SDI – Sistemas Distribuidos e Internet

ENUNCIADO PRÁCTICA 1 – SPRING 2022/2023

INFORME

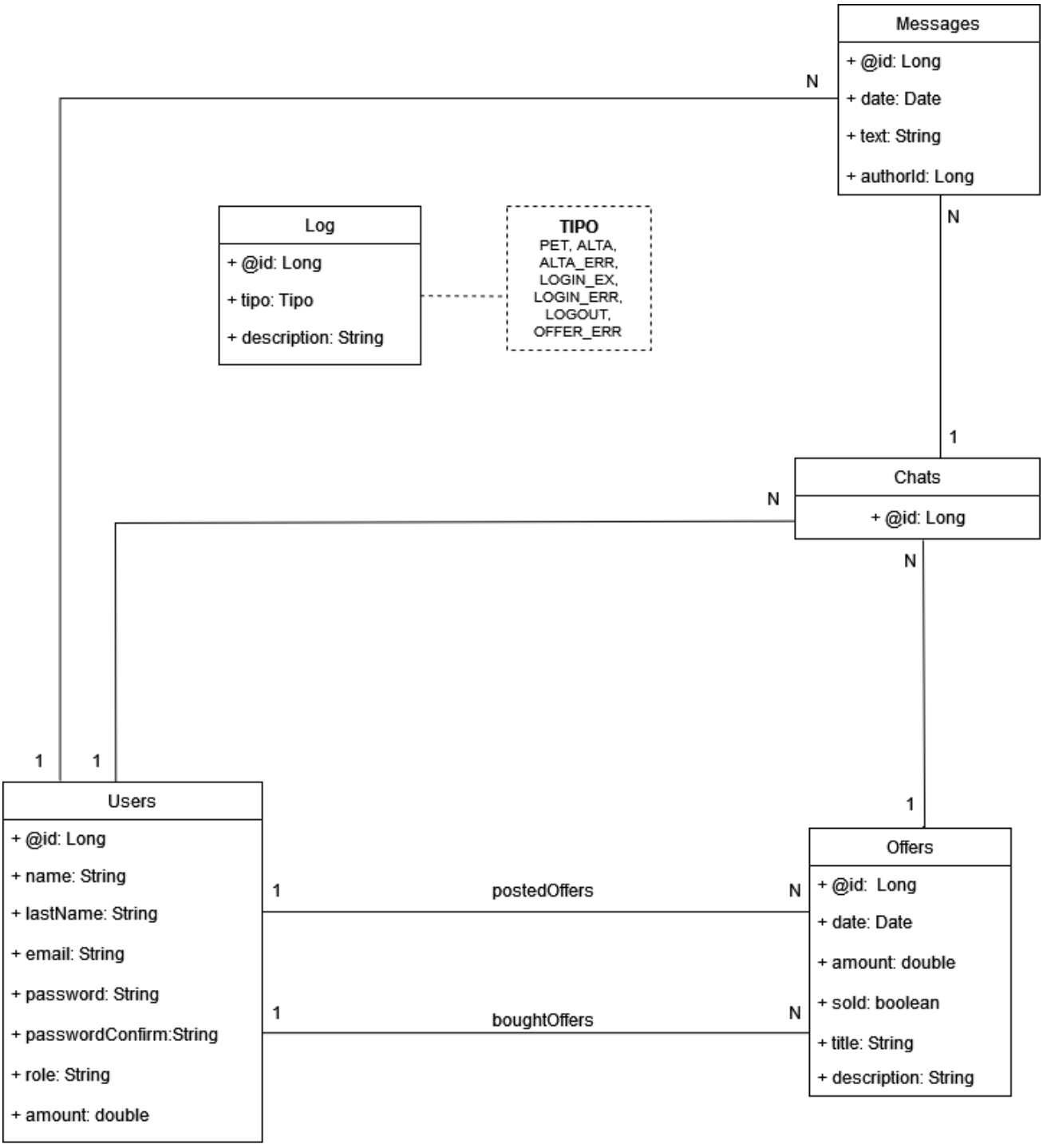
Nombre/Apellidos, ID-GIT#1	Eduardo Blanco Bielsa, 2223-503, uo285176@uniovi.es
Nombre/Apellidos, ID-GIT#2	Aarón Orozco Fernández, 2223-508, uo281997@uniovi.es
Nombre/Apellidos, ID-GIT#3	Chen Xin Pan Wang, 2223-510, uo276967@uniovi.es
Nombre/Apellidos, ID-GIT#4	Diego Villa García 2223-511, uo277188@uniovi.es
Nombre/Apellidos, ID-GIT#5	Santiago López Laso, 2223-506, uo277369@uniovi.es
Nombre/Apellidos, ID-GIT#6	
Cód. ID EQUIPO	53
Repositorio Github	https://github.com/gitblanc/sdi2223-entrega1-53

