontEnd

4.1 Aplicación Web

El desarrollo de las aplicaciones web como ya he explicado previamente se realizará en .NET Framework proporcionado por Microsoft en concreto utilizaremos Razor Webpage, que nos brindará muchas más utilidades que el entorno de desarrollo base de .NET además del .aspx embeberemos algo de codigo JS y HTML para las validaciones base de los datos de los usuarios en el registro y inicio de sesión.

4.2 Aplicaciones Móviles

Las aplicaciones moviles serán desarrolladas de manera sencilla para IOS y Android en, XCode y Android Studio respectivamente.

5 Backend

Todo el backend incluida las Query de conexión a la BBDD serán escritas en C# para mayor sencillez y para evitar la mezcla de lenguajes en el BackEnd además del uso de Dapper para las conexiones.

5.1 API's y microservicios

En cuanto a las API's los datos serán enviados de manera encriptada en https en un archivo json con el cuerpo de la petición a un endpoint del backend el cual será especificado desde la url mediante los metodos ya conocidos que usará la bbdd y responderá con un archivo json.

6 Servicios Implementados

6.1 Servicio de gestión de Identidad

6.1.1 Loggin

En el servicio de Loggin se dispondrá ante el usuario una ventana donde este deberá introducir sus credenciales, correo electrónico y contraseña, mediante código JS alojado en el Front se verificará que estos campos sean adecuados, acto seguido se enviarán a la BBDD y se iniciará la sesión del usuario, la sesión será guardada mediante cookies, estas se eliminarán cada 10 minutos a no ser que el usuario recargue la página o realice alguna petición al servidor. Su URL estará especificada de la siguiente manera.

```
URL POST - dominiocomprado.es/api/loggin
```

6.1.2 Register

En el servicio de registro se dispondrá ante el usuario una ventana donde este deberá introducir sus credenciales, correo electrónico y contraseña, mediante código JS alojado en el Front se verificará que estos campos sean adecuados, acto seguido se enviarán a la BBDD y se registrará al usuario Su URL estará especificada de la siguiente manera.

```
URL POST - dominiocomprado.es/api/register
```

6.2 Catálogo

En el catálogo la página principal de nuestra tienda los usuarios tanto registrados como no registrados y de los cuales se haya almacenado una coockie en el navegado, podrán visualizar todos los artículos almacenados en la BBDD, además mediante esta coockie y el servicio de marketing se podrán vislumbrar a los lados de nuestro catálogo los artículos los cuales el usuario ha comprado previamente o ha metido en la cesta. La URL de nuestro catálogo será la siguiente.

```
URL GET - dominiocomprado.es/api/catalog
```

Con esta URL sacaremos todos los item del catálogo y modificando los campos correctos de la URL desde el frontEnd filtraremos el mismo para mostrar los artículos deseados y accediendo a la API del filtro realizaremos el filtrado del mismo.

```
URL GET - dominiocomprado.es/api/catalog=%type="all"&&catalog=%brand="all"
```

6.3 Cesta de la compra

Los usuarios que se encuentre loggueados en la página podrán acceder a la funcionalidad de la cesta de la compra, en ella podrán añadir o eliminar artículos, una vez el usuario añadido los artículos a la cesta estos serán enviados al microservicio de marketing el cual almacenará la información de los artículos que ha añadido el usuario a su cesta para que cuando el usuario entre de nuevo en la página se muestren a los lados de la misma los artículos que ha comprado recientemente.

```
URL POST - dominiocomprado.es/api/basket
```

6.4 Comprar

Una vez seleccionados todos los artículos que se desea el usuario podrá pasar a realizar el pago, para ello primero se validarán todos los datos introducidos y se enviarán directamente al microservicio de pago seguro donde se procesarán con las credenciales de la empresa, los datos de pago de los usuarios no se guardarán en ningún momento, ya que eso podría ser una brecha de seguridad en nuestro sistema que podría se susceptible a varios hackeos.

Una vez realizado el pago se enviará mediante la API de gestión de pedidos alojada en el backend y mediante ORM se almacenará el pedido en la BBDD y acto seguido a la API del almacén y gestión de pedidos.

```
URL POST - dominiocomprado.es/api/buysistem%type="state"
URL GET - dominiocomprado.es/api/buysistemstate%type="state"
```

En el campo state de la url será el que nos diga el estado del pago, este enviará cifrado.

6.5 Pedidos

En esta API se utilizará la coockie de inicio de sesión del usuario para sacar todos los pedidos que ha realizado el usuario desde la BBDD, estos se enviarán al frontend mediante json para que se puedan visualizar de forma segura.

```
URL GET - dominiocomprado.es/api/order/:id
```

Estos serán enviados extrayéndolos por el id del usuario.

