DESCRIPCIÓN DEL PROYECTO-MICROSERVICIO PEDIDOS

Escuela Técnica Superior de
Ingeniería Informática

Datos del grupo

- María Lourdes Linares Barrera (marlinbar@alum.us.es)
- Pablo Reina Jiménez (prjimenez@us.es)

ÍNDICE DE CONTENIDOS

DATOS DEL GRUPO Y NIVEL DE ACABADO

DATOS DEL EQUIPO	
APLICACIÓN	Grupo 2. Temática: Noctua Sapienta. Aplicación para gestionar la compra y venta de libros a través de Internet.
MICROSERVICIO	Microservicio de pedidos. Encargado de gestionar los pedidos de libros realizados en la aplicación por los usuarios a los vendedores.
REPOSITORIOS	Repositorio del backend servicio de pedidos: https://github.com/Noctua-sapientia/orders-service
	Repositorio de frontend común: https://github.com/Noctua-sapientia/frontend
	Organización de Github con todos los repositiorios: https://github.com/Noctua-sapientia
	Repositorio de github con la documentación (Customer Agreement, SLA, Análisis de la Capacidad, Pruebas de Aceptación de Usuario, Presentación, Demostración de aplicación): https://github.com/Noctua-sapientia/docs/tree/main
NIVEL DE ACABADO	Superior a 7 (se han realizado algunos extras adicionales entre el 7 y el 9)

DESCRIPCIÓN DE LA APLICACIÓN

La principal función de nuestra aplicación es la compra y venta de libros online. El servicio permite registrar nuevos compradores y vendedores además de iniciar sesión con los ya existentes. Asimismo, permite crear libros a los usuarios vendedores y comprar y publicar reseñas a los usuarios compradores.

Centrándonos en el microservicio de pedidos, este representa la lógica de funcionamiento de la aplicación. En definitiva, permite a los compradores realizar nuevos pedidos añadiendo libros a su cesta de la compra y comprándolos. También les facilita consultar sus pedidos.

Por si parte, los vendedores igualmente tienen opción de acceder al sistema y consultar los pedidos que les han hecho, pudiendo modificar el estado del pedido o su fecha de entrega, entre otros aspectos.

En el frontend, el microservicio de pedidos está representado por tres vistas principales: el carrito de la compra y el historial de pedidos.

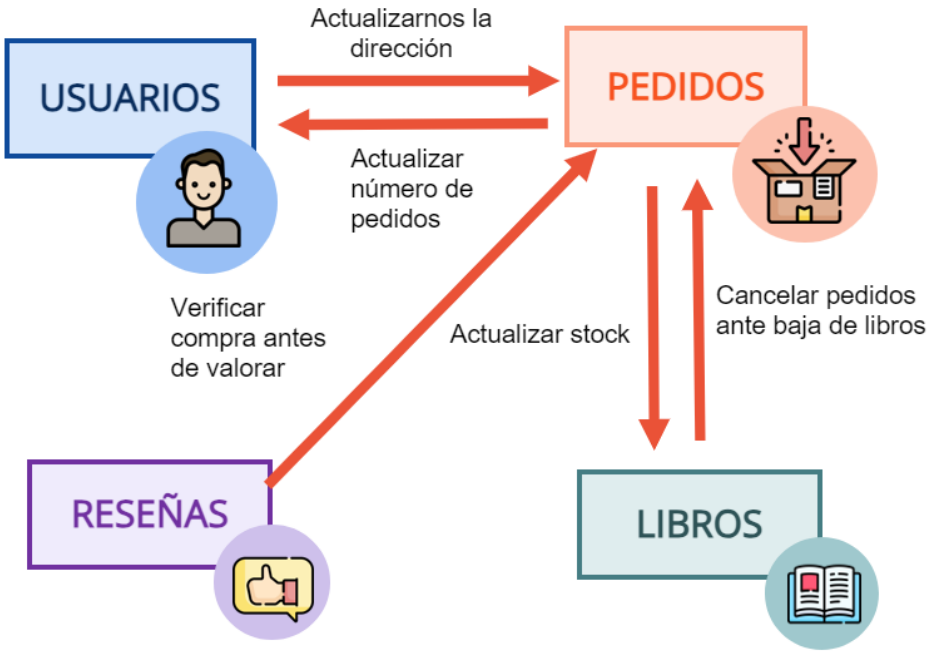
- El carrito de la compra solo está disponible para los usuarios compradores y en él podemos ir agregando libros de distintos vendedores, seleccionar la cantidad de cada uno o eliminarlos y realizar los pedidos deseados. El carrito se gestiona utilizando el localStorage e implementando una lógica que permite añadir libros a carritos ya existentes o a nuevos carritos (según si hay carritos activos del comprador al vendedor en cuestión o no).
- En cuanto al historial de pedidos, para los vendedores, aparecerán los pedidos que le han realizado, mientras que para los compradores aparecerán los pedidos que han realizado. Para cada pedido se proporciona información sobre su identificador, la fecha de inicio del pedido, fecha estimada de entrega precio y estado, pudiendo ser este en preparación, enviado, entregado o cancelado. Podemos filtrar y ordenar este listado de pedido.
- Además, podemos acceder a los detalles del pedido. En esta vista, a la que se navega desde la anterior, se proporciona información extra sobre cada uno, como el nombre del comprador y vendedor, dirección de entrega y listado detallado de los libros. En la vista de detalles, los vendedores podrán actualizar el estado del pedido o su fecha de entrega, en caso de considerarlo oportuno por las necesidades del negocio. También se puede eliminar un pedido en estado terminal (entregado o cancelado), si el usuario deja de estar interesado que dicho pedido figure en su listado de pedidos.

