SDI – Sistemas Distribuidos e Internet

PRÁCTICA 2 DE ENTREGA – NODEJS

INFORME

Grupo 104-106

URL AWS	**NO DESPLEGADO (no necesario)**
Nombre1:	Marcos
Apellidos1:	Canal López
Email1:	UO258899@uniovi.es
Cód. ID GIT	104
Nombre1:	Martín
Apellidos1:	Fernández Prieto
Email1:	UO258619@uniovi.es
Cód. ID GIT	106

Índice

Introducción

Desarrollamos una aplicación Web de compra-venta entre particulares (al estilo Wallapop) en NodeJS, usamos el IDE WebStorm de Jetbrains.

El propósito de la aplicación es poner en contacto a vendedores de productos con usuarios que pueden estar interesados en ellos. Dichos usuarios pueden comunicarse entre sí para resolver cualquier duda que surja. Además, estos podrán comprar o vender productos y su saldo personal se vera modificado en dichos casos.

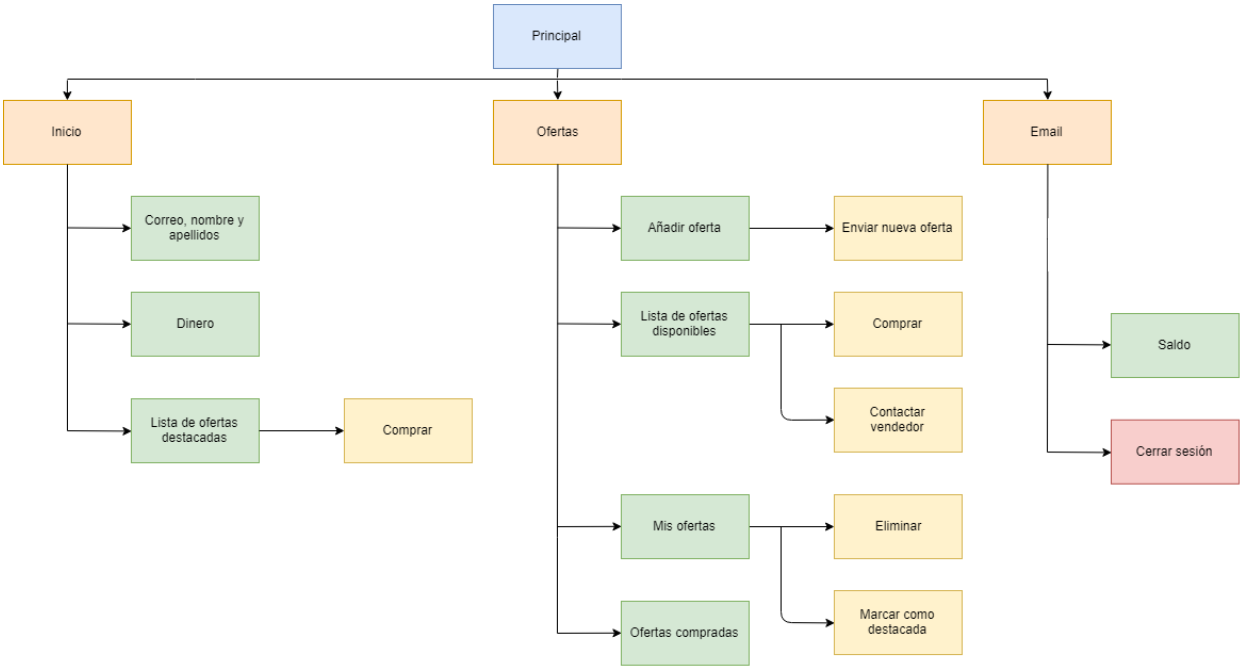
Como esta es una aplicación de prueba, los nuevos usuarios empezaran con un saldo de 100€, para permitirles comprar ofertas y ver su funcionamiento, además de por supuesto poder vender y también recibir el dinero de dicha compra.

