



# **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**

M<sup>a</sup> Isabel Fernández Pérez

**TUTOR**

José Manuel Redondo López

Julio 2021

Copyright (C) 2020 **ELENA ALLEGUE GONZÁLEZ, JOSÉ MANUEL REDONDO LÓPEZ**

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

<b>1. ¿Qué es este trabajo?</b>	<b>10</b>
1.1. Resumen . . . . .	10
1.2. Palabras Clave . . . . .	11
1.3. Abstract . . . . .	12
1.4. Keywords . . . . .	13
<b>2. PSI: PLANIFICACIÓN DEL SISTEMA DE INFORMACIÓN</b>	<b>14</b>
2.1. PSI 1: INICIO DEL PLAN DE SISTEMAS DE INFORMACIÓN . . . . .	15
2.1.1. PSI 1.1: Análisis de la Necesidad del PSI . . . . .	15
2.1.2. PSI 1.2: Identificación del Alcance del PSI . . . . .	15
2.1.3. PSI 1.3: Determinación de Responsables . . . . .	15
2.2. PSI 7: DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA . . . . .	17
2.2.1. PSI 7.2: Selección de la Arquitectura Tecnológica . . . . .	17
<b>3. ESTUDIO DE VIABILIDAD DEL SISTEMA</b>	<b>18</b>
3.1. EVS 4, 5, 6: ESTUDIO Y VALORACIÓN DE ALTERNATIVAS DE SOLUCIÓN. SELECCIÓN DE ALTERNATIVA FINAL . . . . .	19
3.1.1. Evaluación de alternativas de desarrollo . . . . .	19
3.1.2. Evaluación de alternativas de gestor de bases de datos . . . . .	20
<b>4. PLANIFICACIÓN Y GESTIÓN DEL TFG</b>	<b>21</b>
4.1. PLANIFICACIÓN DEL PROYECTO . . . . .	22
4.1.1. Identificación de Interesados . . . . .	22
4.1.2. OBS y PBS . . . . .	22
4.1.3. Planificación Inicial. WBS . . . . .	22
4.1.4. Riesgos . . . . .	22
4.1.5. Presupuesto Inicial . . . . .	22
4.2. EJECUCIÓN DEL PROYECTO . . . . .	23

4.2.1.	Plan Seguimiento de Planificación . . . . .	23
4.2.2.	Bitácora de Incidencias del Proyecto . . . . .	23
4.2.3.	Riesgos . . . . .	23
4.3.	CIERRE DEL PROYECTO . . . . .	24
4.3.1.	Planificación Final . . . . .	24
4.3.2.	Informe Final de Riesgos . . . . .	24
4.3.3.	Presupuesto Final de Costes . . . . .	24
4.3.4.	Informe de Lecciones Aprendidas . . . . .	24
<b>5.</b>	<b>ANÁLISIS DEL SISTEMA DE INFORMACIÓN</b>	<b>25</b>
5.1.	ASI 2: ESTABLECIMIENTO DE REQUISITOS . . . . .	26
5.1.1.	Obtención de los Requisitos del Sistema . . . . .	26
5.1.2.	Identificación de Actores del Sistema . . . . .	30
5.1.3.	Especificación de Casos de Uso . . . . .	30
5.2.	ASI 4: ANÁLISIS DE LOS CASOS DE USO . . . . .	34
5.2.1.	Caso de Uso 1 . . . . .	34
5.2.2.	Caso de Uso 2 . . . . .	34
5.2.3.	Caso de Uso 3 . . . . .	35
5.2.4.	Caso de Uso 4 . . . . .	35
5.2.5.	Caso de Uso 5 . . . . .	36
5.2.6.	Caso de Uso 6 . . . . .	36
5.2.7.	Caso de Uso 7 . . . . .	37
5.2.8.	Caso de Uso 8 . . . . .	37
5.2.9.	Caso de Uso 9 . . . . .	38
5.2.10.	Caso de Uso 10 . . . . .	38
5.2.11.	Caso de Uso 11 . . . . .	39
5.3.	ASI 5: ANÁLISIS DE CLASES . . . . .	40
5.3.1.	Diagrama de Clases . . . . .	40
5.3.2.	Descripción de las Clases . . . . .	41
5.4.	ASI 8: DEFINICIÓN DE INTERFACES DE USUARIO . . . . .	51
5.4.1.	Definición del aspecto de la interfaz . . . . .	51
5.4.2.	Diagrama de Navegabilidad . . . . .	56
5.5.	ASI 10: ESPECIFICACIÓN DEL PLAN DE PRUEBAS . . . . .	58
5.5.1.	Pruebas unitarias . . . . .	58
5.5.2.	Pruebas del sistema . . . . .	60
5.5.3.	Pruebas de usabilidad . . . . .	60

