Creación del sitio web para el Museo de la Informática de la Escuela de Ingeniería Informática de Oviedo

GRADO EN INGENIERÍA INFORMÁTICA DEL SOFTWARE

Trabajo Fin de Grado

AUTOR

Mª Isabel Fernández Pérez

TUTOR

José Manuel Redondo López

Julio 2021

Teaching Innovation Project: PINN-19-A-029 (University of Oviedo)
This work has been published in [1] [2]

Esta versión de la plantilla para Trabajos de Fin de Grado ha sido posible gracias a la donación de la ex-alumna Elena Allegue González de su documentación de Trabajo de Fin de Grado, que ha servido como base para elaborar esta versión. Aquí podréis encontrar todos los títulos y subtítulos de las secciones, pero las explicaciones se mantendrán en la versión *Word* de la plantilla (se proporciona una versión PDF de la misma para facilitar el acceso a las mismas). No obstante, del trabajo de Elena se han conservado ejemplos de como hacer elementos clave como imágenes, tablas, etc.

Desarrollar una versión *Latex* de la plantilla desde cero es una trabajo bastante largo, pero gracias al trabajo de Elena se ha podido equiparar esta versión con las de *Word* mucho más rápidamente.

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

¿Qué es este trabajo?

1.1. Resumen

Este proyecto consiste en el desarrollo del sitio web para el Museo de la Escuela de Ingeniería Informática de Oviedo, en el que se exponen antiguos componentes hardware, principalmente CPUs.

El usuario podrá navegar por los diferentes periodos históricos en los que se agrupan los componentes, conocer efemérides tecnológicas de la época y otras curiosidades. De cada pieza podrá ver características, sistemas famosos que la utilizaban, e imágenes tanto del componente como de dichos sistemas famosos.

Además, el administrador podrá añadir, editar y eliminar los periodos y componentes que se mostrarán en la web del museo.

1.2. Palabras Clave

Museo, informática, sitio web, componentes, hardware, CPU, Oviedo, Escuela de Ingeniería Informática.

1.3. Abstract

The aim of this project is to develop the Computer Museum's website for the Computer Science School, to exhibit old hardware, mainly CPUs.

The user will be able to visit the different historical periods in which components are grouped, to know technological ephemerides of that time and other curiosities. For each piece, the user will also be able to see its characteristics, famous systems that used it and images of the piece and the famous systems.

In addition, the administrator will be able to add, update and delete the periods and components to be displayed in the museum's website.

1.4. Keywords

Museum, Computer Science, website, components, hardware, CPU, Oviedo, School of Computer Science.

Capítulo 2

PSI: PLANIFICACIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE PLANIFICACIÓN

PSI

2.1. PSI 1: INICIO DEL PLAN DE SISTEMAS DE INFORMACIÓN

2.1.1. PSI 1.1: Análisis de la Necesidad del PSI

El tutor de este trabajo de fin de grado, José Manuel Redondo, ha propuesto el desarrollo de una aplicación web para el Museo de la Informática de Asturias, que contenga toda la información disponible sobre los componentes del museo y la muestre de forma ordenada para que las personas interesadas puedan acceder a ella fácilmente. El sistema será gestionado directamente por el tutor del trabajo.

El sistema debe identificar cada componente y mostrar la información disponible del mismo,. El software permitirá añadir la información de las nuevas piezas que puedan ser incluidas en la exposición en un futuro gracias a donaciones o compras. Los componentes serán ordenados según su tipo y la época a la que pertenecen.

2.1.2. PSI 1.2: Identificación del Alcance del PSI

Actualmente los carteles informativos sobre las piezas del museo se encuentran expuestos en la Escuela de Ingeniería Informática. Los objetivos de este proyecto son los siguientes:

- Recopilar los datos disponibles de las piezas que se encuentran actualmente en el Museo e introducirlos en una base de datos.
- Mostrar una linea temporal con los diferentes periodos a los que pertenecen los componentes del Museo.
- Permitir acceder a cada periodo para ver los componentes del mismo.
- Organizar las diferentes piezas en función de su tipo y de la familia de la que forman parte.
- Presentar la información disponible de cada pieza, así como imágenes de la misma y otras curiosidades.

En definitiva, estos objetivos se pueden resumir en:

- Permitir a los usuarios visitar el Museo de la Informática de forma online, ofreciendo la misma información que se encuentra disponible en la exposición física.
- Facilitar al administrador la inserción de nuevos periodos y componentes.

2.1.3. PSI 1.3: Determinación de Responsables

- **El proyectante** se encargará del desarrollo del software descrito y de realizar la carga de los datos disponibles a la base de datos correspondientes.

- **El tutor del proyecto** se encargará de la supervisión de las fases del proyecto y de su validación.
- **Una serie de usuarios escogidos aleatoriamente** realizará pruebas del software para comprobar su correcto funcionamiento.

2.2. PSI 2: DEFINICIÓN Y ORGANIZACIÓN DEL PSI

2.2.1. PSI 2.1: Especificación del Ámbito y Alcance

2.2.2. PSI 2.2: Organización del PSI

2.3. PSI 3: ESTUDIO DE LA INFORMACIÓN RELEVANTE

2.3.1. PSI 3.1: Selección y Análisis de Antecedentes

Capítulo 3

PSI 7: DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA

FASE DE PLANIFICACIÓN

PSI

3.1. PSI 7.1: Identificación de las Necesidades de Infraestructura Tecnológica

3.2. PSI 7.2: Selección de la Arquitectura Tecnológica

Capítulo 4

ESTUDIO DE VIABILIDAD DEL SISTEMA

FASE DE DESARROLLO

EVS

4.1. EVS 4, 5, 6: ESTUDIO Y VALORACIÓN DE ALTERNATIVAS DE SOLUCIÓN. SELECCIÓN DE ALTERNATIVA FINAL

4.1.1. Evaluación de alternativas de desarrollo

4.1.1.1. JavaScript y Node.js

JavaScript es uno de los lenguajes más populares actualmente. Está basado en el estándar ECMAScript. Se trata un lenguaje interpretado, se compila en tiempo de ejecución. Es orientado a objetos, débilmente tipado y dinámico[3].