En el backend, se implementan las acciones básicas de visualización, creación, modificación y borrado de pedidos, que se exponen en la API para poder ser consumidas por el frontend. Además, se proporcionan una serie de endopoints que tienen por objeto facilitar la comunicación o llamadas con otros microservicios con el microservicio de pedidos.

DESCOMPOSICIÓN EN MICROSERVICIOS

La aplicación consta de 4 microservicios: usuarios, pedidos, libros y reseñas, siendo nuestro microservicio asignado pedidos.

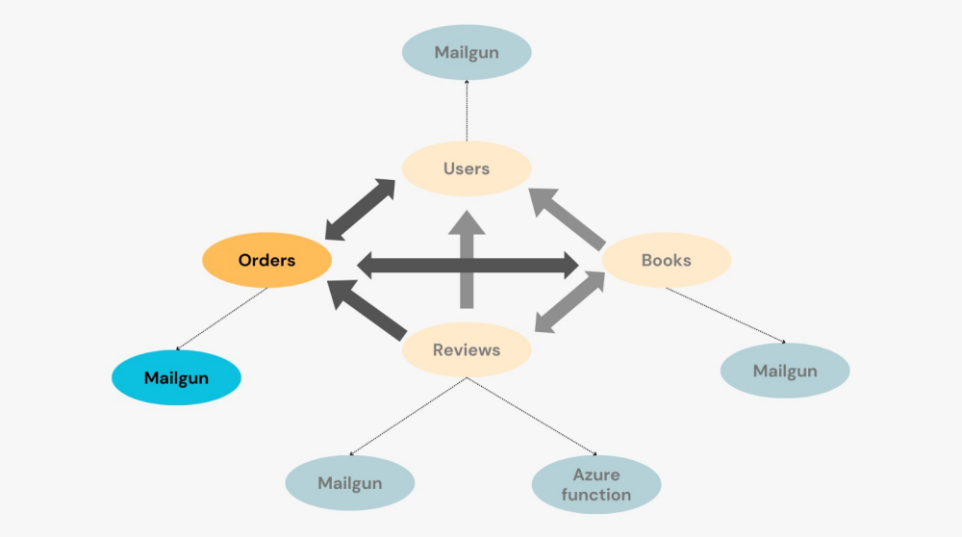
Al implementar una lógica de funcionamiento crucial, el microservicio de pedidos tiene un importante número de comunicaciones con los demás microservicios; más allá de las comunicaciones con el frontend descritas en la sección anterior. Dichas comunicaciones internas del servicio de pedidos con el resto de servicios, están descritas visualmente en la figura inferior.



El servicio cuenta con conexiones tanto de entrada como de salida.

- La conexión con el microservicio de libros se centra en realizar de forma activa un aumento o disminución del stock al realizar un pedido o al cancelarlo. También se expone un método en a API para ser consumido por el microservicio de libros, con el fin de que si un vendedor borra de su listado de libros ofertados algún libro, puedan cancelarse todos los pedidos en curso de ese libro asociados a ese vendedor.
- Por su parte, el microservicio de reseñas consulta en nuestro microservicio si un comprador ha comprado un determinado libro. Solo en este caso le permitirá valorar el libro.
- Con las llamadas a usuarios, podemos actualizar el número de pedidos que ha realizado o recibido un comprador o vendedor respectivamente. Además, exponemos un método en la API que facilita actualizar la dirección de entrega de los pedidos que están aún en reparto si un comprador actualiza su dirección.

En esta figura podemos visualizar la lógica de comunicaciones internas de este servicio dentro de la lógica de comunicaciones globales de la aplicación.



CUSTOMER AGREEMENT

Es posible acceder a los documentos del proyecto relativos a Customer Agreement, SLA y Análisis de Capacidad a través del siguiente enlace: https://github.com/Noctua-sapientia/docs.git

- Customer Agreement: https://github.com/Noctua-sapientia/docs/blob/main/Customer%20Agreement/Customer%20Agreement%20v4.pdf
- SLA: https://github.com/Noctua-sapientia/docs/blob/main/SLA/SLA%20v6.pdf
- Análisis de la capacidad: https://github.com/Noctua-sapientia/docs/blob/main/An%C3%A1lisis%20de%20la%20capacidad/Analisis%20de%20la%20capacidad%20v6.pdf

DESCRIPCIÓN DEL API REST DEL MICROSERVICIO PEDIDOS

MODELO DE DATOS

El microservicio de pedidos utiliza un modelo de datos detallado para representar la información de cada pedido. Antes de pasar a describir las acciones que permite realizar la API REST, describiremos los campos que se contemplan para cada pedido:

- **orderId (Número):** Identificador único de la orden. No es un campo obligatorio, ya que se genera automáticamente
- **userId (Número)**: Identificador del usuario que realiza el pedido.
- **sellerId (Número):** Identificador del vendedor al cuál se realiza el pedido.
- **books (Array de Objetos)**: Representa la lista de libros comprados por el cliente en el pedido. Cada libro en la lista contiene los siguientes campos:
    - bookId (Número): Identificador del libro.
    - units (Número): Cantidad de unidades del libro en el pedido. Campo obligatorio.
    - price (Número): Precio unitario del libro. Campo obligatorio.
- **status (Cadena):** Estado actual del pedido (por ejemplo, 'En preparación', 'Enviado', 'Entregado' o 'Cancelado').
- **deliveryAddress (Cadena):** Dirección de entrega del pedido al cliente.
- **maxDeliveryDate (Fecha):** Fecha máxima antes de la cual el pedido debe ser entregado al comprador.
- **creationDatetime (Fecha):** Fecha y hora en la que se crea o realiza el pedido por parte del cliente.
- **updateDatetime (Fecha):** Fecha y hora de la última vez que fue actualizado el pedido.
- **shippingCost (Número):** Costo de envío asociado al pedido.

Para la base de datos se ha utilizado MongoDB, facilitando su gestión a través de Mongo Atlas y facilitando u gestión en el backend gracias a Mongoose.

BACKEND

ACCIONES (VERBOS Y ENDPOINTS)

ESPECIFICACIÓN DE API		
Verbo	**Endpoint**	**Descripción**
GET	GET /api/v1/orders?userId=:userId&sellerId=:sellerId&status=:status&bookId=:bookId&minPayment=:minPayment&maxPayment=:maxPayment&sort=:sortCriteria	Permite acceder a todos los pedidos como recurso, así como filtrarlos según varios criterios (incluyendo filtrado por usuario, vendedor, estado del pedido, libro comprado, pago mínimo y máximo e incluso criterio de ordenación).
	GET /api/v1/orders/:orderId	Permite acceder a un pedido específico como recurso.
	GET /api/v1/orders/price/:orderId	Obtiene el pago total de un pedido específico. El precio total incluye el precio de los libros y el costo de envío.
POST	POST /api/v1/orders	Permite guardar un pedido (cuyos campos se proporcionen en el cuerpo de la petición en formato JSON).
PUT	PUT /api/v1/orders/:orderId	Permite actualizar un pedido concreto (pasando de forma oportuna los campos en el cuerpo de la petición en formato JSON).
	PUT /api/v1/orders/books/:bookId/sellers/:sellerId/cancelledRemove	Elimina un libro específico de todos los pedidos en preparación, o cancela el pedido si solo contiene ese libro.
	PUT /api/v1/orders/sellers/:sellerId/cancelled	Cancela todos los pedidos en preparación para un vendedor específico.
	PUT /api/v1/orders/users/:userId/cancelled	Cancela todos los pedidos en preparación para un usuario específico.
	PUT /api/v1/orders/user/:userId/deliveryAddress	Actualiza la dirección de entrega para todos los pedidos en preparación de un usuario específico.
DELETE	DELETE /api/v1/orders/:orderId	Permite eliminar un pedido del historial de pedidos si este se encuentra en un estado terminal (pedido cancelado o pedido entregado).

Dichos métodos API gestionan las peticiones devolviendo respuestas en un formato adecuado y con códigos de error oportunos (según el origen del fallo).

TESTS DE COMPONENTES Y DE INTEGRACIÓN

- Se evaluó el funcionamiento de los métodos expuestos por la API mediante exhaustivos tests de componentes que contemplaban casos de prueba tanto positivos como negativos (considerando distintos escenarios negativos como errores internos, errores de petición, campos faltantes...).
- También se realizaron tests de integración, con el objetivo de comprobar la correcta integración con el sistema de base de datos.

```
Test Suites: 1 passed, 1 total
Tests:       32 passed, 32 total
Snapshots:   0 total
Time:        9.118 s
Ran all test suites.
```

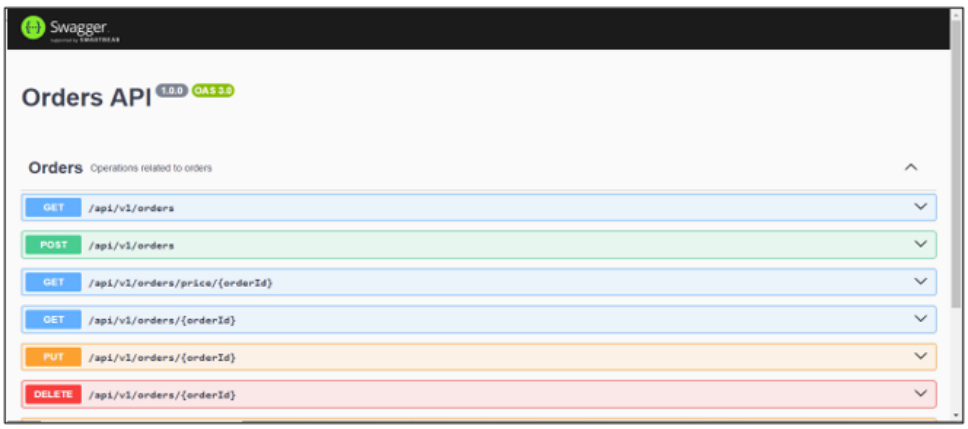
Tests de componentes.

```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        6.679 s, estimated 34 s
Ran all test suites.
```

Tests de integración.