<b>6. DISEÑO DEL SISTEMA DE INFORMACIÓN</b>	<b>61</b>
6.1. DSI 4: DISEÑO DE CLASES . . . . .	62
6.1.1. Diagrama de Clases . . . . .	62
6.2. DSI 5: DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SISTEMA	64
6.2.1. DSI 5.1 Diseño de Módulos del Sistema . . . . .	64
6.2.2. DSI 5.2 Diseño de Comunicaciones entre Módulos . . . . .	64
6.2.3. DSI 5.3 Revisión de la Interfaz de Usuario . . . . .	64
6.3. DSI 6: DISEÑO FÍSICO DE DATOS . . . . .	71
6.3.1. Descripción del SGBD Usado . . . . .	71
6.3.2. Integración del SGBD en Nuestro Sistema . . . . .	71
6.3.3. Diagrama E-R . . . . .	71
6.4. DSI 9: DISEÑO DE LA MIGRACIÓN Y CARGA INICIAL DE DATOS . .	73
6.5. DSI 10: ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS . . . .	74
6.5.1. Pruebas Unitarias . . . . .	74
6.5.2. Pruebas de Integración y del Sistema . . . . .	74
6.5.3. Pruebas de Usabilidad y Accesibilidad . . . . .	74
6.5.4. Pruebas de Accesibilidad . . . . .	74
<b>7. CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN</b>	<b>75</b>
7.1. CSI 1: PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONS- TRUCCIÓN . . . . .	76
7.1.1. Estándares y normas seguidos . . . . .	76
7.1.2. Lenguajes de programación . . . . .	76
7.1.3. Herramientas y programas usados para el desarrollo . . . . .	77
7.2. CSI 2: GENERACIÓN DEL CÓDIGO DE LOS COMPONENTES Y PRO- CEDIMIENTOS . . . . .	79
7.3. CSI 3: EJECUCIÓN DE LAS PRUEBAS UNITARIAS . . . . .	80
7.4. CSI 4: EJECUCIÓN DE LAS PRUEBAS DE INTEGRACIÓN . . . . .	81
7.5. CSI 5: EJECUCIÓN DE LAS PRUEBAS DEL SISTEMA . . . . .	82
7.5.1. Prueba de Usabilidad . . . . .	82
7.5.2. Pruebas de Accesibilidad . . . . .	82
7.6. CSI 6: ELABORACIÓN DE LOS MANUALES DE USUARIO . . . . .	83
7.6.1. Manual de Instalación . . . . .	83
7.6.2. Manual de Ejecución . . . . .	83
7.6.3. Manual de Usuario . . . . .	84
7.6.4. Manual del Programador . . . . .	91

7.7. CSI 8: CONSTRUCCIÓN DE LOS COMPONENTES Y PROCEDIMIENTOS DE MIGRACIÓN Y CARGA INICIAL DE DATOS . . . . .	93
<b>8. IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA</b>	<b>94</b>
8.1. IAS 1: ESTABLECIMIENTO DEL PLAN DE IMPLANTACIÓN . . . . .	95
8.2. IAS 4: CARGA DE DATOS AL ENTORNO DE OPERACIÓN . . . . .	96
8.3. IAS 5: PRUEBAS DE IMPLANTACIÓN DEL SISTEMA . . . . .	97
8.4. IAS 7: PREPARACIÓN DEL MANTENIMIENTO DEL SISTEMA . . . . .	98
8.5. IAS 8: ESTABLECIMIENTO DEL ACUERDO DE NIVEL DE SERVICIO	99
8.6. IAS 9–10: PRESENTACIÓN Y APROBACIÓN DEL SISTEMA Y PASO A PRODUCCIÓN . . . . .	100
<b>9. CONCLUSIONES Y AMPLIACIONES</b>	<b>101</b>
9.1. CONCLUSIONES . . . . .	102
9.2. AMPLIACIONES . . . . .	103
<b>ANEXOS</b>	<b>104</b>
PLAN DE GESTIÓN DE RIESGOS . . . . .	105
CONTENIDO ENTREGADO EN LOS ANEXOS . . . . .	106