Node.js es un entorno de ejecución de JavaScript orientado a eventos asíncronos, en el que no hace falta ultilizar hilos. Utiliza un modelo de entrada y salida sin bloqueo, lo que asegura un rendimiento más eficiente de la aplicación y evita que se produzca una gran sobrecarga del lado del servidor. Por ello, es muy apropiado para desarrollar sistemas escalables[4]. Además, puede ser utilizado tanto en el lado del cliente como en el servidor, por lo que no se necesitaría una tecnología adicional para el back-end.

Esta fue la primera opción barajada, ya que había utilizado anteriormente estas tecnologías y podría aprovechar este proyecto para profundizar en su aprendizaje.

Figura 4.1: Logos de JavaScript y Node.js

4.1.1.2. Angular, TypeScript y PHP

La otra opción considerada fue Angular y TypeScript, debido a su popularidad. No había trabajado con ellas antes, y esta sería una buena oportunidad para conocerlas.

Angular es un framework desarrollado en TypeScript y utilizado habitualmente para crear aplicaciones de una sola página. Se basa en la utilización de componentes web reutilizables para crear aplicaciones web fácilmente escalables. Angular extiende la sintaxis de HTML y actualiza automáticamente el árbol DOM cuando el estado de un componente cambia. Cuenta con gran cantidad de librerías y es uno de los frameworks más utilizados en la industria actual[5].

TypeScript es un lenguaje de programación que extiende JavaScript añadiendo la definición de tipos estáticos. Al compilarlo se transforma en código JavaScript siguiendo todos los estándares, y puede ejecutarse en cualquier lugar que ejecute JavaScript[6].

En este caso, Angular y TypeScript son ambas tecnologías de front-end, por tanto necesitamos una tercera tecnología para el back-end de esta aplicación. Para ello consideré

como opción PHP, lenguaje que se ejecuta en el servidor y envía el resultado generado al cliente, y que es otra de las tecnologías más reconocidas y usadas en el desarrollo web actualmente y desde su creación.

Figura 4.2: Logos de Angular, TypeScript y PHP

Ambas opciones son de código abierto, lo que me parece un punto positivo ya que, gracias a la colaboración de la comunidad, se consigue una alta calidad en el software.

Finalmente, me decidí por Angular, TypeScript y PHP, principalmente por la razón de profundizar en el aprendizaje de estas tecnologías tan importantes actualmente en el desarrollo de aplicaciones web.

4.1.2. Evaluación de alternativas de gestor de bases de datos

4.1.2.1. MySQL

MySQL es un SGBD relacional de código abierto con un modelo cliente-servidor. Ha sido la única opción considerada al tratarse de la base de datos relacional que es comúnmente utilizada con Angular, y no se ha encontrado ninguna necesidad o ventaja de usar un sistema no relacional.

Figura 4.3: Logo de MySQL

Capítulo 5

PLANIFICACIÓN Y GESTIÓN DEL TFG

5.1. PLANIFICACIÓN DEL PROYECTO

5.1.1. Identificación de Interesados

5.1.2. OBS y PBS

5.1.3. Planificación Inicial. WBS

5.1.4. Riesgos

5.1.4.1. Plan de Gestión de Riesgos

5.1.4.2. Identificación de Riesgos

5.1.4.3. Registro de Riesgos

5.1.5. Presupuesto Inicial

5.1.5.1. Presupuesto de Costes

5.1.5.2. Presupuesto de Cliente

5.2. EJECUCIÓN DEL PROYECTO

5.2.1. Plan Seguimiento de Planificación

5.2.2. Bitácora de Incidencias del Proyecto

5.2.3. Riesgos

5.3. CIERRE DEL PROYECTO

5.3.1. Planificación Final

5.3.2. Informe Final de Riesgos

5.3.3. Presupuesto Final de Costes

5.3.4. Informe de Lecciones Aprendidas

Capítulo 6

ANÁLISIS DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

ASI

6.1. ASI 1: DEFINICIÓN DEL SISTEMA

6.1.1. Determinación del Alcance del Sistema

6.2. ASI 2: ESTABLECIMIENTO DE REQUISITOS

6.2.1. Obtención de los Requisitos del Sistema

6.2.1.1. Requisitos de interfaces externas

Interfaces de usuario

RIU-1. El sistema será accesible desde cualquier dispositivo que cuente con conexión a internet y un navegador web.

RIU-2. El sistema estará disponible en diferentes idiomas.

RIU-2.1. Español

RIU-2.2. Inglés

RIU-3. El sistema deberá ser accesible para todos los usuarios a través de los navegadores más comunes.

RIU-3.1. Google Chrome

RIU-3.2. Mozilla Firefox

RIU-3.3. Microsoft Edge

RIU-4. El usuario podrá utilizar todas las funcionalidades desarrolladas de la aplicación sin inconvenientes.

RIU-5. El usuario no necesitará de conocimientos tecnológicos avanzados.

Interfaces hardware

RIH-1. El sistema dispondrá de una base de datos para almacenar la información necesaria.

