Desarrollo de Aplicaciones Web
[Proyecto]

Dejando Huella

Documentación

Rafael Ortega Armario

Índice

Propuesta, Definición y Análisis del Proyecto

Definición del proyecto

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 centralización de la información de todos los centros de adopción animal, así como las diferentes protectoras animales que se encuentran en el país. Para facilitar en la medida de lo posible la adopción de los animales que requieran de un hogar.

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 los propios centros de acogida.

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, y finalmente fueron estas las elegidas para desarrollar la aplicación.

Descripción detalla del proyecto

En esta aplicación habrá dos perfiles importantes, que son el de Usuario y el de Centro de Adopción. Cada uno de estos perfiles tendrá acceso a diferentes apartados en la aplicación, así como diferentes funciones dentro de ella.

Los Centros de Adopción serán los encargados de subir anuncios de los animales que se encuentren en adopción en ese momento, así como de su modificación o borrado si fuera necesario. Además de esto, una vez un usuario se decida a adoptar, deberán registrarlo en la web, para que haya constancia en la base de datos. De esta forma el animal dejará de aparecer en la búsqueda de animales en adopción.

A su vez, los Usuarios podrán tener acceso a la información de los animales en adopción o perdidos en ese momento. Una vez obtengan la información deseada, se podrán poner en contacto con el Centro para arreglar las adopciones. También podrán subir y gestionar anuncios de animales perdidos, en caso de que estén buscando a su mascota.

Por último, el administrador de la aplicación será el encargado de gestionar el alta de los Centros, así como la gestión de los Usuarios/Centros que hagan un mal uso de la aplicación. También podrán gestionar los anuncios de los animales perdidos y registrar vía web a otros administradores.

Ámbito del proyecto.

El proyecto busca la creación de una aplicación que sirva para centralizar y facilitar la adopción de los animales que necesiten un hogar, de forma que se comprima la información que de otra forma se encuentra completamente dispersa por diferentes páginas webs de los diferentes centros y protectoras, simplificando así la forma en la que un usuario pueda encontrar al animal indicado lo más cerca posible de su hogar.

Por tanto, nuestro ámbito se puede asemejar a los que realizar una web de adopciones normal y corriente, pero aglutinando toda la información del ámbito nacional en cuanto a animales en adopción.

¿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 un centro de acogida de animales tradicional en la que los usuarios se ponen en contacto con los centros para adoptar a un animal. Pero a su vez no he encontrado a otros centros que al mismo tiempo tengan una sección para que los usuarios puedan colgar anuncios para sus mascotas perdidas.

Y no solo eso, la web acepta la adopción de todo tipo de animales, no solo centrarse en perros y/o gatos.

Conceptos básicos del proyecto software.

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

- Popup: Se trata de ventanas emergentes que genera la aplicación, con el fin de dar información, realizar confirmaciones o avisar al usuario.
- 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 convertir 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.
- Materialize: es el Framenwork de diseño responsivo más innovador creado y diseñado por Google y basado en Material Design.

- Spring Framework: se trata de un Framenwork para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

Planificación del proyecto web

Definición de tareas

La 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 dibujaron 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 online Figma, se realiza una maqueta con el aspecto que se desea que tenga la aplicación.

Fase de desarrollo del aplicativo:

- Realización del prototipo: basándonos en el Figma realizado, se monta el prototipo empleando HTML, CSS y Materialize 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).
- Estructura inicial de Front-end. Se realiza la primera aproximación a la tecnología elegida para el Front (Materialize).
    - 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).
- 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 SpringToolSuite4: se consumen los servicios creados en Back-end para asegurar el correcto funcionamiento de estos.
- Realización del responsive de la aplicación: para ello se realiza empleando la librería de Materialize.
- 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 un PC normal usando Docker, SpringToolSuite4 y HeidiSQL.
- 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.

Estimación de tiempos

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

Tarea	Tiempo (Horas)
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 - 243.5**
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
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	
Migración diseño HTML/CSS a Materialize	6
Creación de registros en la base de datos	5
Prueba de los servicios	7
Codificación	
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

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 SpringToolSuite4 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 Materialize.
- Base de Datos.
    - Se ha empleado una base de datos relacional SQL, usando como gestor de base de datos MySQL y HeidiSQL 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 GitHub.
- 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.
    - o 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 un PC ordinarios, en el que se han instalado y empleado los entornos/programas similares que en el desarrollo.

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 trata 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 la sección de los animales en adopción serían las tarjetas de los distintos animales 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, al formulario de contacto y los links amigos de otras webs relacionadas con el tema de acogida animal.