Mapa de navegación (aplicación web)

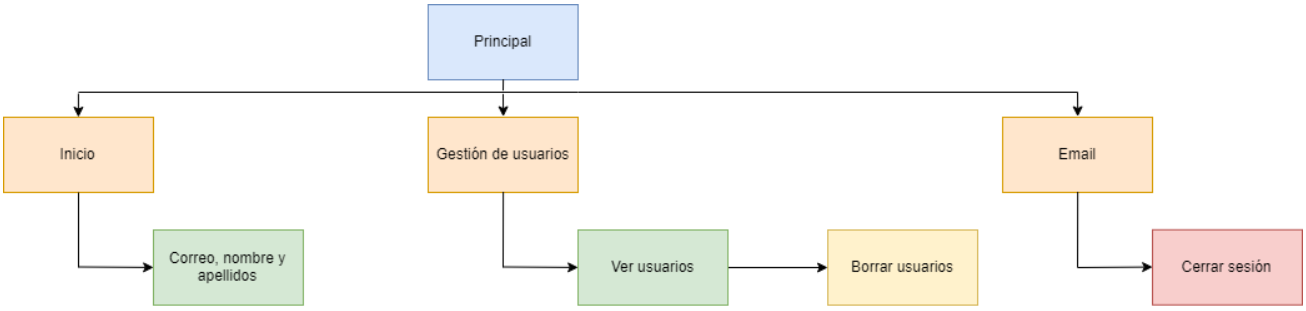
Se muestran a continuación los tres mapas de navegación para los distintos tipos de usuario: Usuario Registrado, Administrador y Usuario Anónimo.

LEYENDA: Azul (pagina principal de la aplicación), Naranja (Elementos del Nav), Verdes (Desplegables del nav o elementos de la vista), Amarillo (Acción) y Rojo (Cierre de sesión).

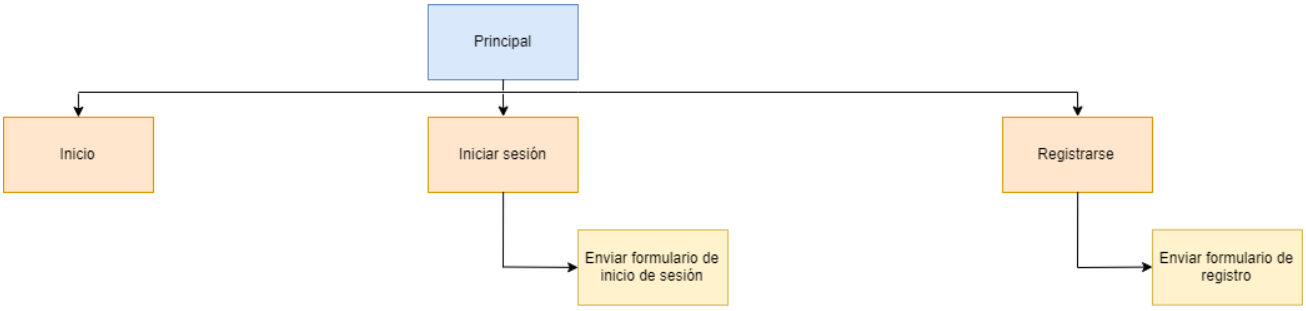
1. Mapa de navegación Usuario Registrado