Interfaces de comunicaciones

RIC-1. El sistema contendrá enlaces a diferentes sitios web.

RIC-2. El sistema mostrará por defecto enlaces a los siguientes sitios web.

RIC-2.1. Twitter oficial de la Escuela de Ingeniería Informática

RIC-2.2. Página web de la Escuela de Ingeniería Informática

RIC-2.3. Página web de la Universidad de Oviedo

6.2.1.2. Requisitos funcionales

RF-1.

6.2.1.3. Requisitos de rendimiento

6.2.1.4. Requisitos lógicos de BD

6.2.1.5. Requisitos de desarrollo

6.2.1.6. Restricciones de diseño

6.2.1.7. Atributos del sistema

6.2.2. Identificación de Actores del Sistema

6.2.2.1. Usuario administrador

Actor que interactúa con el sistema. Es responsable de gestionar el sistema y su mantenimiento. Es el único actor con acceso a la base de datos del sistema y capacidad de modificarla. Debe tener amplios conocimientos sobre el sistema.

6.2.2.2. Usuario estándar

Actor que interactúa con el sistema. Tiene acceso de lectura a toda la aplicación web, exceptuando la parte dedicada al mantenimiento. Solo debe tener un conocimiento básico para navegar por internet.

6.2.3. Especificación de Casos de Uso

Ejemplo de tabla para especificación de casos de uso

Tabla 6.1: Especificación Caso de Uso 1

Nombre del caso de uso
Registro
Descripción
Un usuario no registrado debe poder registrarse en el sistema mediante su cuenta de Google, lo que hará que automáticamente se inicie sesión en la aplicación.

6.3. ASI 3: IDENTIFICACIÓN DE SUBSISTEMAS DE ANÁLISIS

6.3.1. Descripción de los Subsistemas

6.3.2. Descripción de los Interfaces entre Subsistemas

6.4. ASI 4: ANÁLISIS DE LOS CASOS DE USO

6.4.1. Caso de Uso 1

Ejemplo de tabla para análisis de casos de usos

Tabla 6.2: Análisis del Caso de Uso 1

Registro	
Precondiciones	El usuario no debe haber iniciado sesión nunca.
Postcondiciones	-
Actores	Usuario no registrado
Descripción	El usuario accederá a la pantalla principal de la aplicación cuando no se está registrado, y seleccionará el botón de inicio de sesión, que, al ser la primera vez, registrará.Seleccionará la cuenta de Google con la que desee registrarse y el sistema completará el resto del registro.
Escenarios Secundarios	El usuario no tiene cuenta de Google: escenario que puede ser posible si accede a la aplicación a través del App Market. En este caso se le solicitará crear una cuenta.

