SnowyDune

Documentación del Proyecto

Proyecto Desarrollo de Aplicaciones Web

Oria de Rueda Pérez, Pablo
14-12-2020

ÍNDICE

Propuesta, Definición y Análisis del Proyecto.

1 - Definición del proyecto.

1.1 - Obtención de la información necesaria para la definición del proyecto.

La información para la definición del presente proyecto se ha recogido de fuentes muy variadas. La idea general surge de una inquietud personal por la realización de viajes a zonas invernales y de las experiencias vividas, producto de ese cúmulo de experiencias se resume que resultaría cómodo tener una web con los servicios centralizados y no dispersos por infinidad de webs distintas, así como tener que realizar pagos en muchas webs y muchos documentos de verificación.

La visita asidua en aplicaciones webs en las que visitar y poder ver servicios para conocer el funcionamiento en cuanto a la autogestión de los servicios por parte de las propias empresas.

Una vez definida la idea principal de la aplicación, se contemplaron las posibles tecnologías para su desarrollo, para ello se valoraron las tecnologías empleadas durante el curso, aunque finalmente se decidió desarrollar empleando otras diferentes y con mayor potencial en el mercado laboral.

1.2 Descripción detallada del proyecto.

Se trata de una aplicación en la que los usuarios van a poder customizar un viaje a su gusto, eligiendo la estación que deseen de países europeos y posteriormente los servicios que consideren necesarios o mejores para pasar la mejor estancia y experiencia en sus viajes.

Se tiene en cuenta, aparte de la creación de este viaje, la gestión de la información del propio usuario, pudiendo modificar su información, ver los viajes realizados, los viajes pendientes y realizar comentarios en los servicios/estaciones a las que hayan viajado.

La creación del viaje se inicia escogiendo unos datos básicos iniciales, que son concretamente el número de personas que van a realizar el viaje, la fecha de inicio del viaje, el número de días que se desea estar y además debe seleccionar una de las estaciones que se encuentran disponibles en el sistema.

Los viajes, inicialmente va a poder realizar a países y estaciones que se encuentra en la Unión Europea, pudiendo en un futuro ampliar a diversos países de otros continentes.

Una vez introducidos los datos iniciales, el usuario va a seleccionar los servicios, pudiendo escoger uno o varios servicios que desee. Esto se presenta en distintas páginas, cada una destinada a un tipo de servicio en las que se muestran los detalles de los mismos.

Posteriormente a realizar la creación de un viaje, este pasará a ser visible en una interfaz independiente que sirve de "carrito", en el que se podrá tener tantos viajes como el usuario decida crear.

Cuando el usuario decide que quiere realizar alguno de los viajes guardados en el carrito, debe añadir dicho viaje al "checkout" donde podrá realizar el pago que puede ser de un viaje o varios viajes, a través de un formulario de datos y añadiendo una forma de pago válida.

Todo lo anteriormente hace referencia a las acciones que un usuario normal puede realizar, pero de forma adicional, estos usuarios normales pueden solicitar obtener el rol de empresa. Con el rol de empresa, van a poder registrar su negocio, el cual va a tener que ser revisado y aceptado por un usuario administrado. Cuando el negocio es aceptado, el usuario empresa puede realizar el registro de los servicios de los que dispone su empresa que al igual que la empresa, deben ser revisados por los administradores y cuando el usuario cuenta con los servicios aceptados, va a poder visualizar un listado de los servicios al igual que un listado de los comentarios recibidos por parte de los usuarios que han dado uso de sus servicios.

En lo que se refiere al usuario administrador, será el encargado de revisar y gestionar de forma correcta los comentarios, las empresas, los servicios y realizar la inclusión de nuevas estaciones en la aplicación.

Para ingresar en la aplicación es necesario tener una cuenta previamente creada, a través de la cual, acceder a la interfaz de bienvenida.

El registro se realiza mediante un formulario que comprobará que los datos son válidos de forma previa al envió al servidor.

La aplicación cuenta con seguridad suficiente como para impedir accesos indeseados a distintas páginas de esta, segmentando también las capacidades de los distintos roles disponibles.

1.3 Ámbito del proyecto.

El proyecto busca la creación de una aplicación que sirva para centralizar y facilitar la obtención de información relativa a los viajes a zonas invernales, de forma que se comprima información que de otra forma se encuentra completamente dispersa por diferentes páginas webs, simplificando la forma en la que un usuario pueda reservar un viaje sin necesidad de buscar de forma individual cada servicio.

Por tanto, nuestro ámbito se puede asemejar a los que realizar una agencia de viajes en un entorno online, centrándonos en lo que se refiere a negocios y viajes a zonas concretas, donde practicar y disfrutar de servicios específicos de esas zonas.

1.4 ¿Adaptación o creación?