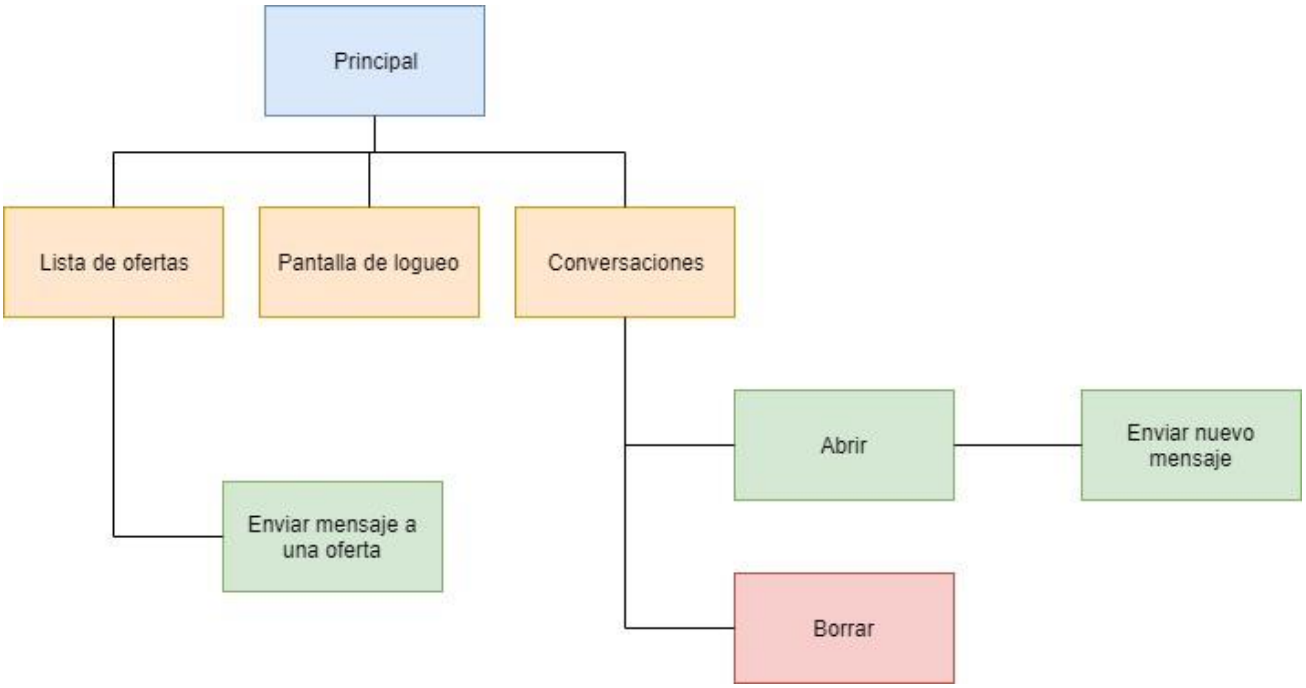
2. Mapa de navegación de Administrador



3. Mapa de navegación de publico (Anónimo)



Mapa de navegación (JQuery)



En el caso de JQuery, necesitas iniciar sesión para realizar cualquier acción, por mucho que intentes acceder solo te dejara hacer login. Una vez hecho iras automáticamente a la lista de ofertas, en las cuales únicamente podrás mandar un mensaje a dicha oferta. Lo mismo con las conversaciones que tan solo podrás abrir y cerrar. También dentro del apartado de la conversación abierta podrás enviar nuevos mensajes.

Casos de uso implementados

Parte 1

1. Público: Registro de usuario
2. Usuario Registrado: Inicio de sesión
3. Usuario Registrado: fin de sesión
4. Administrador: listado de usuarios
5. Administrador: Borrado múltiple de usuarios
6. Usuario: Dar de alta una nueva oferta
7. Usuario: listado de ofertas propias
8. Usuario: Dar de baja una oferta
9. Usuario: Buscar ofertas
10. Usuario: Comprar una oferta
11. Usuario: Ver el listado de ofertas compradas
12. OPTATIVO: Marcar una oferta como destacada

Parte 2

1. Identificarse como usuario vía token
2. Usuario: Mostrar listado de ofertas disponibles (de otros usuarios)
3. Usuario: Enviar mensajes a una oferta
4. Usuario: Mostrar mensajes de una conversación
5. OPTATIVO: Marcar mensaje como leído
6. OPTATIVO: Eliminar una conversación
7. Autenticación del usuario
8. Mostrar listado de ofertas disponibles
9. Enviar y visualizar mensajes a una oferta
10. OPTATIVO: Ver el listado de conversaciones
11. OPTATIVO: Eliminar una conversación
12. OPTATIVO: Marcar mensajes como leídos de forma automática
13. OPTATIVO: Mostrar el número de mensajes sin leer

En **resumen**, fueron implementados todos los casos de uso.