Índice

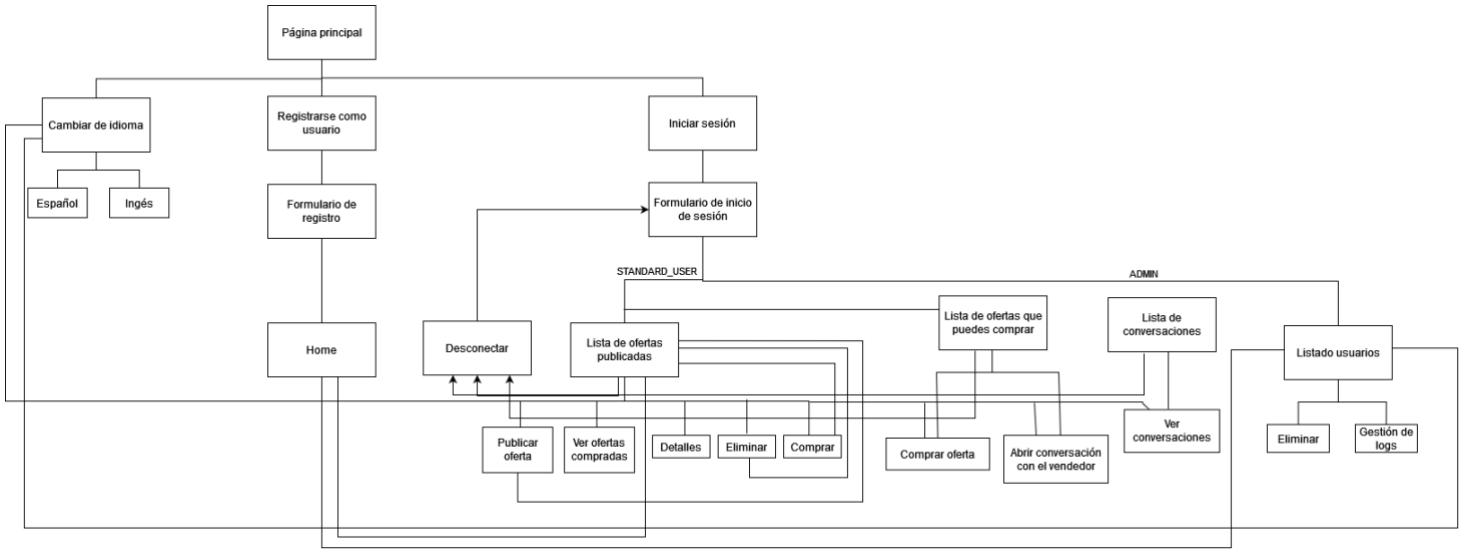
Introducción

MyWallapop es una aplicación basada en el modelo Vista-Controlador desarrollada en Spring creada para la asignatura de Sistemas Distribuidos e Internet. En esta aplicación se plasma parte de la funcionalidad de la tan conocida aplicación Wallapop, pero en una versión más reducida y funcional. En ella, como usuario, podrás comprar ofertas, ver las existentes, buscar las que te interesen, publicar nuevas ofertas y entablar conversaciones con otros usuarios.

Modelo de datos



Mapa de navegación



Aspectos técnicos y de diseño relevantes

Se va a utilizar el Modelo Vista Controlador, pues es la arquitectura vista en clase. A continuación, se detallan ciertos aspectos técnicos y de diseño que consideramos importantes clasificados en función a la capa de la arquitectura elegida.