Diseño gráfico

Para el diseño, se ha buscado que sean interfaces claras e intuitivas, se han empleado colores claros y que guardan relación con una actitud alegre y positiva.

Los colores adoptados son el verde claro, mostaza y el rojo, empleados tanto en las distintas páginas de la aplicación, son colores que transmiten alegría o ternura.

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.

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.

Ingeniería del Proyecto Web

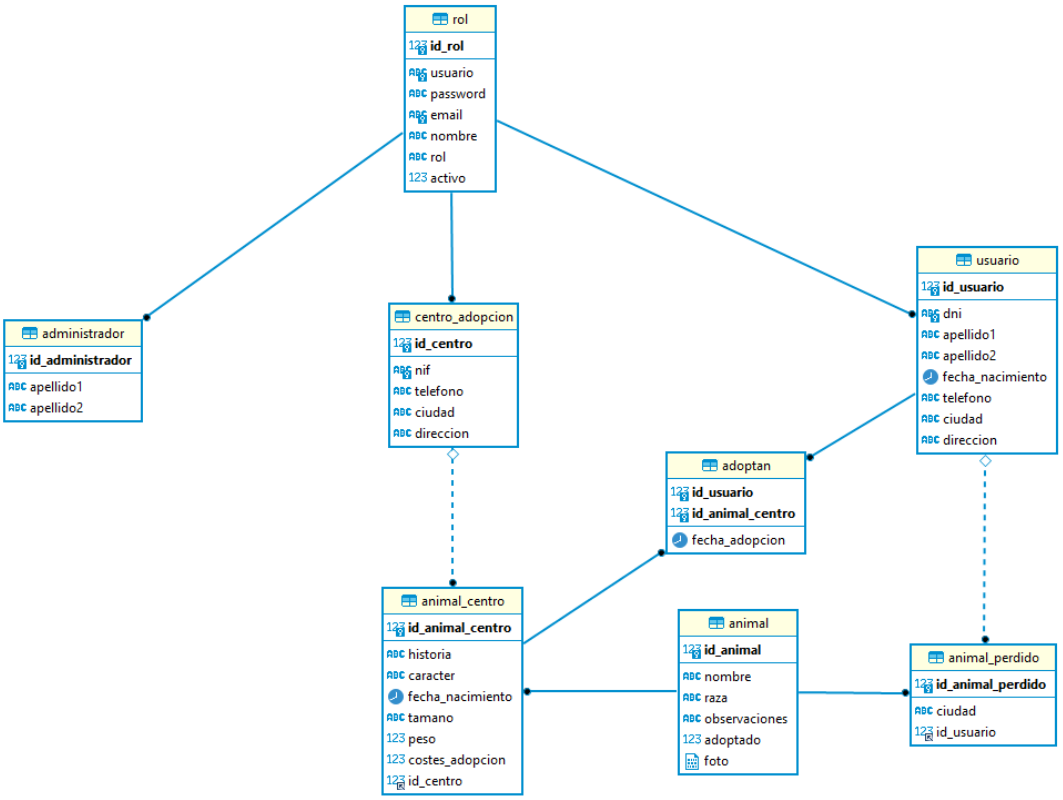
Diseño del software

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:

- Administrador
- Adoptan
- Animal
- Animal_centro
- Animal_perdido
- Centro_adopcion
- Rol
- Usuario

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



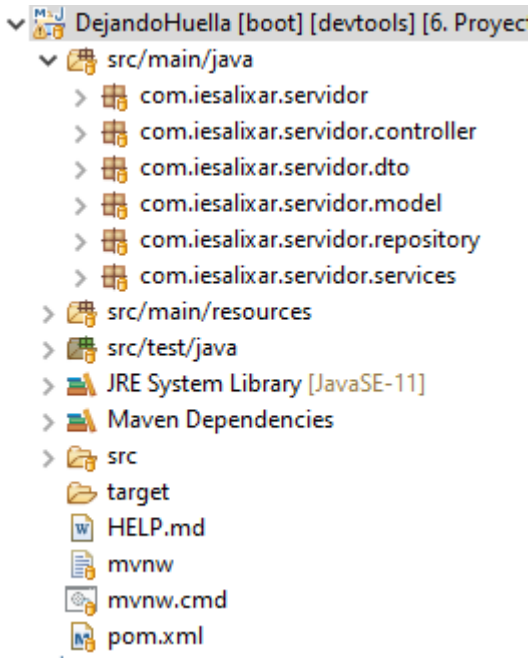
Diseño del sistema

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 en SpringToolSuite4.

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.
- Model: 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.
- Servidor: recoge todo lo relativo a la seguridad implementada en el aplicativo.
- Service: sirve como intermediario entre los repositorios y los controladores.
- 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.