Se puede decir que el proyecto aúna tanto adaptación como creación ya que, por una parte, se trata de una adaptación más actual de una agencia de viajes tradicional en la que los usuarios personalizan sus viajes hasta cierto punto mientras que es una aplicación que personalmente, no he encontrado a este nivel de detalle, por tanto, se puede considerar como una prolongación en cuanto a funcionalidades de algo ya creado.

1.5 Conceptos básicos del proyecto software.

Para la correcta comprensión del documento conviene conocer ciertos conceptos básicos que se explican a continuación.

- Carrito: Se trata de una zona de la aplicación que reúne todos los viajes creados en la página web.
- Popup: Se trata de ventanas emergentes que genera la aplicación, con el fin de dar información, realizar confirmaciones o avisar al usuario.
- Checkout: Hace referencia al proceso en el cual, el usuario realiza el pago del viaje.
- Base de datos Relacional: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
- ORM(Mapeo Objeto-Relacional): Es una técnica de programación para convirtir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.
- Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con módulos de software concretos, que pueden servir de base para la organización y desarrollo de software.
- TypeScript: Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft, se trata de un superconjunto de JavaScript.
- Angular: Es un framework para aplicaciones webs desarrollado en TypeScript, de código abierto que se utiliza para crear y mantener aplicaciones web de una sola página.
- Spring Framework: se trata de un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

Planificación del proyecto web.

1.6 Definición de tareas.

- Motivación Inicial. La necesidad de crear un proyecto provoca la idea inicial que implicó una búsqueda de una opción de creación de una aplicación viable.

- Planificación Inicial. Proceso en el que se llevó a cabo una base sobre la que cimentar los primeros pasos del desarrollo del aplicativo y son los siguientes:

- Descripción del proyecto. Se trata de la primera descripción de la idea inicial y sobre la cual se desarrolla la aplicación.
- Alcance del proyecto. Se fija el ámbito sobre el cual se va a trabajar.
- Stack tecnológico. Basándonos en la situación actual de los lenguajes/frameworks del mercado se dijaron los lenguajes a emplear en el proyecto.
- Especificación de requisitos. Se crean los requisitos que va a tener que cumplir el proyecto, tanto los funcionales como los no funcionales.
- Mapa del sitio. Se realiza una descripción de las pantallas necesarias, así como el flujo existente entre estas pantallas.
- Modelado de datos. Creación del modelo inicial, empleando una base de datos relacional, fijando las tablas con sus correspondientes campos y las relaciones necesarias que se establecen entre las distintas tablas.
- Casos de uso. Se crea un documento registrando los casos de usos más relevantes de la aplicación, fijando los diagramas pertinentes con los actores y el proceso de los casos tratados.
- Diagramas de actividad. Con los casos de uso establecidos, se realizan los diagramas de actividad los cuales desarrollan los distintos usos de forma más detallada.
- Realización de maqueta. Empleando el software Axure, se realiza un MockUp con el aspecto que se desea que tenga la aplicación.

- Fase de desarrollo del aplicativo.

- Realización del prototipo. Basándonos en el MockUp realizado en Axure, se realiza el prototipo empleando HTML y CSS estableciendo la base estética del proyecto y la navegación entre las distintas interfaces de la aplicación.
- Estructura inicial de Back-end. Realización de una estructura inicial empleando Spring.
    - Creación del modelo de datos empleando un ORM.
    - Creación de los repositorios necesarios empleando Spring.
    - Creación de los servicios necesarios para comenzar la aplicación.
    - Creación de los controladores que emplean estos servicios y mapean el acceso que se realizará desde el Front-end.
- Implementación de la seguridad en la aplicación (Back-end).
    - Investigación para empleo de Json Web Token.
    - Implementación de JWT en Back-End.
- Estructura inicial de Front-end. Se realiza la primera aproximación a la tecnología elegida para el Front (Angular).
    - Creación de las vistas, sin lógica aún, que se desarrollaran.
    - Creación de los distintos servicios, sin lógica aún.
    - Creación de los distintos componentes.
- Implementación de la seguridad en la aplicación (Front-end).
    - Investigación del empleo de Json Web Token en Front.
    - Implementación de JWT en Front-end.
- Migrar el diseño HTML/CSS a el proyecto Angular. Se lleva a cabo una migración del diseño a Angular.
- Creación de datos en la base de datos relacional empleada. Se crean datos necesarios para llevar a cabo las pruebas en Front-End.
- Prueba de los servicios empleando Postman. Se consumen los servicios creados en Back-end para asegurar el correcto funcionamiento de estos.
- Codificación en Angular. Se comienza la realización de la codificación necesaria empleando Angular.