Controladores:

- **UserController**: este controlador se encarga de manejar todas las peticiones relacionadas con las posibles acciones de un usuario. En este controlador se usan los servicios UserService, SecurityService, RolesService y LogsService y el validador SignUpFormValidator, para el correcto tratamiento de las diversas peticiones que puede llevar a cabo un usuario.
- **OffersController:** es el encargado de gestionar tanto las peticiones relativas a las ofertas propias del usuario (listar, detalles, crear, borrar) como las que se refieren a las demás ofertas (comprar, listar). Para ello hace uso de los servicios OffersService, UsersService, LogsService y el validador AddOfferValidator. Cuando se realiza una petición comprar, puede ocurrir algún error y hay que indicarlo al usuario. En este caso el mensaje de error se pasa como parámetro a la URL de listar todas las ofertas, y la función que responde a esa URL mete en el modelo de Offer/list.html el mensaje. Después la vista muestra el error al usuario.
- **ChatsController:** su función es manejar las peticiones sobre los chats, es decir, listar los chats del usuario, crear uno nuevo y ver un chat en concreto. Para identificarlos mediante la URL se utiliza el ID la oferta sobre la cual se está hablando y el email del otro usuario.

Vistas:

- **LogList.html**: en esta vista se ha implementado un filtrado por búsqueda de cadena, es decir, el usuario introduce un tipo de los que se le sugiere previamente al introducir el filtro deseado (da igual que introduzca mayúsculas o minúsculas). En el caso de que busque un tipo de log que no exista, no devolverá ningún resultado. Este filtro mostrará los diversos logs de más actual a más antiguo.

- **UserList.html**: en esta vista se ha implementado un borrado de usuarios mediante el uso de checkboxes, de forma que el usuario administrador, además de ver a todos los usuarios de la app, podrá eliminar múltiples usuarios de forma inmediata, sin tener que ir uno a uno.
- **Offer/listPosted.html:** es la vista encargada de mostrar la lista de las ofertas propias del usuario. Sobre cada una se muestran los botones de eliminar y ver detalles.
- **Offer/list.html:** a diferencia de la anterior, esta vista muestra todas las ofertas disponibles en la aplicación (a excepción de las del propio usuario). Desde aquí se pueden comprar y crear conversaciones con el vendedor.
- **Chats/chatList.html:** muestra el listado de los chats del usuario que ya han sido creados. Desde aquí se puede acceder a cada uno mediante un botón.
- **Chat/chat.html:** contiene un chat concreto. Aparecen los mensajes organizados en forma de tabla, cada uno con su remitente, fecha y contenido.

Servicios:

- **LogsService**: se ha utilizado un Logger, que es un objeto que se encarga de ir almacenando toda la información de peticiones, logins y errores y mostrarlos por pantalla al tiempo que suceden. Además, todos estos logs se persistirán en la base de datos por medio del repositorio LogsRepository. También se añadieron dos nuevos tipos de logs para manejar ciertos errores: ALTA_ERR (para los errores surgidos durante un registro) y OFFER_ERR (para los errores surgidos con temas relacionados con ofertas).
- **RolesService**: servicio donde se almacenan los dos tipos de roles de la aplicación: ROLE_STANDARD (para cualquier usuario menos el administrador) y ROLE_ADMIN (para el administrador del sistema)
- **ChatsService**: contiene operaciones relativos a mensajes y chats (añadir mensaje, recuperar un chat o un listado de chats de un usuario, crear un chat…).
- **OffersService**: se encarga de las operaciones a realizar sobre ofertas (añadir, borrar, recuperar listados, comprar…).

Repositorios:

- **LogsRepository**: se añadieron ciertos métodos para el manejo de los logs y los filtrados para ordenarlos por fecha y por tipo.
- **ChatsRepository**: se utilizaron dos métodos nuevos. Uno para recuperar todos los chats de un usuario y otro para recuperar un chat concreto entre dos usuarios.
- **MessagesRepository**: contiene un método para encontrar todos los mensajes de un usuario.
- **OffersRepository**: contiene métodos para actualizar una oferta al comprarla, recuperar ofertas por usuario y búsqueda por título, entre otros.