Catálogo de pruebas realizadas

1. Registro de Usuario con datos válidos (email vacío).
2. Registro de Usuario con datos inválidos (repetición de contraseña inválida)
3. Registro de Usuario con email existente.
4. Inicio de sesión con datos válidos

5. Inicio de sesión con datos inválidos (email existente, pero contraseña incorrecta).
6. Inicio de sesión con datos válidos (campo email o contraseña vacíos).
7. Inicio de sesión con datos inválidos (email no existente en la aplicación).
8. Hacer click en la opción de salir de sesión y comprobar que se redirige a la página de inicio de sesión (Login).
9. Comprobar que el botón cerrar sesión no está visible si el usuario no está autenticado.
10. Mostrar el listado de usuarios y comprobar que se muestran todos los que existen en el sistema.
11. Ir a la lista de usuarios, borrar el primer usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.
12. Ir a la lista de usuarios, borrar el último usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.
13. Ir a la lista de usuarios, borrar 3 usuarios, comprobar que la lista se actualiza y dichos usuarios desaparecen.
14. Ir al formulario de alta de oferta, rellenarla con datos válidos y pulsar el botón Submit. Comprobar que la oferta sale en el listado de ofertas de dicho usuario.
15. Ir al formulario de alta de oferta, rellenarla con datos inválidos (campo título vacío) y pulsar el botón Submit. Comprobar que se muestra el mensaje de campo obligatorio.
16. Mostrar el listado de ofertas para dicho usuario y comprobar que se muestran todas los que existen para este usuario.
17. Ir a la lista de ofertas, borrar la primera oferta de la lista, comprobar que la lista se actualiza y que la oferta desaparece.
18. Ir a la lista de ofertas, borrar la última oferta de la lista, comprobar que la lista se actualiza y que la oferta desaparece.
19. Hacer una búsqueda con el campo vacío y comprobar que se muestra la página que corresponde con el listado de las ofertas existentes en el sistema
20. Hacer una búsqueda escribiendo en el campo un texto que no exista y comprobar que se muestra la página que corresponde, con la lista de ofertas vacía.
21. Hacer una búsqueda escribiendo en el campo un texto en minúscula o mayúscula y comprobar que se muestra la página que corresponde, con la lista de ofertas que contengan dicho texto, independientemente que el título esté almacenado en minúsculas o mayúscula.
22. Sobre una búsqueda determinada (a elección de desarrollador), comprar una oferta que deja un saldo positivo en el contador del comprobador. Y comprobar que el contador se actualiza correctamente en la vista del comprador.
23. Sobre una búsqueda determinada (a elección de desarrollador), comprar una oferta que deja un saldo 0 en el contador del comprobador. Y comprobar que el contador se actualiza correctamente en la vista del comprador.
24. Sobre una búsqueda determinada (a elección de desarrollador), intentar comprar una oferta que esté por encima de saldo disponible del comprador. Y comprobar que se muestra el mensaje de saldo no suficiente.
25. Ir a la opción de ofertas compradas del usuario y mostrar la lista. Comprobar que aparecen las ofertas que deben aparecer.