DOCUMENTACIÓN: SWAGGER

La API REST incluye documentación, desarrollada utilizando Swagger y siguiendo la especificación OpenApi 3.0.0. Esta documentación está accesible para los consumidores de la API a través del endpoint **/api/v1/apidocs/orders**.



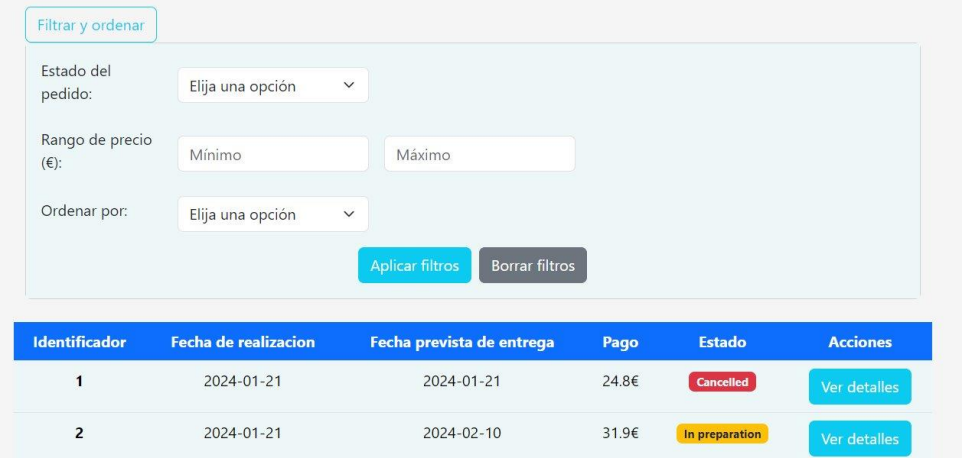
DOCKERIZACIÓN DE LA APLICACIÓN

La aplicación se encuentra adecuadamente Dockerizada y lista para su despliegue en cualquier entorno, garantizando consistencia y eficiencia en su ejecución.

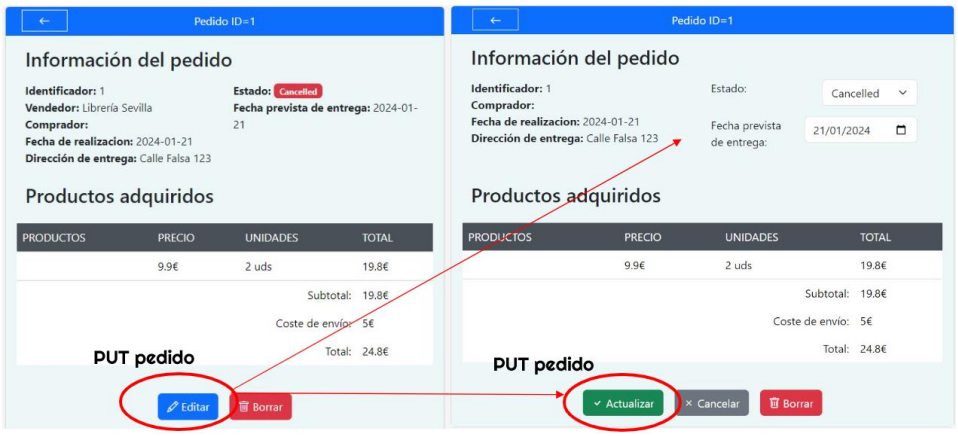
FRONTEND

En la interfaz de usuario del microservicio de pedidos, se destacan tres componentes principales: el carrito de compras y el registro de pedidos.

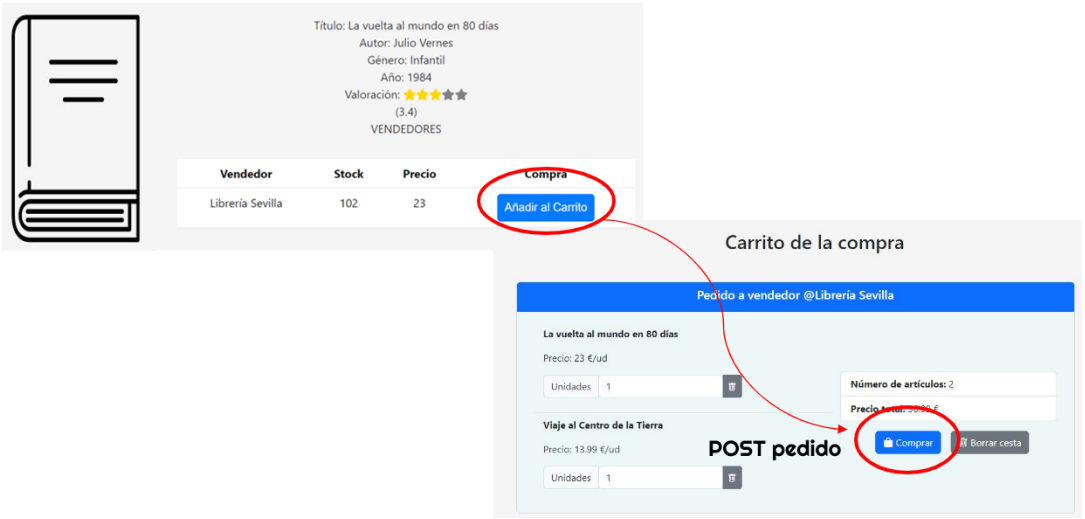
- El registro de pedidos muestra a los vendedores los pedidos recibidos y a los compradores los que han realizado. Cada pedido incluye detalles como su identificador, fechas de inicio y entrega estimada, precio y estado actual, que puede variar entre en preparación, enviado, entregado o cancelado. Esta sección ofrece opciones para filtrar y organizar la lista de pedidos.