Validadores:

- **SignUpFormValidator**: validador para el formulario de registro en el que controlamos si el email ya existe en el sistema y si ambas contraseñas introducidas son idénticas.
- **AddOfferValidator**: comprueba que los datos que provienen del formulario de añadir oferta son válidos (los campos no son vacíos, el precio no es negativo y el título está entre 5 y 24 caracteres).

Información necesaria para el despliegue y ejecución

Versiones del software usado:

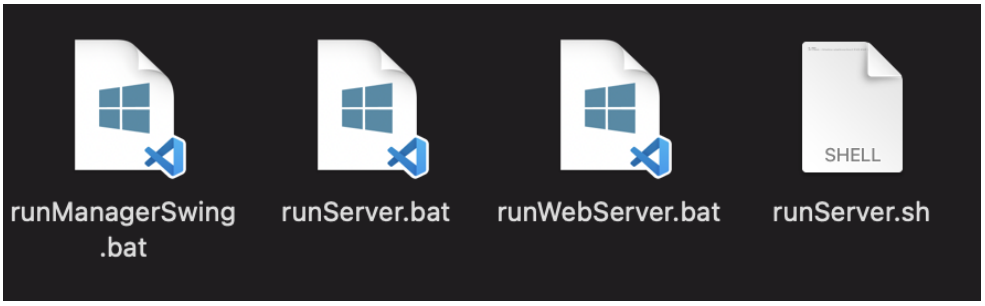
- **Java**: 17 amazon coretto (≈jdk17)
- **Hsqldb**: 4.7.1
- **Spring**: 2.7.8
- **Nota**: por motivos de seguridad se ha utilizado una versión más reciente de SpringSecurity, concretamente la versión 5 en lugar de la 4 (la utilizada en clase)

Instrucciones de despliegue de la aplicación:

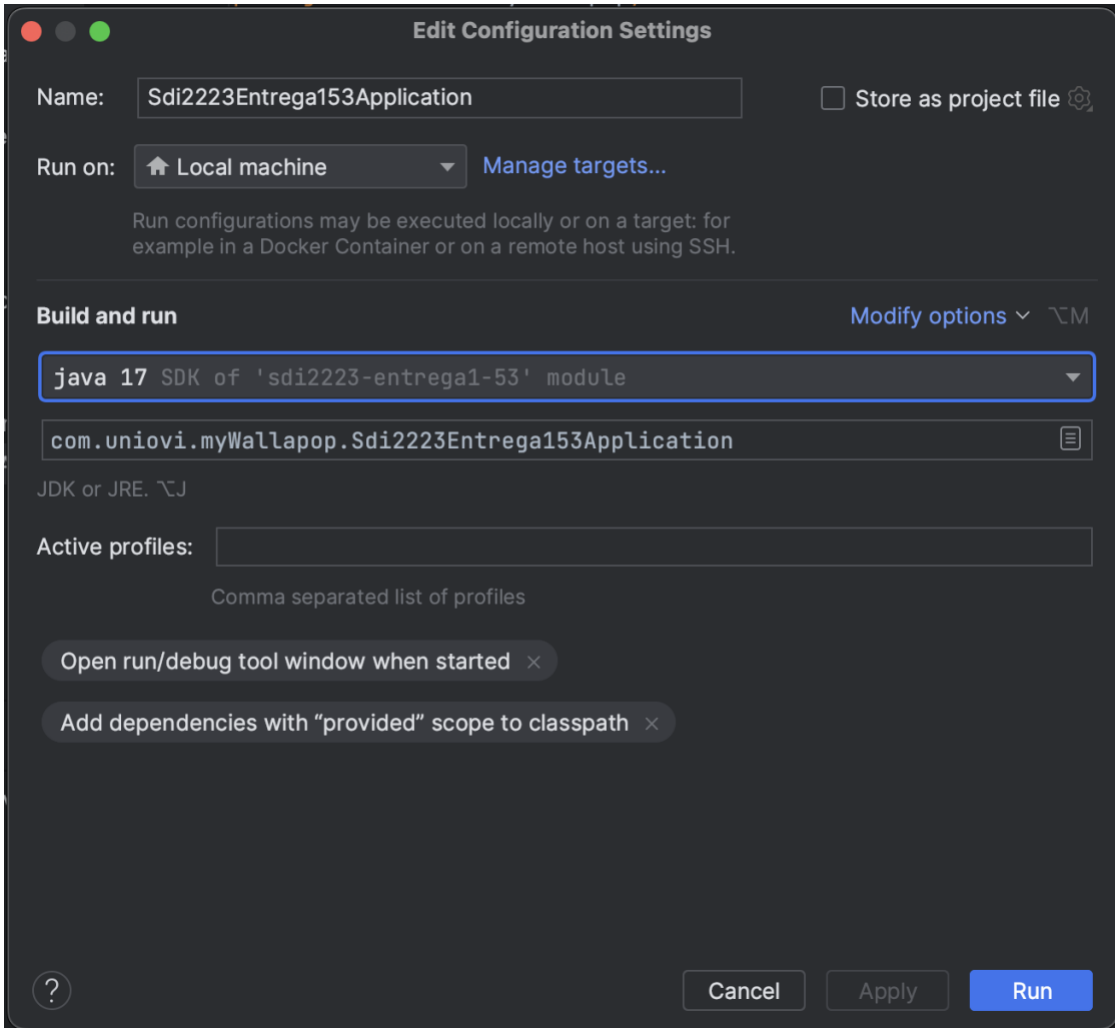
- Tener instalado una versión razonablemente actual de IntelliJ Ultimate o cualquier otro editor que permita ejecutar Java