26. Al crear una oferta marcar dicha oferta como destacada y a continuación comprobar: i) que aparece en el listado de ofertas destacadas para los usuarios y que el saldo del usuario se actualiza adecuadamente en la vista del ofertante (-20).
27. Sobre el listado de ofertas de un usuario con menos de 20 euros de saldo, pinchar en el enlace Destacada y a continuación comprobar: i) que aparece en el listado de ofertas destacadas para los usuarios y que el saldo del usuario se actualiza adecuadamente en la vista del ofertante (-20).
28. Sobre el listado de ofertas de un usuario con menos de 20 euros de saldo, pinchar en el enlace Destacada y a continuación comprobar que se muestra el mensaje de saldo no suficiente.
29. Inicio de sesión con datos válidos.
30. Inicio de sesión con datos inválidos (email existente, pero contraseña incorrecta).
31. Inicio de sesión con datos válidos (campo email o contraseña vacíos).
32. Mostrar el listado de ofertas disponibles y comprobar que se muestran todas las que existen, menos las del usuario identificado.
33. Sobre una búsqueda determinada de ofertas (a elección de desarrollador), enviar un mensaje a una oferta concreta. Se abriría dicha conversación por primera vez. Comprobar que el mensaje aparece en el listado de mensajes.
34. Sobre el listado de conversaciones enviar un mensaje a una conversación ya abierta. Comprobar que el mensaje aparece en el listado de mensajes.
35. Mostrar el listado de conversaciones ya abiertas. Comprobar que el listado contiene las conversaciones que deben ser.
36. Sobre el listado de conversaciones ya abiertas. Pinchar el enlace Eliminar de la primera y comprobar que el listado se actualiza correctamente.
37. Sobre el listado de conversaciones ya abiertas. Pinchar el enlace Eliminar de la última y comprobar que el listado se actualiza correctamente.
38. Identificarse en la aplicación y enviar un mensaje a una oferta, validar que el mensaje enviado aparece en el chat. Identificarse después con el usuario propietario de la oferta y validar que tiene un mensaje sin leer, entrar en el chat y comprobar que el mensaje pasa a tener el estado leído.
39. Identificarse en la aplicación y enviar tres mensajes a una oferta, validar que los mensajes enviados aparecen en el chat. Identificarse después con el usuario propietario de la oferta y validar que el número de mensajes sin leer aparece en la en su oferta.

En **resumen**, fueron implementados todos las pruebas unitarias.

Problemas encontrados a la hora de hacer las pruebas

Al resetear la BBDD, cuando insertamos los datos, los _id los metemos como String, ya que intentamos meterlos como ObjectID pero no nos dejaba, entonces modificamos los controladores para que traten los _id como strings en vez de como ObjectID.

De los test realizados los test 11,12,13 no pasan, desconcemos el motivo, pero la lista de usuarios sale vacía, sin embargo, si lo hacemos manual con los mismos pasos funcionan. El test 20 cuando se ejecuta con todos a la vez falla, si se ejecuta individual funciona correctamente. Al resetear la BBDD, cuando insertamos los datos, los _id los metemos como String, ya que intentamos meterlos como ObjectID pero no nos dejaba, entonces modificamos los controladores para que traten los _id como strings en vez de como ObjectID.

Aspectos técnicos y de diseño relevantes

Desarrollamos el proyecto usando NodeJS (v10.15.3), en el entorno de desarrollo WebStorm de Jetbrains en su última versión. Como base de datos usamos MongoDB Cloud versión 4.0.

Usamos como patrón para la capa de presentación el MVC (Modelo Vista Controlador) visto en clase, y se intento respetar en medida de lo posible a medida que el proyecto se fue realizando.

Información necesaria para el despliegue y ejecución

Para desplegar la aplicación es necesario descargarla de GitHub (https://github.com/martinlacorrona/sdi-actividad2-104-106) o directamente usar la adjuntada a esta documentación.

Usando la versión de Node v10.15.3, cargar el proyecto en la ultima versión de WebStorm, y usar la base de datos que ya esta incluida en el proyecto.

Realizando un "npm install" y correr la aplicación a partir del archivo "app.js" todo debería de funcionar a la perfección.

Para acceder a la aplicación web deberíamos ir a http://localhost:8081 y para la parte de JQuery a http://localhost:8081/cliente.html. La API esta alojada en http://localhost:8081/api, todos sus métodos son usados desde la aplicación de JQuery.

Para el proyecto de pruebas, descargarlo de nuevo de GitHub (https://github.com/martinlacorrona/sdi-actividad2-test-104-106) o directamente de la copia que esta adjuntada a esta documentación.

Usar STS con la versión 3.9.7, la versión 8 de Java e instalar todas las competencias con Maven, todo eso seria suficiente para que el proyecto funcionase a la perfección.