6.4.2. Caso de Uso 2

6.5. ASI 5: ANÁLISIS DE CLASES

6.5.1. Diagrama de Clases

6.5.2. Descripción de las Clases

6.6. ASI 8: DEFINICIÓN DE INTERFACES DE USUARIO

6.6.1. Descripción de la Interfaz

6.6.1.1. Museo

Inicio

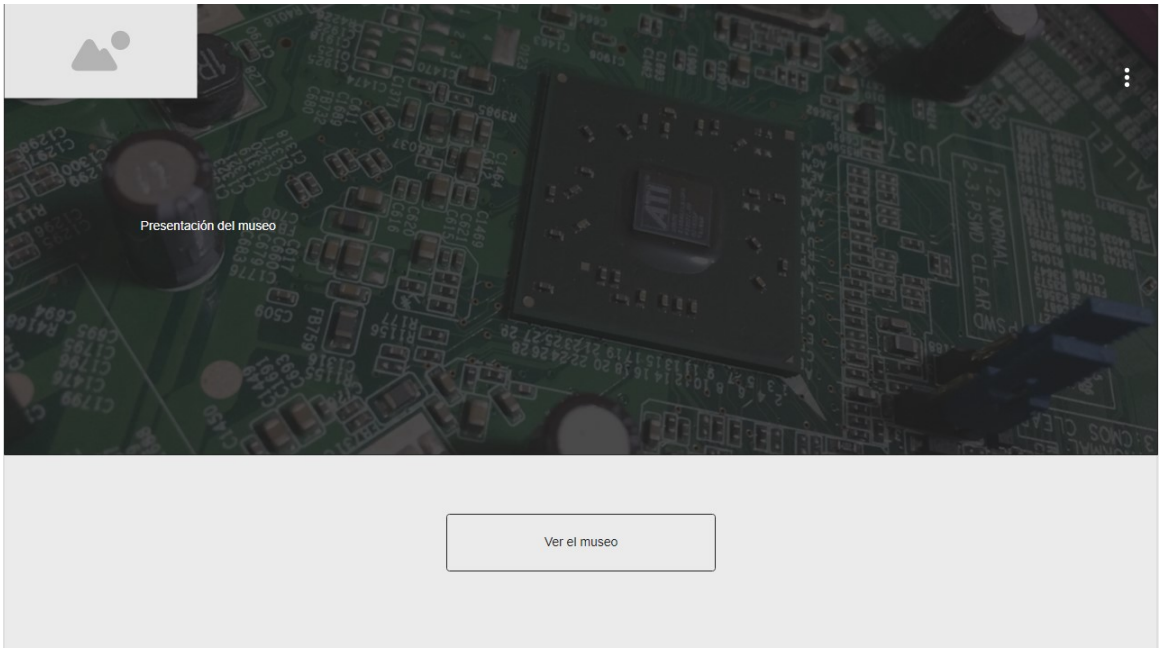

Figura 6.1: Prototipo: Página de inicio

Vista general del museo

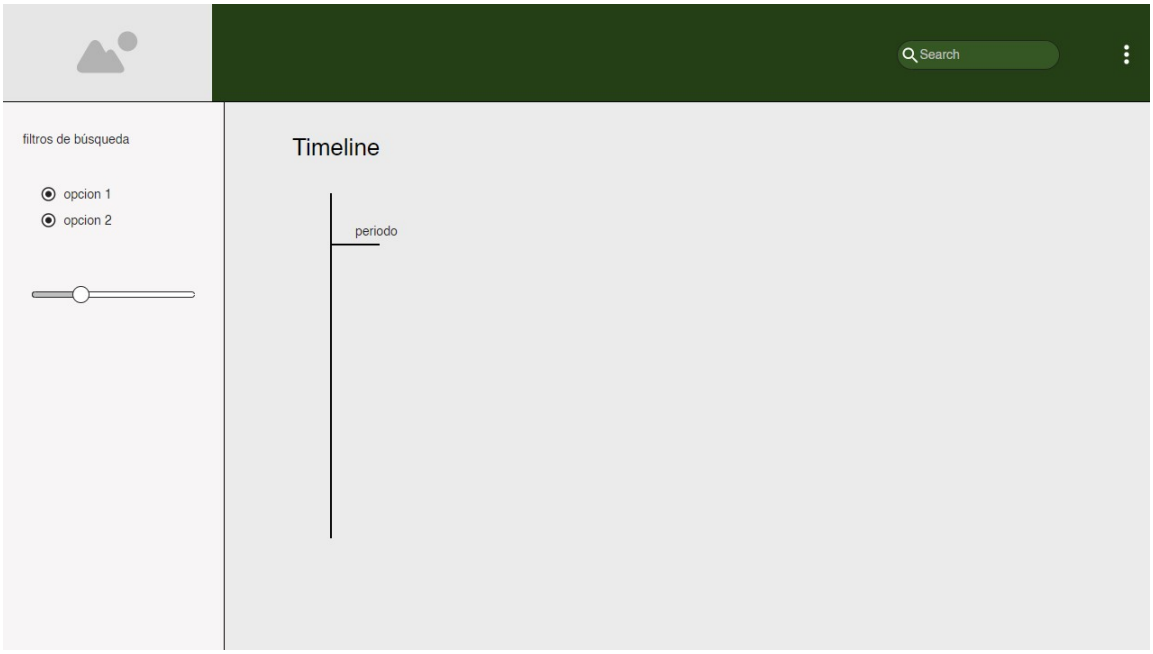

Figura 6.2: Prototipo: Página de la vista general del museo

Detalles del periodo

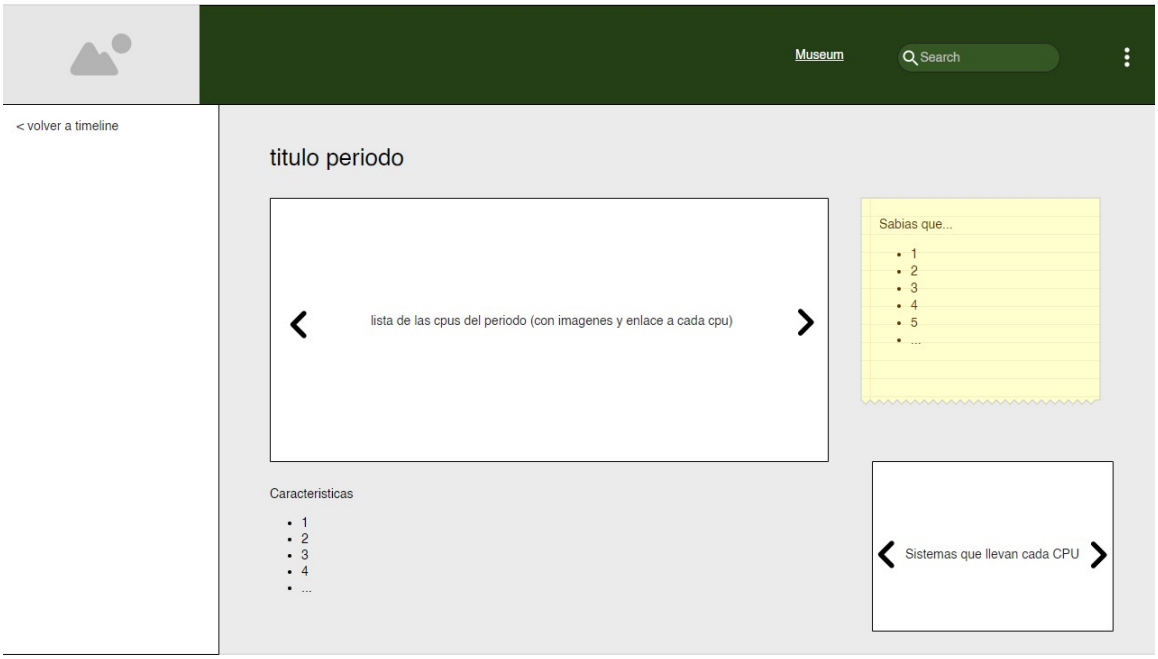

Figura 6.3: Prototipo: Página de detalles del periodo

Detalles del componente

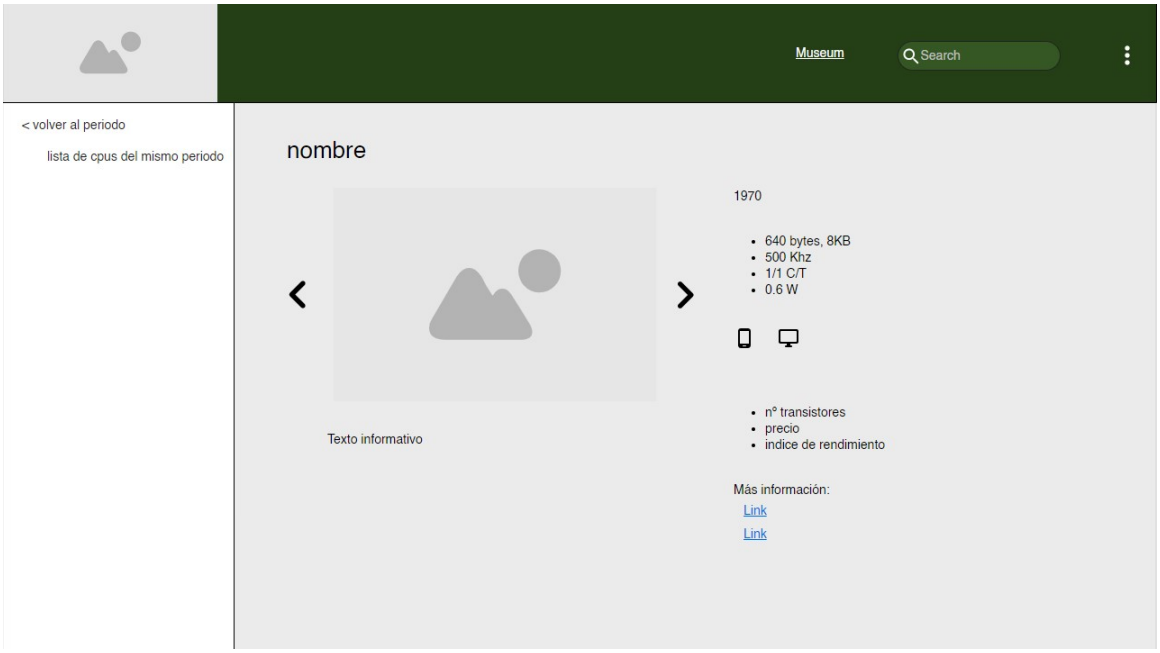

Figura 6.4: Prototipo: Página de detalles del componente

6.6.1.2. Administración del museo

Iniciar sesión

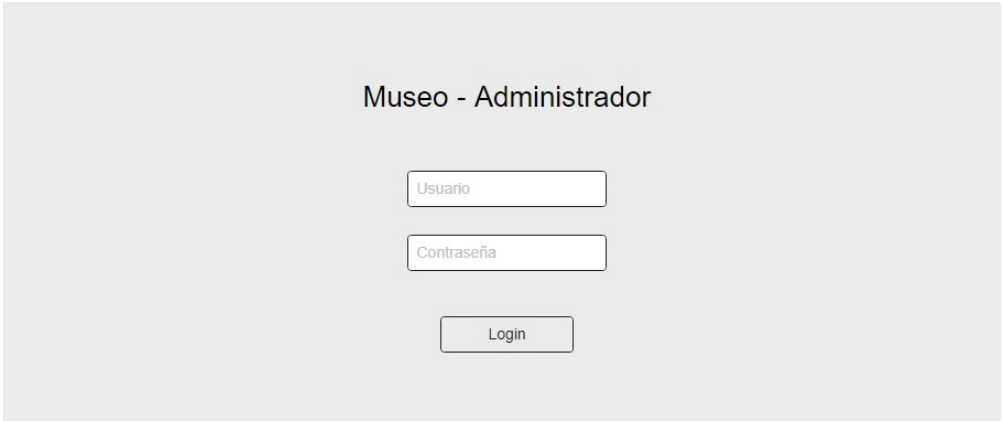

Figura 6.5: Prototipo: Página de inicio de sesión

Añadir periodo

Figura 6.6: Prototipo: Formulario para añadir un periodo

Editar periodo

Figura 6.7: Prototipo: Formulario para editar un periodo

Añadir componente

Figura 6.8: Prototipo: Formulario para añadir un componente

Editar componente