- Instalar hsqldb 4.7.1

- Ejecutar el archivo runServer.bat o runServer.sh dentro de la carpeta bin en función de nuestro SO

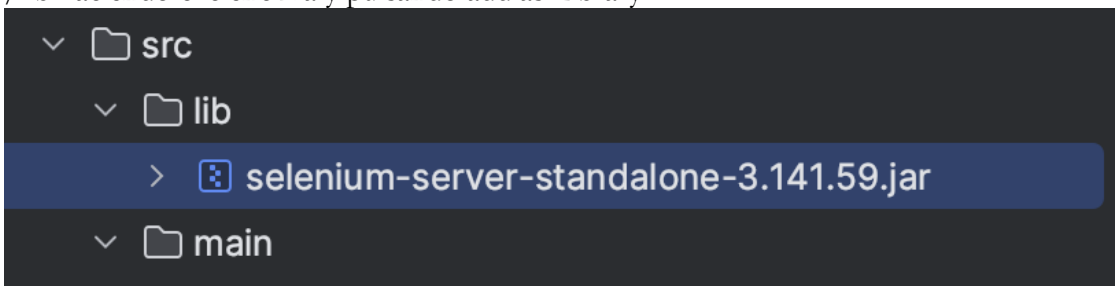


- Ejecutar el main de la aplicación de Spring (configurar Java 17 si lo solicita o no se autodetecta en Edit configurations)

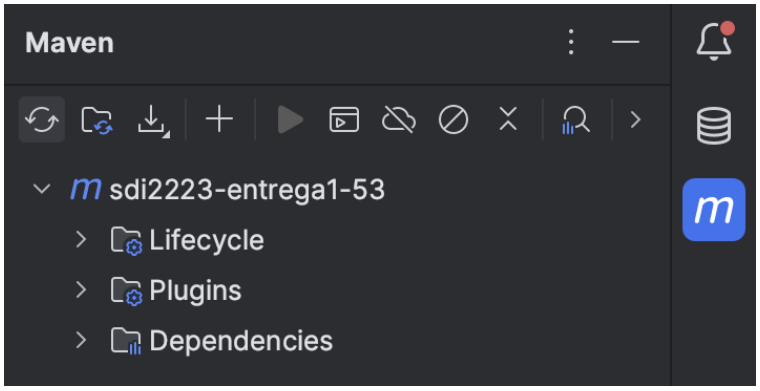


Instrucciones de despliegue de las pruebas:

- Dentro del proyecto, hay que añadir manualmente la librería de Selenium que se haya en la carpeta /lib haciendo clic encima y pulsando add as Library.