    - Realización de los servicios que se consumirán.
    - Construcción de las distintas vistas empleando los servicios.
    - Adición de comprobaciones en formularios.
    - Construcción de diversos componentes empleando librerías de Angular.
    - Implementación de la navegabilidad entre las distintas vistas.
- Realización del responsive de la aplicación. Para ello se realiza empleando la librería de Bootstrap 4.
- Pruebas manuales de la aplicación. Revisiones manuales de las distintas funcionalidades del aplicativo, detectando bugs y corrigiéndolos.
- Despliegue de la aplicación. Se lleva a cabo el despliegue en una máquina virtual Proxmox, donde será visible.
- Revisión de la aplicación posterior al despliegue. Se realiza la revisión de las diferentes funcionalidades de la aplicación.
- Creación de la documentación necesaria referente al desarrollo de la aplicación, a la instalación/despliegue y a los manuales de uso.

1.7 Estimación de tiempos.

Los tiempos de realización son aproximados y puede que en algunos casos poco específicos. Se definen a continuación:

Tarea	Tiempo (H)
Planificación Inicial	**Total - 34**
Descripción del proyecto.	1
Alcance del proyecto.	1.5
Stack tecnológico.	0.5
Especificación de requisitos.	1
Mapa del sitio.	2
Modelado de datos.	3

Casos de Uso.	4
Diagramas de Actividad.	5
Realización de Maqueta.	16
Fase De Desarrollo	**Total – 265.5**
Realización de Prototipo.	25
Estructura Inicial Back-end.	
Creación de Modelo de Datos.	10
Creación Repositorios.	4
Creación de Servicios.	25
Creación de controladores.	17
Implementación de seguridad en Back-end.	
Investigación JWT.	3
Implementación JWT.	7
Estructura Inicial Front-end	
Creación de las vistas.	1.5
Creación de los servicios.	1.5
Creación de los componentes.	1.5
Implementación de seguridad en Front-end.	
Investigación JWT.	4
Implementación JWT.	8
Migración diseño HTML/CSS a Angular.	6
Creación de registros en la base de datos.	5
Prueba de los servicios con Postman.	7
Codificación en Angular.	

Realización de los servicios.	20
Construcción de las distintas vistas.	30
Comprobaciones en formularios.	12
Construcción de componentes.	30
Navegabilidad.	5
Realización del responsive.	13
Pruebas manuales.	5
Despliegue de la aplicación.	5
Revisión posterior al despliegue.	2
Creación de la documentación.	18

Análisis del Proyecto Web.

1.8 Requisitos técnicos Web.

En este apartado se recogen los distintos entornos de desarrollo, los lenguajes empleados en la aplicación.

- Desarrollo de Back-end.

Para el desarrollo de la parte del back-end se ha empleado como entorno de desarrollo Eclipse y el lenguaje empleado ha sido Java.

Se ha llevado a cabo empleando Spring, que es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

- Desarrollo de Front-end.

La parte dedicada al Front-end se ha desarrollado empleado como entorno de desarrollo Visual Studio Code.

En el diseño se ha empleado HTML y CSS, adicionalmente para el responsive se ha empleado el framework Bootstrap 4.

Como lenguaje de programación el desarrollo se ha realizado empleando TypeScript con el framework Angular.

- Base de Datos.

Se ha empleado una base de datos relacional SQL, usando como gestor de base de datos MySQL y MySQL Workbench como herramienta visual de bases de datos.

- Gestor de versiones.

Durante el desarrollo del proyecto se ha realizado un control de versiones, empleando Git y gestionado usando el software SourceTree.

- Realización de pruebas.

Las pruebas se han realizado en dos niveles.

  - Back-end. Pruebas manuales realizadas a los servicios que se consumen en Front-end. Pruebas realizadas empleando el programa Postman.
  - Front-end. Pruebas manuales realizadas en la aplicación usando un navegador web como Chrome. Detectando y depurando los errores encontrados.
- Despliegue de la aplicación.

  Se ha llevado a cabo el despliegue de la aplicación en una máquina virtual Proxmox, en la que se han instalado y empleado los entornos/programas similares que en el desarrollo.

1.9 Elementos del contenido.

En cuanto a los elementos del contenido de la web, distinguimos tres partes generales en todas las interfaces.

- Barra de navegación. Es el elemento que permite la navegación entre las distintas páginas de la aplicación y en función de los roles disponibles, se mostrarán distintas opciones en el menú. Este elemento se mantiene en todas las páginas.
- Contenido principal. Es el grueso de nuestras interfaces, se trara de la sección donde se recoge la información relativa a la aplicación, al igual que los formularios, las tarjetas de información y los popups.
  Este contenido se estructura generalmente de la siguiente forma.
  - Nombre descriptivo de la página. En la cabecera se introduce una descripción breve y concisa, de pocas palabras que sirve para orientar al usuario de la zona de la aplicación en la que se encuentra.
  - Contenido. Este contenido va en función de la página, por ejemplo, en las secciones de los servicios serían las tarjetas de los distintos servicios disponibles.
  - Botones para navegación. Permiten el flujo del usuario a través del aplicativo.