Figura 6.9: Prototipo: Formulario para editar un componente

6.6.2. Definición del aspecto de la interfaz

6.6.3. Descripción del Comportamiento de la Interfaz

6.6.4. Diagrama de Navegabilidad

6.7. ASI 10: ESPECIFICACIÓN DEL PLAN DE PRUEBAS

Capítulo 7

DISEÑO DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

DSI

7.1. DSI 3: DISEÑO DE CASOS DE USO REALES

7.1.1. Caso de Uso 1.1

7.1.1.1. Diagramas de Interacción (Comunicación y Secuencia)

7.1.1.2. Diagramas de Estados de las Clases

7.1.1.3. Diagramas de Actividades

7.1.2. Caso de Uso 1.2

7.2. DSI 4: DISEÑO DE CLASES

7.2.1. Diagrama de Clases

7.3. DSI 5: DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SISTEMA

7.3.1. DSI 5.1 Diseño de Módulos del Sistema

7.3.2. DSI 5.2 Diseño de Comunicaciones entre Módulos

7.3.3. DSI 5.3 Revisión de la Interfaz de Usuario

7.4. DSI 6: DISEÑO FÍSICO DE DATOS

7.4.1. Descripción del SGBD Usado

Se ha creado una base de datos relacional, utilizando MySQL 8 como sistema gestor de bases de datos, debido a su gran popularidad en todo el mundo y en el desarrollo de aplicaciones con Angular y PHP.