- Además, hay una vista detallada de cada pedido accesible desde el registro. Esta vista proporciona información adicional como los nombres de comprador y vendedor, dirección de entrega y un desglose completo de los libros pedidos. En esta sección, los vendedores pueden actualizar el estado o la fecha de entrega del pedido según las necesidades del negocio. También se permite eliminar pedidos en estado finalizado (entregado o cancelado), si el usuario decide no mantenerlo en su registro de pedidos.



- El carrito de compras, reservado exclusivamente para los usuarios compradores, permite añadir libros de diferentes vendedores, ajustar cantidades, eliminar ítems y finalizar compras. Este carrito opera mediante el uso de localStorage, con una lógica que facilita la adición de libros a carritos existentes o la creación de nuevos, dependiendo de la presencia de carritos activos entre comprador y vendedor.



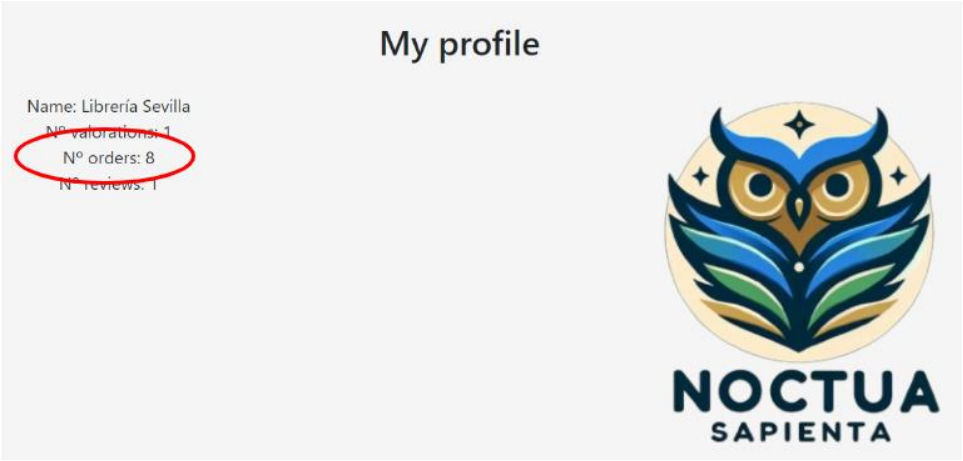
- Además de estas pantallas propias del frontend del servicio, se han definido componentes React en las vistas de otros microservicios, consiguiendo así un frontend totalmente integrado y homogéneo. En la pestaña de libros se integra el botón para añadir al carrito de la compra u libro de un vendedor y un mensaje para indicarle al comprador si ese libro ya lo ha comprado anteriormente y si es vendedor si lo ha vendido anteriormente y en cuántos pedidos. En la cuenta del usuario, en caso de ser vendedor sale cuantos pedidos se le han hecho.


Ya has comprado este libro anteriormente.

En libros....


Número de pedidos de este libro: 1


Título: La vuelta al mundo en 80 días
Autor: Julio Vernes
Género: Infantil
Año: 1984
Valoración:
(3.4)
VENDEDORES
Vendedor
Stock
Precio
Compra
Librería Sevilla
102
23
Añadir al Carrito