    - Validaciones. En los formularios se disponen validaciones en los campos de datos.
    - Confirmaciones. En muchas acciones se solicita la confirmación para realizar las acciones.
- Pie de página. Elemento que, igual que la barra de navegación, se mantiene en todas las páginas y permite el acceso a redes sociales y al formulario de contacto.

1.10 Diseño gráfico.

Para el diseño, se ha buscado que sean interfaces claras e intuitivas, se han empleado colores claros y que guardan relación con la tipología de los viajes que se tratan en la aplicación.

Los colores adoptados son el blanco y el azul, empleados tanto en las distintas páginas de la aplicación como en el logotipo realizado, son colores que transmiten limpieza, paz en el caso del blanco y cielo, agua o tranquilidad en el caso del azul. Esto se transmite en el logotipo, un símbolo de la cristalización del agua de color azul.

Tanto el tipo de letra como el tamaño de la letra buscan que sea de fácil y cómoda lectura, en el caso del tamaño de la letra, varía en función de la resolución de la pantalla, permitiendo lecturas en móviles y tablets.

También se emplean espacios libres y fondos tranquilos e indicativos de la interfaz en la que nos encontramos.

En un futuro se va a intentar optar por la opción de agregar una opción de colores “nocturnos”, dando la opción al usuario a poder verlo correctamente, evitando los colores que deslumbren y provoquen mucha luz.

1.11 Herramientas para Web.

No se han empleado muchas herramientas para webs de cara al diseño final, solamente se ha empleado Google Font para modificar la fuente de letra.

Durante la fase de creación del prototipo se ha empleado una extensión para poder “calcar” el diseño denominada “Perfect Pixel”.

Ingeniería del proyecto web.

1.12 Diseño del software.

Para el diseño del aplicativo se ha escogido una arquitectura de tres capas (presentación, aplicación y persistencia). La utilización de esta arquitectura se debe a que los distintos niveles son independientes unos de otros de manera que se puede cambiar fácilmente el comportamiento de las clases en el nivel de la aplicación sin que ello influya en las otras capas.

En los siguientes apartados vamos a explicar las distintas capas del software.

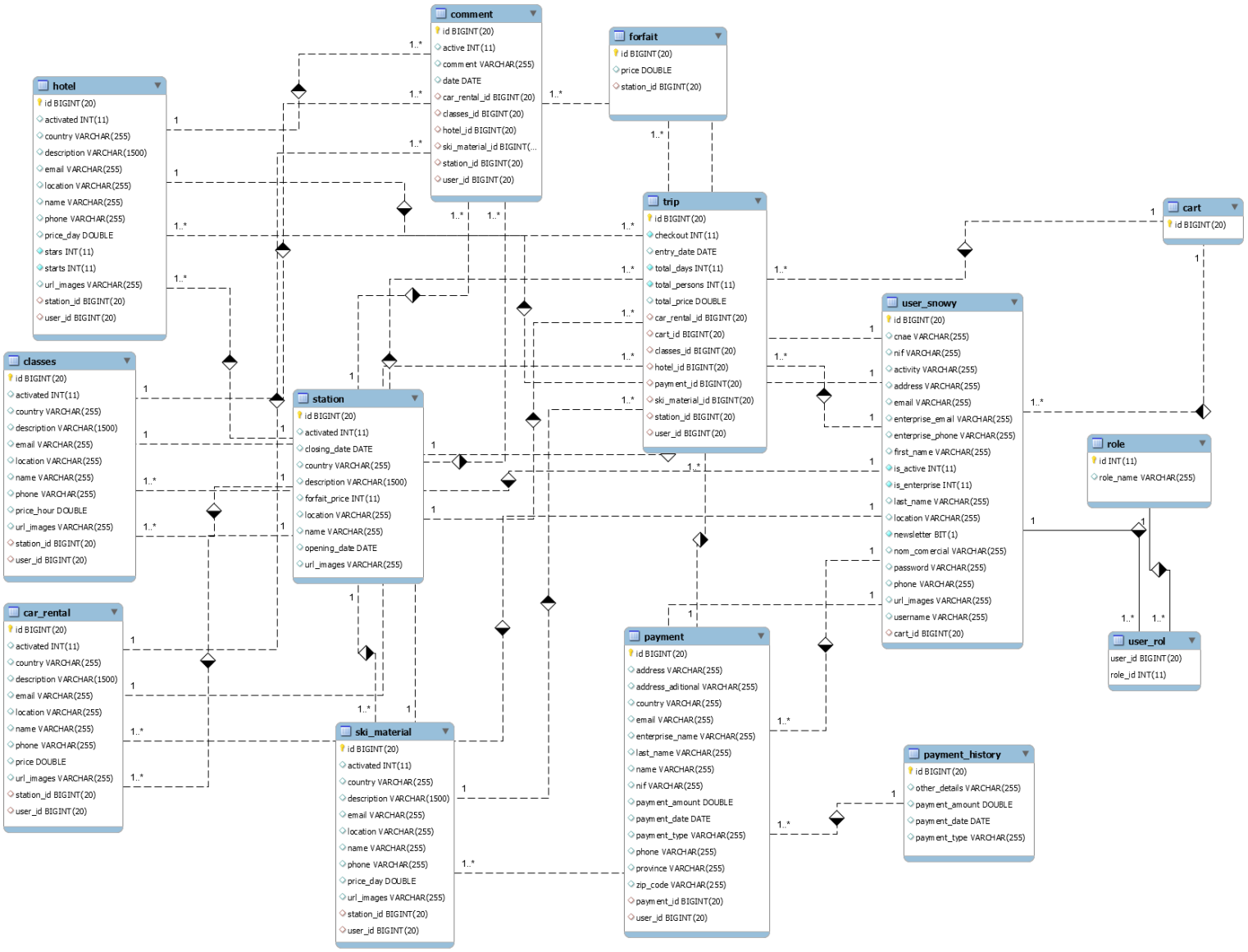
1.13 Diseño de datos.