6.6 Gestión de almacenes y logística

A esta API se enviará toda la información relacionada con los artículos que deben ser enviados desde los almacenes, esta API tendrá un método get para que los gestores del almacén puedan preparar los pedidos y una vez preparados los pedidos se enviará la información al frontend de la app de gestión de los repartidores para que puedan enviar el pedido a los hogares de los usuarios.

```
URL GET - dominiocomprado.es/api/order2send/:id
URL GET - dominiocomprado.es/api/ordersend/:id
```

6.7 Panel de gestión de los administradores

El panel de gestión de los administradores tendrá el control de todos los topics, productores, consumidores y API's para poder gestionar la información que llega a cada una de las API's, para ello además de poder modificar el código y información de los archivos json, estos serán capaces de entrar al archivo de logs y usar Dapper para acceder a la BBDD mediante ORM. Sus Endpoint serían los siguientes:

```
URL GET - dominiocomprado.es/api/order2send/:id
URL GET - dominiocomprado.es/api/ordersend/:id
URL POST - dominiocomprado.es/api/order2send/:id
URL POST - dominiocomprado.es/api/ordersend/:id
URL DEL - dominiocomprado.es/api/order2send/:id
URL DEL - dominiocomprado.es/api/ordersend/:id
URL PUT - dominiocomprado.es/api/order2send/:id
URL PUT - dominiocomprado.es/api/ordersend/:id


URL GET - dominiocomprado.es/api/order/:id
URL PUT - dominiocomprado.es/api/order/:id
URL DEL - dominiocomprado.es/api/order/:id
URL POST - dominiocomprado.es/api/order/:id

URL POST - dominiocomprado.es/api/buysistem%type="state"
URL GET - dominiocomprado.es/api/buysistemstate%type="state"
URL DEL - dominiocomprado.es/api/buysistem%type="state"
URL DEL - dominiocomprado.es/api/buysistemstate%type="state"
URL PUT - dominiocomprado.es/api/buysistem%type="state"
URL PUT - dominiocomprado.es/api/buysistemstate%type="state"

URL GET - dominiocomprado.es/api/order/:id
URL DEL - dominiocomprado.es/api/order/:id
URL POST - dominiocomprado.es/api/order/:id
URL PUT - dominiocomprado.es/api/order/:id
```

Además cabe recalcar que los administradores podrán acceder al historial de todos los mensajes enviados entre los productores y consumidores de gestor del colas que quedarán almacenados en formato texto en los logs siguiendo la siguiente estructura

```
[Date][Topic...Sistem...API][User...cookie][Action-performed][Trace]
```

6.8 Gestor de colas Kafka

Este será el servidor de mensajería que utilizaremos para nuestra aplicación, el porque del uso de esta tecnología es por su versatilidad y integración con varios lenguajes, esto nos permitirá producir en JS y consumir en C# y viceversa, esto será de gran ayuda para los desarrolladores ya que solo deberán ponerse de acuerdo en el formato del mensaje que se enviará, los topics que utilizaremos para comunicar nuestras aplicaciones serán los siguientes:

```
Topics:
Registro de Usuarios: User-Register, id, contraseña, nombre
Iniciar sesión: user-alt, id, contraseña -> recibirá una coockie confirmed-alt tokencookie
Catálogo: catalog-show.    o si es con filtros catalog-show-filter -p[option] -a[searched]
Pedidos: order-confirmed
Envio: order-send
Compra: buy-state
```

7 Traza de uso de la aplicación

Se muestra una traza de uso de la aplicación en el registro, inicio de sesión, compra y envío de un artículo:

1. Usuario accede a la web o aplicación desde algún dispositivo con connecitivad a internet.
2. Se registra utilizando sus credenciales que serán almacenadas en nuestra BBDD.
3. Iniciará sesión en nuestra página web accediendo con las credenciales previamente establecidas, se buscará al usuario en la BBDD y si se encuentra se le adjudicará una cookie o token que tendrá durante el tiempo que dure sus sesión.
4. El usuario visualizará los productos de la página, si selecciona uno o añade uno a la cesta el microservicio de marketing registrará la acción y la próxima vez que se loggue se mostrará el artículo de interés a los lados de la página.
5. .El usuario añade un artículo a la cesta.
6. .Se procede al pago, si hay suficiente stock en los almacenadas (cosa que estará contemplada en la bbdd)se procederá al pago utilizando nuestro microservicio a elección de pago seguro.
7. Cuando dicho pago esta confirmado se enviará el mensaje a la API de gestion y almacen para que se prepare el pedido
8. Una vez el pedido es preparado se enviará la información de reparto al dispositivo móvil del repartidor.