My profile
Name: Librería Sevilla
Nº orders: 8
NOCTUA
SAPIENTA

REQUISITOS Y DESCRIPCIÓN DE CUMPLIMIENTO

MICROSERVICIO BÁSICO

Microservicio básico	
Requisito	**Descripción**
API REST	Como se ha explicado en la sección anterior, el backend implementa todos los métodos necesarios, incluyendo endpoints adicionales necesarios para facilitar la realización de algunas acciones a otros microservicios. Los distintos endpoints gestionan los errores devolviendo códigos de errores apropiados, indicando si la petición se ha desarrollado correctamente o, si no se ha realizado correctamente, si el error es debido al usuario o al servidor. El código que evidencia el desarrollo de este requisito puede encontrarse en el repositorio del backend (orders-service). Especialmente en la carpeta routes/orders.js podrá encontrar la implementación de las acciones. Si se desea consultar con mayor profundidad se puede acceder a dicho archivo.
Autenticación	Se realiza el proceso de autenticación mediante JWT. Una vez el usuario haya realizado el login en la aplicación, se le facilita un token. Utilizando este token el usuario podrá interactuar con los métodos del backend. Pese a que el token se reciba al hacer login la validación se realiza en los microservicios. Para más detalles, consultar en el repositorio orders-service los archivos AuthContext.js del frontend y routes/verificarToken.js del backend (así como la verificación del token en los endpoints en routes/orders.js) y AuthContext.js del frontend.
Frontend	Toda la aplicación presenta un frontend común, localizado en el repositorio frontend. Aquí se encuentran todas las vistas que presenta la aplicación, entre ellas las del microservicio (historial de pedidos, detalles de pedido y carrito). También se encuentran los componentes integrados en otros micros (en la pantalla de cuenta de usuario el número de pedidos realizados al vendedor, en libros el botón de añadir al carrito, el mensaje de haber comprado el libro...). La subcarpeta src/api/OrersApi.js y src/components/Orders del repositorio frontend contiene la mayor parte de la lógica y componentes correspondientes a la vista del historial de pedidos, detalles de pedidos y carrito. Mientras que los componentes integrados con libros y usuarios pueden encontrarse en los componentes correspondientes de sus carpetas-
Dirección bien versionada	Toda la aplicación está bien versionada y es accesible mediante la dirección /api/v1/orders.
Documentación	Las funciones de la API se encuentran documentadas haciendo uso de swagger, que permite acceder a la misma en el endpoint api/v1/apidocs/orders, indicando entradas y posibles salidas.

MongoDB	Toda la aplicación presenta persistencia de datos haciendo uso de la base de datos no SQL MongoDB y su servicio Atlas. Para más detalles sobre la conexión consultar el archivo db.js.
Mongoose	Haciendo uso de mongoose validamos nuestros datos para que tengan los campos y formatos correctos antes de guardarlos en la base de datos. Puede consultarse en el archivo models/order.js del repositorio orders-service.
Gestión del código fuente e integración continua	Haciendo uso de GitHub, todos los microservicios y el frontend se encuentran repositados. Para ello, se ha seguido GitHub Flow, haciendo uso de ramas para poder trabajar de forma paralela en distintas funcionalidades del microservicio (features). Mediante GitHub Actions lo test definidos con Jest se ejecutan de forma automática tras cada push.
Docker	El proyecto consta de un Dockerfile (en el repositorio orders-service encontramos el fichero Dockerfile) mediante el cual se puede construir un contenedor funcional que contenga todas las funcionalidades del microservicio.
Test	Se han realizado pruebas de componentes mediante Jest para todas las funciones de la API, en los que se han probado tanto casos positivos como de los posibles errores de cada función. Asimismo, se han realizado pruebas de integración con la base de datos. Todos estos test son ejecutados de forma automática mediante GitHub Actions. Podemos encontrar los tests en la carpeta tests del repositorio orders-service.

MICROSERVICIO AVANZADO

Microservicio avanzado	
Requisito	**Descripción**
Frontend con navegación	Todo el frontend es accesible mediante rutas y navegación (como la navegación del historial a los detalles de pedidos, entre otros casos que pueden apreciarse en el frontend). Para más detalle se puede consultar el repositorio de frontend.
API REST documentada con swagger	La API REST incluye documentación, desarrollada utilizando Swagger y siguiendo la especificación OpenApi 3.0.0. Esta documentación está accesible para los consumidores de la API a través del enfpoint /api/v1/apidocs/orders.

Autenticación con JWT	Como comentamos en la sección anterior, la autenticación se logra mediante JasonWebToken (JWT). Si se desea más información, se puede encontrar toda la lógica en los archivos routes/verificarToken.js del repositorio orders-service (backend) y src/components/AuthContext del frontend.
API externa	Se ha hecho uso de la API de mailgun, que permite enviar correos desde nuestro backend. Para ello, cada vez que se crea un pedido nuevo, accedemos al correo del comprador y se le envía un correo con información respectiva a su pedido.
Almacenamiento externo	Aunque no se ha logrado una integración por problemas de incompatibilidad de versiones, se ha realizado una implementación mediante la API de DropBox para poder almacenar facturas de los usuarios en la nube. Está característica se ha implementado en una rama separada del resto.