En este apartado hablamos de la capa de persistencia, que se corresponde con la base de datos de la aplicación y las distintas tablas que la conforman.

Estas tablas son:

- User_snowy.
- Role.
- Cart.
- User_rol.
- Payment
- Payment_history.
- Trip.
- Station.
- Forfait.
- Comment.
- Hotel.
- Classes.
- Ski_material.
- Car_rental.

A continuación, se muestra el diagrama Entidad-Relación utilizado para la implementación de la base de datos del aplicativo.



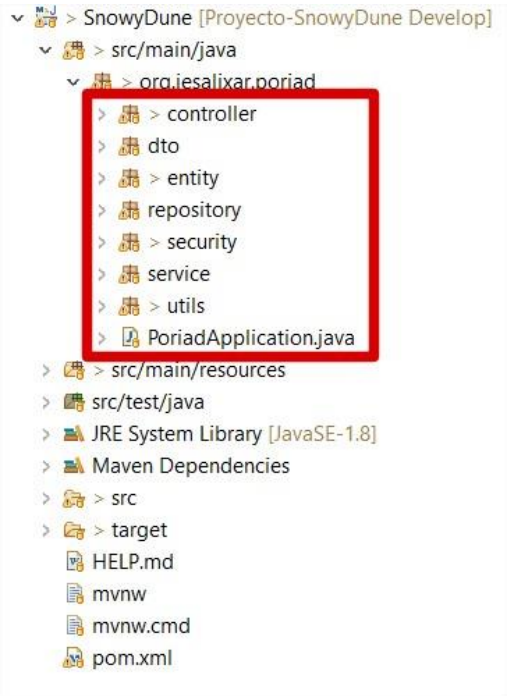
1.14 Diseño del sistema.

En este apartado nos centramos en la capa de lógica de la aplicación, es decir, el conjunto de componentes que implementa la funcionalidad de la aplicación web.

Esta capa sirve de enlace entre los niveles de presentación y de persistencia, ya que la capa de presentación no accede a la base de datos directamente, si no que se comunica con la capa de la aplicación para demandarle el servicio deseado y es la capa que se comunica con la capa de persistencia para recuperar los datos necesario.

El diseño se ha realizado empleando el lenguaje Java y un framework de Java (Spring 5) en Eclipse IDE.

Nuestra capa de presentación presenta la siguiente paquetería.