La base de datos creada es museo-eii, y se compone de cinco tablas:

- periods: alamcena toda la información relativa a un periodo.
- components: contiene los datos que serían comunes a cualquier tipo de componente idependientemente de su tipo, como nombre, año de salida, precio, ect.
- components_images: asigna el conjunto de imágenes de cada componente.
- cpus: almacena los datos específicos de una CPU, como la memoria RAM, la velocidad de reloj o el tamaño de palabra. El identificador de esta tabla referencia al elemento correspondiente de la tabla components, simulando así una herencia en la base de datos. Esta herencia simulada hace que la base de datos sea fácilmente ampliable si se añade un tipo de componente que no sea una CPU, ya que bastaría con añadir una nueva tabla con los campos necesarios que también referencie a components.
- administrator: almacena los datos de inicio de sesión del administrador (email y contraseña cifrada). En caso de que hubiera más usuarios que tuviesen que iniciar sesión se habría creado una base de datos exclusiva para almacenarlos, pero como en este caso solo existe un administrador, he decidido incluir esta tabla en la base de datos existente para este proyecto.

7.4.2. Integración del SGBD en Nuestro Sistema

Para integrar el sistema con la base de datos se han creado tres servicios en angular: uno para periodos, otro para componentes y el último para el usuario. Estos tres servicios utilizan la librería HttpClient de Angular, que permite realizar peticiones HTTP para obtener o enviar datos al lado del servidor, donde se encuentran los archivos PHP que contienen las consultas que han de realizarse sobre la base de datos.

7.4.3. Diagrama E–R

A continuación se muestra el diagrama entidad-relación de la base de datos del sistema, museo-eii:

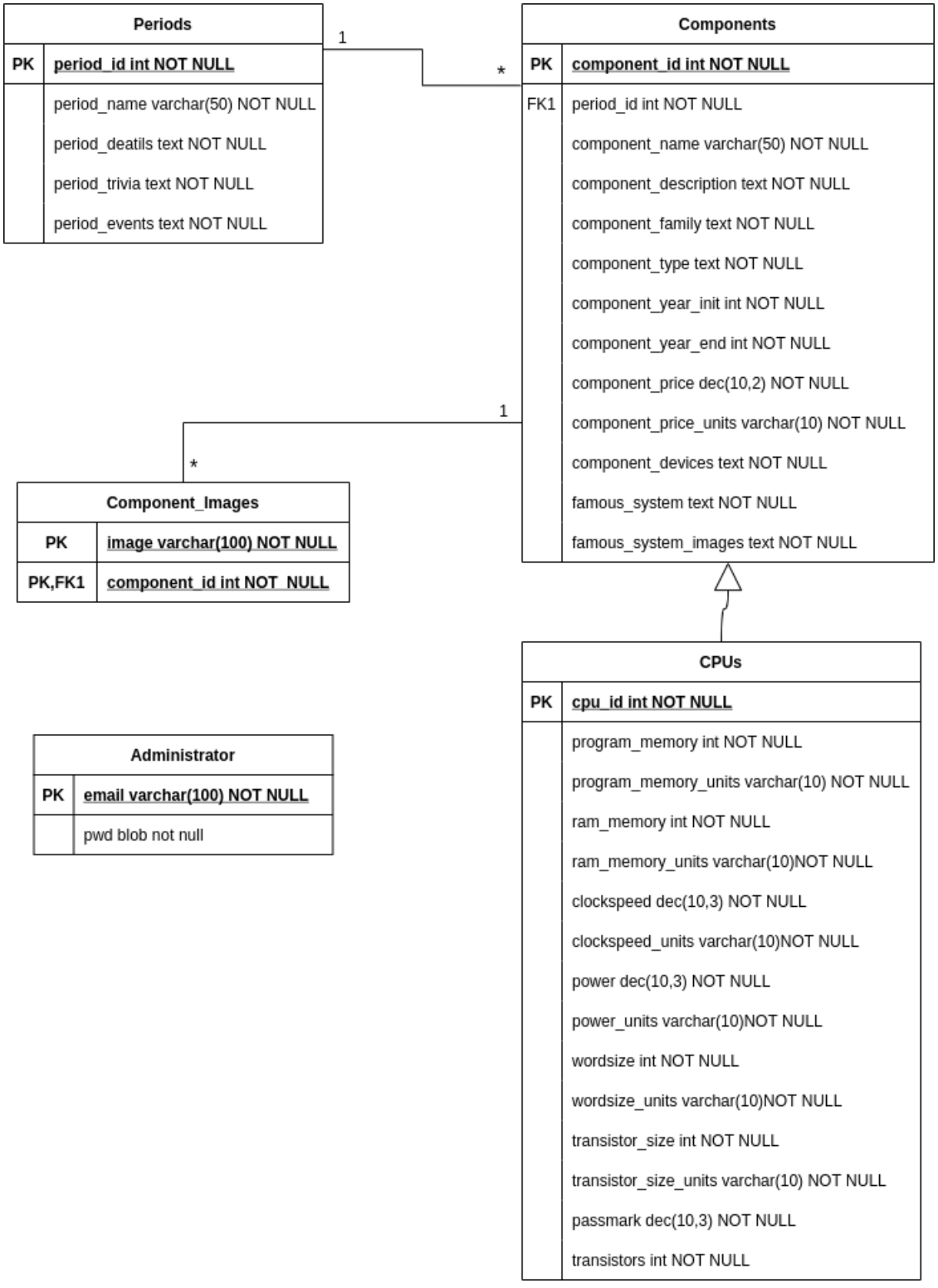

Figura 7.1: Diagrama Entidad-Relación de la base de datos creada

7.5. DSI 9: DISEÑO DE LA MIGRACIÓN Y CARGA INICIAL DE DATOS

7.6. DSI 10: ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS

7.6.1. Pruebas Unitarias

7.6.2. Pruebas de Integración y del Sistema

7.6.3. Pruebas de Usabilidad y Accesibilidad

7.6.3.1. Diseño de Cuestionarios

7.6.3.2. Actividades de las Pruebas de Usabilidad

7.6.4. Pruebas de Accesibilidad

7.6.5. Pruebas de Rendimiento

Capítulo 8

CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

CSI

8.1. CSI 1: PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONSTRUCCIÓN

8.1.1. Estándares y normas seguidos

8.1.1.1. Angular Style Guide

La guía de estilos de Angular[7] es un conjunto de recomendaciones sobre la sintaxis, estructura y convenciones de código en proyectos de Angular.

8.1.1.2. HTML5

HTML5 es la versión más reciente y la actualmente usada de HTML, y está estandarizada por el W3C (World Wide Web Consortium).

8.1.1.3. CSS

Hojas de estilos estandarizadas por el W3C.

8.1.1.4. PHP Code Style Guide

La guía de estilos de PHP[8] contiene normas de código y buenas prácticas.

8.1.2. Lenguajes de programación

8.1.2.1. TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft. Extiende JavaScript añadiendo la definición de tipos estáticos.

8.1.2.2. HTML

HTML (HyperText Markup Language) es un lenguaje de marcado utilizado en la elaboración de páginas web.

8.1.2.3. CSS

CSS (Cascading Style Sheets) es un lenguaje de diseño gráfico que permite modificar la presentación de los elementos definidos en los documentos HTML.

8.1.2.4. PHP

PHP es un lenguaje de programación utilizado en el desarrollo web que es procesado en el lado del servidor.

8.1.2.5. SQL

SQL (Structured Query Language) es un lenguaje de consultas utilizado para leer, insertar, actualizar o eliminar datos de la base de datos relacional utilizada.

8.1.3. Herramientas y programas usados para el desarrollo

8.1.3.1. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, gratuito y de código abierto. Tiene soporte integrado para TypeScript y Node.js, extensiones para otros lenguajes como PHP. También cuenta con soporte para depuración, control integrado de Git e *IntelliSense*, una función de autocompletado de código[9].

Figura 8.1: Logo de Visual Studio Code

8.1.3.2. XAMPP

XAMPP es una distribución de Apache gratuita que contiene MariaDB, PHP y Perl[10].

Figura 8.2: Logo de XAMPP

8.1.3.3. Git

Git es un software de control de versiones gratuito y de código abierto, diseñado para gestionar los cambios de un repositorio[11].

Figura 8.3: Logo de Git

8.2. CSI 2: GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES Y PROCEDIMIENTOS

Ejemplos de tablas descripción de clases

Tabla 8.1: Descripción de diseño de LoginScreen

LoginScreen	
Descripción	
Es la encargada de las acciones y la renderización de la pantalla de inicio de sesión.	
Atributos propuestos	
-	
Métodos propuestos	
signInWithGoogle	Hace una llamada al objeto Fire para el inicio de sesión con Firebase authentication mediante una cuenta de Google.
render	

Tabla 8.2: Descripción de diseño de HomeScreen

HomeScreen	
Descripción	
Es la encargada de las acciones y la renderización de la pantalla de emergencia.	
Atributos propuestos	
-	
Métodos propuestos	
componentWillMount	
emergencyCalling	Es el método encargado de redirigir la aplicación hacia el marcador con el 112 marcado.
warnProtectors	[Falta implementar] Es el encargado de generar un mensaje de aviso a los protectores creando notificaciones push.
render	

8.3. CSI 3: EJECUCIÓN DE LAS PRUEBAS UNITARIAS

8.4. CSI 4: EJECUCIÓN DE LAS PRUEBAS DE INTEGRACIÓN

8.5. CSI 5: EJECUCIÓN DE LAS PRUEBAS DEL SISTEMA

8.5.1. Prueba de Usabilidad

8.5.2. Pruebas de Accesibilidad

8.5.2.1. Revisión Preliminar

8.5.2.2. Evaluación de Conformidad

8.5.2.3. Checklist del WCAG 2.1

8.5.2.4. Accesibilidad con Dispositivos Móviles

8.6. CSI 6: ELABORACIÓN DE LOS MANUALES DE USUARIO

8.6.1. Manual de Instalación

8.6.2. Manual de Ejecución

8.6.3. Manual de Usuario

8.6.4. Manual del Programador

8.7. CSI 8: CONSTRUCCIÓN DE LOS COMPONENTES Y PROCEDIMIENTOS DE MIGRACIÓN Y CARGA INICIAL DE DATOS

Capítulo 9

IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA

FASE DE DESARROLLO

IAS

9.1. IAS 1: ESTABLECIMIENTO DEL PLAN DE IMPLANTACIÓN

9.2. IAS 4: CARGA DE DATOS AL ENTORNO DE OPERACIÓN

9.3. IAS 5: PRUEBAS DE IMPLANTACIÓN DEL SISTEMA

9.4. IAS 7: PREPARACIÓN DEL MANTENIMIENTO DEL SISTEMA

9.5. IAS 8: ESTABLECIMIENTO DEL ACUERDO DE NIVEL DE SERVICIO

9.6. IAS 9–10: PRESENTACIÓN Y APROBACIÓN DEL SISTEMA Y PASO A PRODUCCIÓN

Capítulo 10

CONCLUSIONES Y AMPLIACIONES

10.1. CONCLUSIONES

10.2. AMPLIACIONES

ANEXOS

PLAN DE GESTIÓN DE RIESGOS

CONTENIDO ENTREGADO EN LOS ANEXOS

Contenidos

Ejemplo de como especificar los contenidos entregados

Además de este documento, se hace entrega de una carpeta comprimida ".zip" en la que ahora se describirán sus contenidos. Se estructurará también la organización del código fuente.