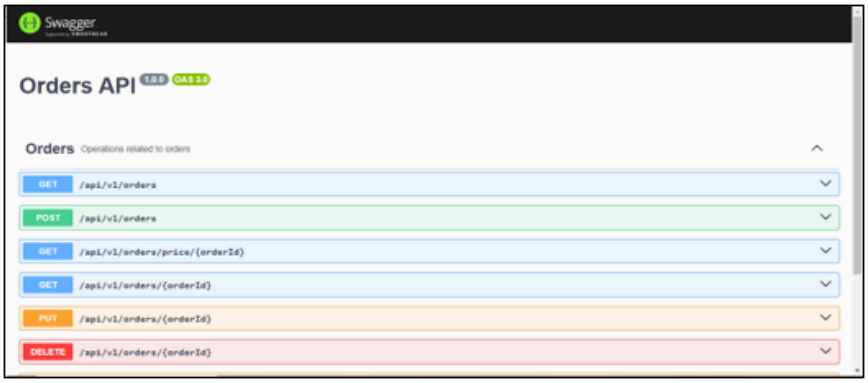
APLICACIÓN AVANZADA

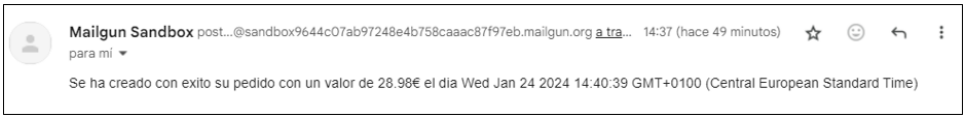
Aplicación avanzada	
Requisito	**Descripción**
Frontend común	El microservicio de pedidos está integrado en un frontend común y presenta componentes incluso fuera de sus vistas principales. Por ejemplo, en la sección de libros se muestran dos tipos de mensajes: si eres comprador te avisa de si ya has comprado ese libro anteriormente y si eres vendedor te indica el número de pedidos que te han realizado de dicho libro. También en la pestaña de libros tenemos el botón de añadir al carrito que integra el frontend de libros con el de pedidos. Por otro lado, en la cuenta de vendedor podemos consultar el número de pedidos que nos han realizado.
Pruebas de aceptación de usuario (UAT)	Se han probado las distintas funcionales ofrecidas por la aplicación una vez integradas con el frontend desde el punto de vista de un usuario real. Esto ha permitido detectar defectos y corregirlos en la versión final. Se puede encontrar este documento en el repositorio de docs.
Mecanismo de autenticación homogéneo	Todo el servicio de autenticación es homogéneo para todos los microservicios, permitiendo acceder a todos ellos iniciando únicamente una vez sesión. Esto ha sido posible mediante el uso de JWT, que es verificado por las funciones del backend.

CUESTIONES ADICIONALES (EXTRA)

- A nivel de microservicio se han realizado los siguientes extras:
  - Backend:
    - Autenticación JWT: Como comentaremos en el apartado extras de aplicación, se generará un token de JWT que será verificado en todas las funciones de la API, impidiendo su uso si este no es válido.

- Documentación swagger: Todas las funciones de la API se encuentran documentadas mediante swagger siendo posibles acceder a ellas en el endpoint /api/v1/apidocs/order
- API externa: Se ha hecho uso de la API mailgun que permite el envío de correos electrónicos. Se ha configurado de tal forma que al crear un nuevo pedido, se envíe un correo al comprador indicando que se ha realizado con éxito e informaicón sobre el mismo
- Almacenamiento externo: Aunque no se ha llegado a integrar con la API por problemas de versiones, se ha realizado una implementación mediante la API de DropBox para poder generar facturas sobre cada pedido y almacenarlas en la nube.





- Frotend:
  - Frontend con mecanismo de navegación: El frontend de todos los micros se encuentra totalmente integrado y permite la navegación entre ellos, no estando todo en la misma pantalla.
  - Componentes integrados: Se han integrado distintos componentes en vistas del frontent de varios microservicios, como por ejemplo un aviso de si ya has comprado un libro para el caso de compradores y un recuento de cuántas

veces le han comprado un libro para el caso de vendedores, un recuento de pedidos en la pantalla de inicio de vendedores y un botón que permite añadir pedidos al carrito en la sección de libros.

- A nivel de aplicación global se han realizado los siguientes extras:
  - Autenticación homogénea con JWT. Mediante JWT, se genera un token al inicio de sesión, que es usado por las funciones de los distintos backends para validar las peticiones.
  - Frontend común integrado mediante setupProxy. La aplicación puede ser desplegada en local y este proxy es el encargado de redireccionar las peticiones recibidas desde el frontend al puerto correspondiente de cada microservicio.
  - Pruebas de aceptación de usuario, donde hemos recopilados defectos encontrados en el front y sus correcciones. El documento final puede encontrarse en https://github.com/Noctua-sapientia/docs/blob/main/Pruebas%20de%20Aceptaci%C3%B3n%20de%20Usuario/Pruebas%20de%20Aceptaci%C3%B3n%20de%20Usuario%20(UAT)%20Latest%20Version.pdf

ANÁLISIS DE ESFUERZOS

La distribución de las tareas se hizo de forma equitativa, de modo que se planificó y cumplió una distribución global del trabajo 50 %-50%.

Otro de los objetivos en la distribución del trabajo era que todos los integrantes del equipo de trabajo del microservicio participaran en todos los bloques de trabajo (es decir, hemos evitado que alguno de los miembros del equipo no tenga ninguna tarea asignada dentro del bloque de testing, por ejemplo).

El objetivo es que los miembros del equipo adquirieran un conocimiento completo del trabajo, para que aprendieran de todos los aspectos y se favoreciera la comunicación y la agilidad del trabajo.