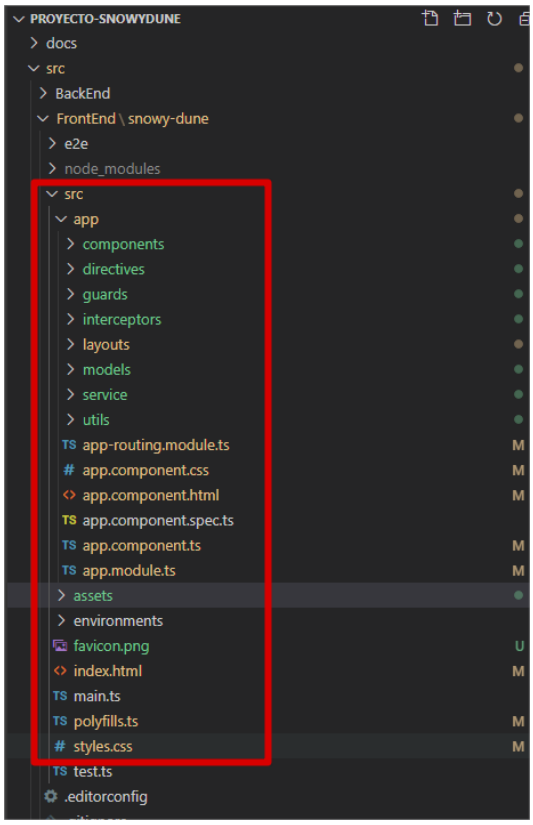
- Controller. Gestiona todos los accesos a los servicios desde la capa de presentación.
- Dto. Sirven para crear objetos planos que contienen información de múltiples fuentes que se concentran en una sola clase para evitar que sean necesarias varias invocaciones.
- Entity. Recoge los modelos de las distintas clases que se recogen en el proyecto.
- Repository. Nos lo provee Spring y nos permite acceder a servicios de forma más cómoda.
- Security. Recoge todo lo relativo a la seguridad implementada en el aplicativo mediante el empleo de Json Web Token.
- Service. Sirve como intermediario entre los repositorios y los controladores.
- Utils. Paquete que almacena las clases que proporciona métodos útiles pero que no pertenecen a la aplicación como tal.
- Carpeta Resources. Almacena los datos de configuración de la aplicación y de la base de datos.

En el caso de nuestra aplicación, los encargados de gestionar el intercambio de información serán los controladores, que proveen de endpoints a los que podrán acceder en función de los roles que los soliciten.

1.15 Diseño de la interfaz de usuario.

Esta parte se corresponde con la capa de presentación que es el conjunto de componentes que implementan la interacción con los usuarios a través de una representación visual de la aplicación. A partir de la interfaz gráfica el usuario podrá navegar por las distintas páginas para poder obtener toda la información que desee, o aportarla en caso de ser necesario.

La gestión de la información se realiza empleando el framework Angular desarrollado usando como lenguaje TypeScript. Para el responsive se ha usado el framework Bootstrap 4.

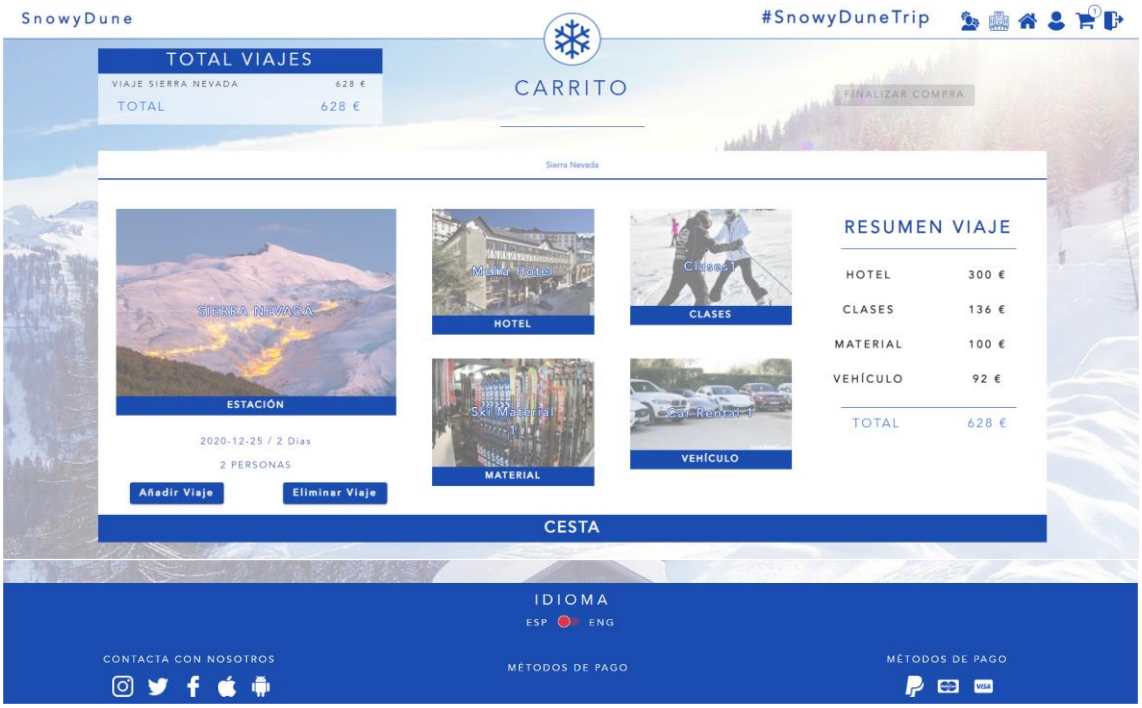


La distribución que se ha adoptado es la siguiente.