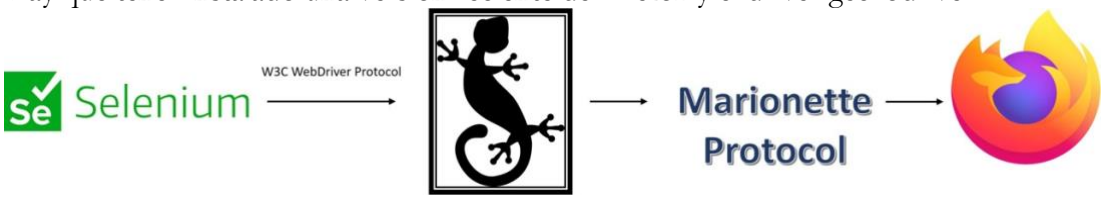


- Una vez hecho esto hacer un Reload con Maven



- Ejecutar la aplicación principal

- Hay que tener instalado una versión reciente de Firefox y el driver geckodriver.



- En la clase de tests Sdi2223Entrega153ApplicationTests retocar el path de Firefox y geckodriver a la localización del tester en cuestión.

```
class Sdi2223Entrega153ApplicationTests {

    1 usage
    static String PathFirefox = "C:\\Program Files\\Mozilla Firefox\\firefox.exe";
    //static String Geckodriver = "C:\\Users\\uo277369\\Desktop\\PL-SDI-Sesión5-material\\geckodriver-v0
    1 usage
    static String Geckodriver = "C:\\Users\\mines\\Desktop\\wallapop\\geckodriver-v0.30.0-win64.exe";
    //static String Geckodriver = "C:\\Users\\aaron\\Desktop\\UNI\\tercero\\SEGUNDO\\SDI\\Practica\\SDI-
    //static String Geckodriver = "C:\\Dev\\tools\\selenium\\geckodriver-v0.30.0-win64.exe";
    //static String PathFirefox = "/Applications/Firefox.app/Contents/MacOS/firefox-bin";
//static String Geckodriver = "/Users/USUARIO/selenium/geckodriver-v0.30.0-macos";
```

- Ejecutar la clase de tests dentro de /test/java/com.uniovi.myWallapop

Conclusión

Miembro	Aportaciones	Ventajas	Desventajas
Aarón Orozco Fernández	La principal función que he realizado ha sido todo lo relacionado con los usuarios y los roles, desde el UserController, el login, registro de sesión y las vistas correspondientes. También he ayudado a Eduardo a realizar su parte de la seguridad. Por último, junto con Diego y Eduardo hemos creado la base de datos.	**Nivel técnico:** He conseguido realizar sin mucho problema algunas cosas que no habíamos realizado previamente en clase, como lo es por ejemplo el borrado mediante checkbox de usuarios. **Nivel trabajo de grupo:** Creo que hemos trabajado bastante bien en equipo, sobretodo en mi caso he trabajado bastante con Eduardo y ha habido una buena comunicación y ganas de trabajar.	**Nivel técnico:** Tuve ciertas dificultades con aspectos relacionados con la seguridad pero que me correspondían a mí, cosas como problemas con los roles de usuario o con el acceso indebido a vistas. **Nivel trabajo en grupo:** La única pega que podría sacar es el desorden que hubo al principio del proyecto, pero después de realizar algunas reuniones todo se encauzó con éxito.
Eduardo Blanco Bielsa	Mi principal función fue encargarme de la seguridad de la aplicación (roles, vistas y logs). Además, colaboré con mi compañero Aarón en el desarrollo del controlador y algunas vistas de usuarios, así como en el correcto funcionamiento de las vistas. También aporté en la creación de la base de datos junto con mis compañeros Aarón y Diego. Por último, cree el repositorio base e hice un esqueleto que fue modificándose a lo largo de la aplicación que sirvió como inicio.	**Nivel técnico:** ha sido una gran ventaja mi decisión de modificar la versión de SpringSecurity, porque dio muchos menos problemas. También mi manejo con la seguridad y las bases de datos han sido de gran ayuda para el proyecto tanto para el manejo de los logs como para el resto de las entidades. **Nivel trabajo de grupo:** ha habido una buena comunicación a lo largo del proyecto por lo no han surgido problemas de conflictos y nos hemos ayudado mutuamente	**Nivel técnico:** lo único que fue tedioso fue el tener que crear un interceptor para evitar las inyecciones por url a las vistas de login y logout. Pero tampoco fue nada realmente difícil. El resto del proyecto fue bastante asequible. **Nivel trabajo de grupo:** según mi forma de trabajar yo junto con Aarón teníamos más avanzadas nuestras respectivas partes que los demás, pero tampoco supuso nada grave, pues ese retardo no era excesivo. Aunque realmente no ha sido una desventaja, sino más bien una pequeña anotación.