ANÁLISIS DE ESFUERZOS EN EL DESARROLLO DEL BACKEND

ANÁLISIS DE ESFUERZOS EN EL DESARROLLO DEL BACKEND			
Tarea	**Pablo Reina**	**Lourdes Linares**	**Comentarios**
Implementación de endpoints	5h	8h	Se contempla la implementación de las operaciones fundamentales (GET, POST, PUT, DELETE). Además, se planea el desarrollo de endpoints más especializados, diseñados para ser utilizados por otros microservicios, facilitando así la realización de tareas con nuestra base de datos. En esta sección, también abordamos las llamadas "colaterales" efectuadas a otros microservicios, que se activan al ejecutar ciertas acciones.
Persistencia del sistema con mongoDB	2h	2h	Desarrollo de una colección de pedidos dentro de la plataforma Mongo Atlas, estableciendo una conexión efectiva de esta base de datos con el backend. Se emplea Mongoose para optimizar y gestionar esta integración.
Documenntación del API con swagger	6h	3h	Esta sección consistió en la documentación del API desarrollada utilizando swagger.
Testing	10h	10h	Engloba la realización de tests de componentes y tests de integración con la BD.
Autenticación	3h	3h	En esta sección se investigó e implementó la autenticación para la realización de operaciones contra el backend utilizando JWT.
Despliegue, control de versiones e integración continua	5h	5h	Se contemplan actividades como la Dockerización de la aplicación y su despliegue en Okteto, que, aunque inicialmente se implementó de manera funcional, tuvo que ser descartado debido a cambios en la estructura de precios. Asimismo, se incluye la integración de pruebas automáticas en el repositorio de GitHub a través de GitHub Actions. También se incluye en esta

			sección la formación del equipo de cara al trabajo con GitFlow.
Servicios externos (APIs externas, almacenamiento…) y otros extras	8 h	5 h	Este bloque de tareas incluye la exploración de diferentes servicios externos que podrían integrarse en el sistema. Para el almacenamiento, se optó por Dropbox para la generación y almacenamiento de facturas, aunque se llegó a desarrollar de manera funcional no llegó a integrar debido a incompatibilidades con otras versiones de Node.js. Sin embargo, se logró con éxito la implementación del servicio de envío de correos electrónicos utilizando MailGun.
TOTAL (BACKEND)	39 h	36 h	Se han invertido aproximadamente unas 80 horas totales en el desarrollo del backend del proyecto.

ANÁLISIS DE ESFUERZOS EN EL DESARROLLO DEL FRONTEND

ANÁLISIS DE ESFUERZOS EN EL DESARROLLO DEL FRONTEND (EN LO RELATIVO AL SERVICIO PEDIDOS)			
Tarea	**Pablo Reina**	**Lourdes Linares**	**Comentarios**
Diseño y estática HTML+CSS	2 h	5 h	Se incluye la tarea inicial de desarrollar cada componente del frontend de forma estática en código html y css, para su posterior traducción a react.
División en componentes. Datos estáticos.	6 h	7h	En este aparatado se realizó la transformación del código html básico a código react estático, dividiendo cada una de las vistas en los componentes necesarios.
Integración frontend con API backend	20 h	25 h	Se realizó la integración del frontend con los datos proporcionados por las distintas APIs permitiendo tener pestañas dinámicas en función de los datos.
Integración con otros servicios	9 h	6 h	Se integraron distintos componentes del microservicio de orders en vistas de otros servicios, como por ejemplo indicar si un comprador ha comprado previamente un libro o no o el número de pedidos realizado a un comprador de un libro concreto
Autenticación	7 h	6 h	En esta sección se logró el proceso de autenticación mediante el uso de JsonWebToken, que nos permite realizar llamadas a las funciones de la API únicamente si estás autenticado con el usuario correcto.
Cuestiones adicionales: SessionStorage, control de opciones, navegación	5 h	3 h	Este apartado recopila todas las tareas adicionales no contempladas en puntos anteriores, como la integración de todos los fronts en uno mismo, implementación de la navegación, el uso de SessionStorage etc.
TOTAL (FRONTEND)	49h	52h	Se han invertido aproximadamente unas 100 horas totales en el desarrollo del backend del proyecto.

ANÁLISIS DE ESFUERZOS EN DOCUMENTOS Y FORMACIÓN

ANÁLISIS DE ESFUERZOS: FORMACIÓN Y DOCUMENTACIÓN		
Tarea	**Pablo Reina Lourdes Linares**	**Comentarios**
Formación del equipo en la materia	15 h	Este bloque incluye tareas de formación que han requerido un tiempo adicional considerable como es la investigación de extras de microservicio o aplicación y el aprendizaje para poder aplicar los conocimientos de la asignatura de forma práctica al proyecto.
Documentación del proyecto. Análisis de capacidad y Customer Agreement.	25 h	Incluye la elaboración de las memorias y los documentos o informes explicando el desarrollo del trabajo, así como la participación activa en el desarrollo de documentos como el Customer Agreement, el SLA, el análisis de la capacidad etc
TOTAL (FORMACIÓN Y DOCUMENTACIÓN)	40 h	Se han invertido aproximadamente unas 40 horas adicionalmente al desarrollo para la documentación del proyecto y la formación.

En conjunto el equipo de trabajo ha invertido 216 horas en el proyecto, siendo la distribución de trabajo equitativa entre ambos miembros (108 horas)