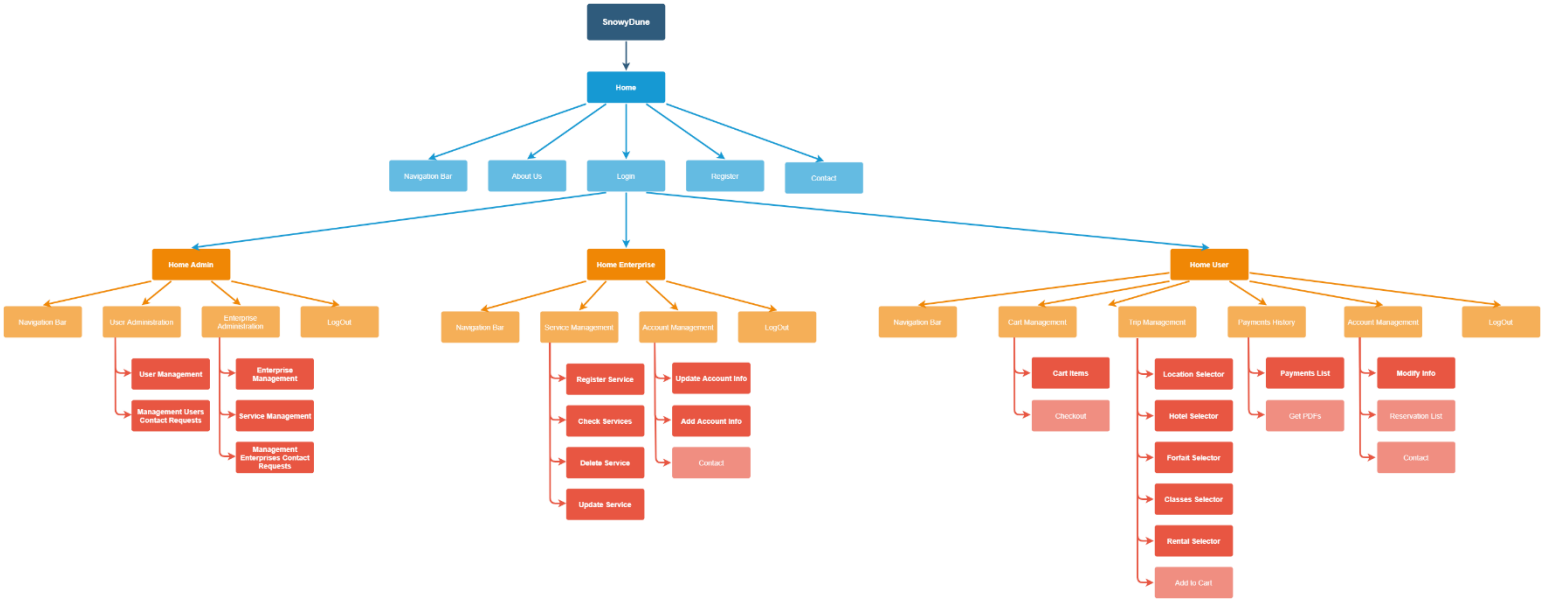
- Components. Se almacenan los distintos componentes creados, proporcionando modularidad a la aplicación, y que posteriormente se emplean en las vistas.
- Directives. Se trata de extensiones de un template con características propias.
- Guards. Encargado de gestionar los roles que se permiten en cada página de la aplicación.
- Interceptors. Encargado de enviar el Json Web Token en cada petición que realizamos a los endpoints del servidor.
- Layouts. Se trata de las vistas que se han creado para la aplicación, incluyen a los componentes creados anteriormente.
- Models. Modelos de las clases que se encuentran en el servidor.
- Service. Parte de la aplicación que se encarga de realizar las peticiones al servidor.
- Utils. Dedicado a métodos o partes de código externos a una funcionalidad concreta.

Los encargados de mostrar la información y las páginas van a ser los "layouts" que reciben módulos de componentes o implementan la lógica necesaria para transformar la información que proviene de los servicios en algo visual.

Como ejemplo visual, se dispone a continuación una interfaz que recibe información de nuestro servidor.



También se adjunta el mapa de navegación de nuestro sitio web, este mapa se puede ver con detalle en el repositorio de GitHub destinado a tal fin.



1.16 Diseño del contenido.

El contenido de la aplicación se encuentra recibido, en su mayor parte, por los usuarios, tanto los usuarios normales como los usuarios empresa, que aportan los servicios relacionado a cada estación de ski.

Además de los servicios que aportan los usuarios empresa, los comentarios juegan un papel importante en el diseño del contenido, por tanto, contar con un flujo de usuarios tanto normales como empresa es básico para obtener un correcto funcionamiento del sistema.