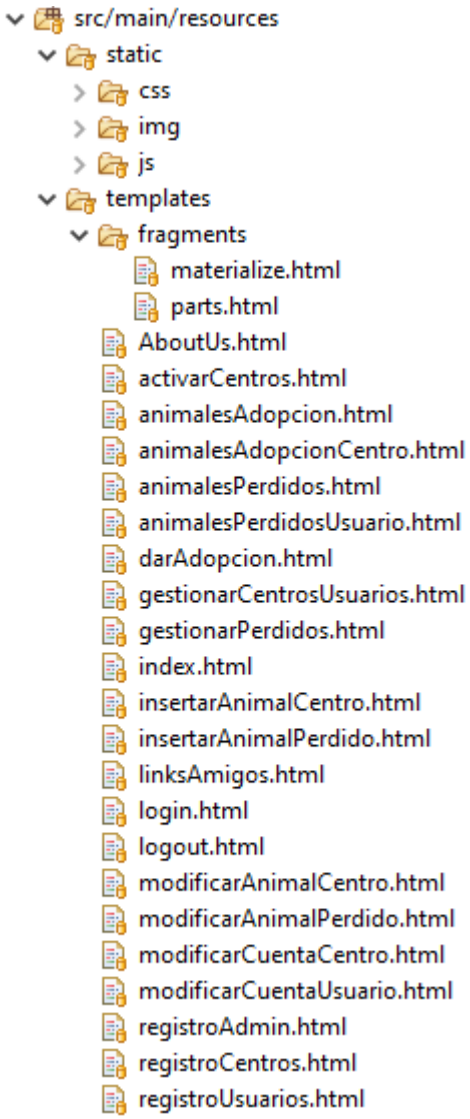
Diseño de la interfaz de usuario

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 Materialize. Para el responsive se ha usado este mismo framework.

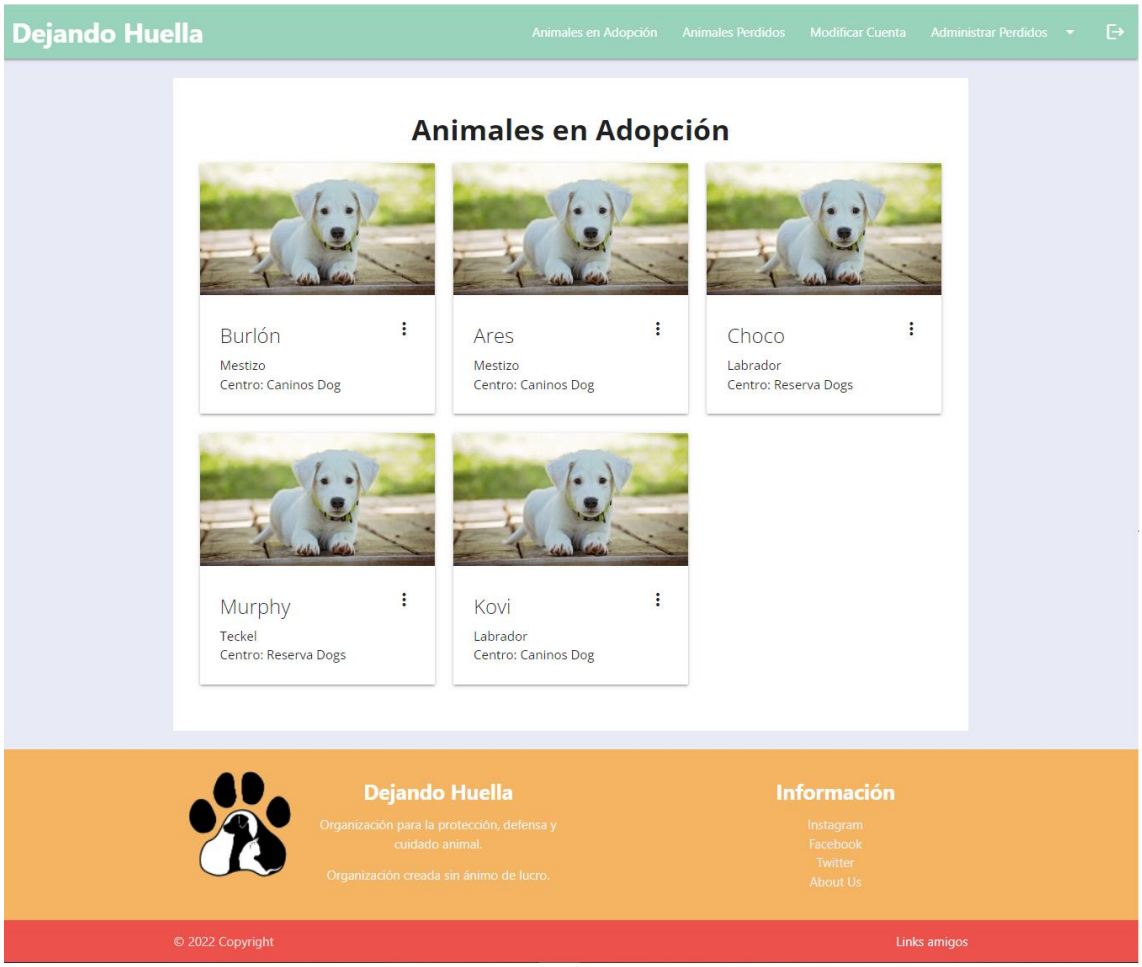
La distribución que se ha adoptado es la siguiente:

- Static: donde alojaremos las siguientes carpetas.
    - CSS: aquí estarán los archivos de diseño de la aplicación.
    - IMG: desde aquí se extraerán las imágenes fijas que se usen en la web.
    - JS: y desde aquí se extraerán los diferentes archivos JavaScript que se usen en la web.
- Templates: donde estarán nuestros archivos HTML.
    - Fragments: aquí estarán los archivos de Materialize y las partes de las vistas que se repitan, para manejar los cambios de manera más eficiente.

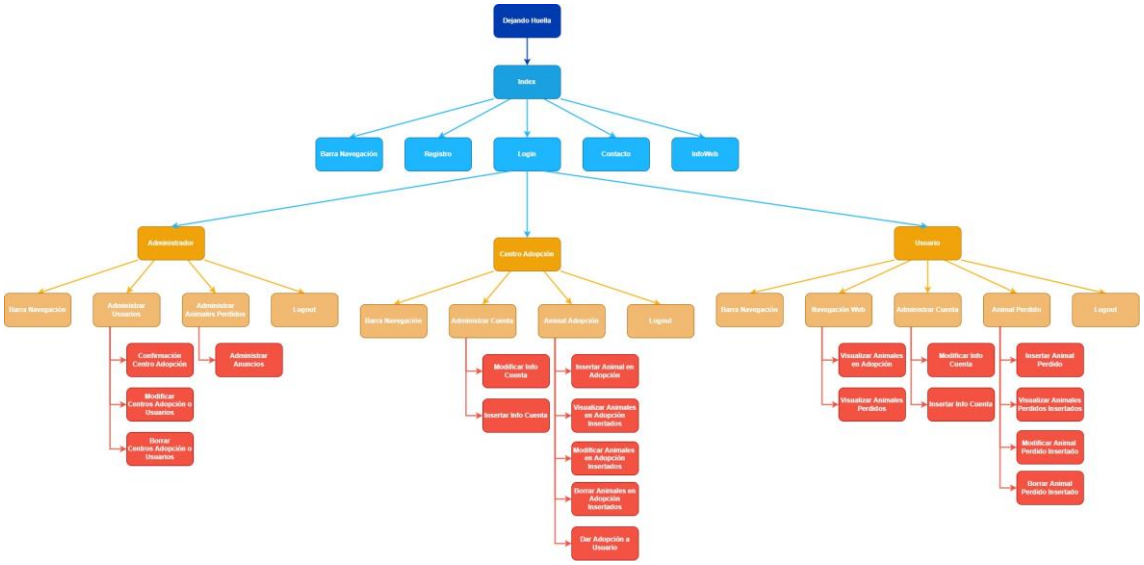


Los encargados de mostrar la información y las páginas van a ser los “layouts” que reciben módulos de componentes o implementan la lógica necesaria para transformar la información que proviene de los servicios en algo visual.

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.



Diseño del contenido

El contenido de la aplicación se encuentra recibido, en su mayor parte, por los usuarios, tanto los Usuarios Estándar como los Usuarios Centro Adopción, que aportan los inputs necesarios para que la aplicación tenga sentido.

Además de los inputs que aportan los usuarios Centro Adopción, juegan un papel importante en el diseño del contenido, por tanto, contar con un flujo de usuarios tanto Estándar como de Centros de Adopción es básico para obtener un correcto funcionamiento del sistema.

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

El contenido que ingresa directamente un usuario de Dejando Huella son los anuncios de animales perdidos o los anuncios de animales en adopción, los cuales hay que registrarlos 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 los Centros que se registren.
    - Ingresar en el sistema.
    - Visualizar la información personal.
    - Registrar Usuarios.

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 Materialize 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.
    - 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.

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 pruebas manuales del aplicativo.

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.