- **Planificación_TFG.mpp** -¿Archivo de Microsoft Project que contiene la planificación del proyecto entera.
- **Presupuesto-GuardMe_TFG.xlsx** -¿Archivo Microsoft Excel que contiene los cálculos del presupuesto del proyecto.
- **Diagramas** -¿Carpeta que contiene todos los diagramas utilizados en este documento.
  - *Diagrama_de_paquetes.png*
  - *Diagrama_firestore.png*
  - *Diagrama_navegabilidad.png*
  - *Diagrama_secuencia_enviar.png*
  - *Diagrama_secuencia_visualizar.png*
  - *Diagrama_UML-Diseño.png*
  - *Diagrama_UML-Analisis.png*
- **TFG_codigo.zip** -¿Carpeta comprimida con todo el código fuente.

Ahora se mostrará el contenido de dicha carpeta comprimida que contiene todo el código fuente de la aplicación la cual esta dividida a su vez en dos carpetas:

AuthServerGuardMe

Contiene el código que se aloja en *Heroku* para darle funcionalidad al servidor. La clase principal es la llamada `mainAuthServer.js`.

GuardMe

Contiene el código fuente de la aplicación y se compone de las siguientes carpetas:

- **assets** -¿Carpeta que contiene los elementos gráficos usados en la aplicación. Se subdivide en una carpeta llamada *images* que contiene todas las imagenes utilizadas para la construcción de la aplicación.
- **components** -¿Carpeta que contiene el código para todos los componentes creados.

- **constants** -¿Carpeta que contiene el código
- **docs** -¿Carpeta que contiene los archivos html generados por JSDoc.
- **files** -¿Carpeta en la que se encuentras los futuros archivos de Términos y Condiciones y Política de Privacidad entre otros.
- **modules_LICENSES** -¿Carpeta que contiene una por una todas las licencias de las librerías utilizadas en el desarrollo.
- **navigation** -¿Carpeta que contiene las clases relativas a la navegación de la aplicación.
- **objects** -¿Carpeta que contiene los objetos utilizados en el desarrollo que en este caso ha sido solo Fire.js.
- **screens** -¿Carpeta que contiene todas las pantallas, agrupadas a su vez en subcarpetas que identifican la pantalla sobre la que están relacionadas.
- **styles** -¿Carpeta que contiene todos los estilos de las pantallas, agrupadas a su vez en subcarpetas que siguen la misma estructura que *screens*.
- **App.js** -¿Clase principal y encargada de que comience la aplicación entera.
- **LICENSE** -¿Licencia sobre el código fuente.
- **README.md** -¿Archivo con la descripción del proyecto para la documentación y el repositorio de GitHub.
- **package.json** -¿Archivo que contiene las librerías utilizadas en el proyecto.
- **app.json** -¿Archivo que contiene la configuración de la aplicación.
- **configJSDoc.json** -¿Archivo de configuración para la creación de documentación por parte de JSDoc.
- **Otros archivos** -¿Los demás archivos no son relevantes ya que muchos se generan por defecto y los demás son configuraciones propias de expo.

Bibliografía

[1] Jose Manuel Redondo, "Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo." https://www.researchgate.net/publication/327882831_Plantilla_de_Proyectos_de_Fin_de_Carrera_de_la_Escuela_de_Informatica_de_Oviedo, 2019. Online; accessed 13 Jul 2020.

[2] Jose Manuel Redondo, "Creación y evaluación de plantillas para trabajos de fin de grado como buena práctica docente," *Revista de Innovación y Buenas Prácticas Docentes*, vol. pp, no. pp, p. pp, 2020.

[3] MDN contributors, "JavaScript." https://developer.mozilla.org/es/docs/Web/JavaScript, 2020. Online; accessed 10 Oct 2021.

[4] Jesús Lucas, "Qué es NodeJS y para qué sirve." https://openwebinars.net/blog/que-es-nodejs/, 2019. Online; accessed 10 Oct 2021.

[5] "What is Angular?." https://angular.io/guide/what-is-angular. Online; accessed 10 Oct 2021.

[6] "Typed JavaScript at Any Scale.." https://www.typescriptlang.org/. Online; accessed 10 Oct 2021.

[7] "Angular Style Guide." https://angular.io/guide/styleguide. Online; accessed 13 Abr 2022.

[8] Ryan Sechrest, "PHP Code Style Guide." https://gist.github.com/ryansechrest/8138375. Online; accessed 13 Abr 2022.

[9] "Visual Studio Code." https://code.visualstudio.com/docs. Online; accessed 13 Abr 2022.

[10] "XAMPP." https://www.apachefriends.org/es/index.html. Online; accessed 13 Abr 2022.

[11] "Git." https://git-scm.com/. Online; accessed 13 Abr 2022.

[12] J. M. Requena, "El consejero de Universidad pide apostar por la innovación y generar conocimiento." https://www.lne.es/asturias/2019/08/13/consejero-universidad-pide-apostar-innovacion/2514937.html, 2019. Online; accessed 13 Jul 2020.