Esta información se encuentra paginada para evitar cargas excesivas por parte del servidor y se distribuyen de forma holgada, dejando márgenes para su correcta visualización.

El contenido que ingresa directamente un usuario de SnowyDune son las estaciones, las cuales hay que registrarlas con los datos requeridos y una imagen. Además todo el contenido que se publica en la aplicación es revisado por un administrador, pudiendo filtrar el contenido si es necesario.

Identificación y cuantificación de contingencias.

Dado que nos encontramos en un entorno muy dinámico, es importante definir los escenarios de interrupción y falla de los distintos componentes de nuestro aplicativo.

Antes de la puesta en producción de nuestra aplicación se debe realizar una serie de requisitos fijados a continuación:

- Identificación de las funcionalidades esenciales. Se debe detectar que funcionalidades son básicas para el correcto funcionamiento de la aplicación. En nuestro caso podrían ser las siguiente:
    - Obtener la información de las estaciones y servicios.
    - Ingresar en el sistema.
    - Visualizar la información personal.
    - Registrar empresas y servicios.

  Si no se recibe la información de nuestro servidor puede ser por dos razones. El servidor no se encuentra operativo o la base de datos está fallando en algún punto.

  En el caso de que el servidor no se encuentre operativo, debe darse prioridad absoluta a solucionar este problema. Si el problema es la base de datos, se podría optar por contar con una base de datos espejo, la cual podría recibir el volcado de la información de la principal y en caso de fallar la principal, apuntar nuestro servidor hacia esa base de datos. Al haberse realizado el proyecto con un ORM, podemos incluso contar con una base de datos distinta a la actual, siempre y cuando sea relacional.

- Identificación de funcionalidades secundarias. No son esenciales, pero afectan a la experiencia del usuario, y pueden ser tanto visuales como de funcionalidades focalizadas.
    - Versiones de software obsoletas. Con el paso del tiempo, las versiones actuales podrían quedar obsoletas, por ejemplo, el

navegador de versión superior puede llegar un punto en el que no podría renderizar ciertas partes de la aplicación.

- Pérdida de la distribución o del responsive. Esto se puede deber a que las librerías de Bootstrap 4 por alguna razón no se encuentran disponibles. Para evitar este hecho, se podría guardar en local unas librerías que nos funcionen con todas las implementaciones realizadas, de forma que, si las librerías externas no están disponibles, se empleen las locales.
- Caída en el servidor de Imgur. Si el servidor de Imgur se encuentra caido, no podremos acceder a la API que guarda las imágenes y, por tanto, no se almacenarán las imágenes.

  Para ello se podría contar con un servicio adicional en estos casos.
- Abandono de servicio de alguna de las librerías usadas. Con el tiempo puede que alguna de las librerías usadas deje de tener servicio. En este caso se debe migrar la aplicación a otras librerías.

- Tener en cuenta la posibilidad de sufrir ataques externos, ya sea a la base de datos o al servidor con el fin de extraer información de ellos. En estos casos se debe contar con una seguridad suficiente.

  Para conocer nuestras vulnerabilidades de forma más detalla y profesional, se pueden contratar auditorías externas de empresas encargadas de seguridad, evitando estos problemas.

2 - Aseguramiento de la calidad.

Para garantizar la calidad de la aplicación, se deben realizar pruebas tanto a nivel interno como a nivel de usuario.

Se han realizado pruebas a nivel de Back-end, comprobando el correcto funcionamiento de los servicios que necesita el Front-end para su funcionamiento empleando Postman.

En un futuro se deben implementar automatizaciones con el fin de obtener los casos posibles tanto que la respuesta sea favorable como los casos en los que la respuesta sea negativa.

Con estas automatizaciones conseguimos que la detección de errores se pueda realizar con cada modificación que sufra nuestra aplicación. Estas pruebas de regresión serán fundamentales en la detección de bugs derivados de una variación de cualquier parte del aplicativo.

También se debe introducir en la aplicación suficientes datos para poder probar de forma inicial, tanto a nivel de usuario como a nivel de servidor.

La parte del Front-end se ha probado de forma manual, probando las funcionalidades, los botones, la correcta visualización de la información/imágenes.

La aplicación ha sido probada por usuarios ajenos al desarrollo, para obtener un feedback sobre las funcionalidades y las observaciones realizadas por ellos.

Aunque se ha obtenido ese feedback, los usuarios corrientes no son los adecuados para la detección de bugs o al menos no en poca cantidad, lo ideal es que sea probado por un tester ajeno al desarrollo.

Aun habiendo probado funcionalidades cada día, el desarrollo sin bugs es muy difícil en este nivel y es posible que se sigan encontrando.