		y reutilizado código entre todos. Cabe destacar la buena comunicación y horas de trabajo que he compartido con mi compañero Aarón.	
Diego Villa García	He completado los apartados de dar de alta, listar y borrar ofertas propias (6, 7 y 8) y establecer conversaciones y ver listado de conversaciones (12 y 13). También contribuí a diseñar el modelo de datos con Eduardo y Aarón.	**Nivel técnico:** Aunque no me pareció muy complicado al principio, completé bastante rápido los apartados de ofertas (6, 7 y 8). **Nivel trabajo en grupo:** Mantuvimos una buena comunicación a lo largo de todo el proyecto, lo que permitió agilizar mucho el proceso de desarrollo. Así pudimos dedicar los últimos días para repasar lo que hicimos y corregir errores menores.	**Nivel técnico:** Se me complicaron un poco los apartados de conversaciones (12 y 13). El problema fue que me costó entender cómo hacer que ambas partes viesen el mismo chat porque un usuario se obtiene a través del propio campo User pero el otro se obtiene a través de Offer. **Nivel trabajo en grupo:** Al principio del proyecto tardamos más de lo que pensé que tardaríamos en resolver el modelo de datos. Además, como unos apartados del enunciado dependían en cierta manera de los anteriores, había que esperar a tener hechas algunas partes para que el siguiente miembro pudiera continuar. Sin embargo, al final fuimos lo suficientemente rápidos para que no fuese un problema.
Santiago López Laso	He completado los requisitos de comprar una oferta y consultar el listado de ofertas compradas (10 y 11). Además, implementé las pruebas automatizadas de todos los requisitos obligatorios (desde la	**Nivel técnico:** El requisito 11 no me supuso ningún problema y gracias a las pruebas automatizadas que implementé conseguimos descubrir y resolver algunos	**Nivel técnico:** En el requisito 10 tuve dificultades para mostrar el mensaje de error al comprar una oferta, ya que al principio no sabía cómo pasar el mensaje a una función diferente a la que respondía a la

	prueba 1 hasta la prueba 34). También rellené el catálogo de casos de prueba.	errores en la aplicación. **Nivel trabajo en grupo:** Durante todo el proceso de desarrollo nos hemos comunicado para repartirnos el trabajo y resolver problemas, lo que permitió implementar todo lo necesario a tiempo. Crear el modelo de datos al principio ayudó mucho. En general, hubo un buen trabajo de equipo y todos los miembros del grupo ayudaron en el desarrollo.	petición de comprar. Al final lo resolví pasándolo como parámetro en la URL. **Nivel trabajo en grupo:** Nos costó un poco comenzar a trabajar porque tardamos en organizarnos y algunos requisitos dependían de otros anteriores. Aun así, luego hemos acelerado el trabajo y no hubo problemas graves.
Chen Xin Pan Wang	Me encargué de realizar el apartado 9 (completo), el 10 y el 11 (los he acabado y completado), todo con sus vistas correspondientes, los controladores, servicios y entidades. También en la creación del Word como el resto de mis compañeros.	**Nivel técnico:** No me pareció difícil porque ya teníamos los guiones de las prácticas que me han servido mucho. Me pude haber atascado en alguna implementación, pero mis compañeros me han ayudado a resolveros rápidamente. **Nivel trabajo en grupo:** Durante todo el trabajo hubo mucha comunicación por lo que pudimos agilizar la realización de este proyecto	**Nivel técnico:** Al no haber cursado la asignatura de SEW, al principio me ha costado un poco empezar (el proyecto y la asignatura), entonces en la parte e las vistas me pude haber atascado al principio un poco. **Nivel trabajo en grupo:** Nos ha costado un poco empezar a realizar el proyecto, la distribución y sobre todo el modelo de datos. También decir que algunos apartados dependían unos de otros y por eso nos ha costado empezar.