# Índice de figuras

2.1. Diagrama de la arquitectura tecnológica . . . . .	17
3.1. Logos de JavaScript y Node.js . . . . .	19
3.2. Logos de Angular, TypeScript y PHP . . . . .	20
3.3. Logo de MySQL . . . . .	20
5.1. Diagrama de casos de uso del museo . . . . .	30
5.2. Diagrama de casos de uso de la administración del museo . . . . .	31
5.3. Análisis de clases: diagrama de clases del museo . . . . .	40
5.4. Análisis de clases: diagrama de clases de la administración . . . . .	41
5.5. Prototipo: Página de inicio . . . . .	51
5.6. Prototipo: Página de la vista general del museo . . . . .	52
5.7. Prototipo: Página de detalles del periodo (museo) . . . . .	52
5.8. Prototipo: Página de detalles del componente (museo) . . . . .	53
5.9. Prototipo: Página de inicio de sesión . . . . .	53
5.10. Prototipo: Página de listado de periodos . . . . .	54
5.11. Prototipo: Página de detalles de un periodo (administración) . . . . .	54
5.12. Prototipo: Página de detalles de un componente (administración) . . . . .	55
5.13. Prototipo: Formulario para añadir o editar un periodo . . . . .	55
5.14. Prototipo: Formulario para añadir o editar un componente . . . . .	56
5.15. Diagrama de navegabilidad del museo . . . . .	56
5.16. Diagrama de navegabilidad de la administración del museo . . . . .	57
6.1. Diseño de clases: diagrama de clases del museo . . . . .	62
6.2. Diseño de clases: diagrama de clases de la administración del museo . . . . .	63
6.3. Página de inicio . . . . .	64
6.4. Página de la vista general del museo . . . . .	65
6.5. Página de detalles del periodo (museo) . . . . .	65
6.6. Página de detalles del componente (museo) . . . . .	66



6.7. Página de inicio de sesión . . . . .	66
6.8. Página de listado de periodos . . . . .	67
6.9. Página de detalles de un periodo (administración) . . . . .	67
6.10. Página de detalles de un componente (administración) . . . . .	68
6.11. Formulario para añadir un periodo . . . . .	68
6.12. Formulario para editar un periodo . . . . .	69
6.13. Formulario para añadir un componente . . . . .	69
6.14. Formulario para editar un componente . . . . .	70
6.15. Diagrama Entidad-Relación de la base de datos creada . . . . .	72
7.1. Logo de Visual Studio Code . . . . .	77
7.2. Logo de XAMPP . . . . .	77
7.3. Logo de MobaXTerm . . . . .	77
7.4. Logo de Git . . . . .	78
7.5. Instalación de Angular CLI . . . . .	83
7.6. Instalación de los paquetes del proyecto del museo . . . . .	83
7.7. Instalación de los paquetes del proyecto de administración . . . . .	83
7.8. Ejecución de la aplicación del museo . . . . .	84
7.9. Ejecución de la aplicación de administración . . . . .	84
7.10. Manual de usuario: Inicio . . . . .	85
7.11. Manual de usuario: Vista general del museo . . . . .	86
7.12. Manual de usuario: Detalles del periodo (museo) . . . . .	86
7.13. Manual de usuario: Detalles del componente (museo) . . . . .	87
7.14. Manual de usuario: Inicio de sesión . . . . .	87
7.15. Manual de usuario: Listado de periodos . . . . .	88
7.16. Manual de usuario: Detalles de un periodo (administración) . . . . .	88
7.17. Manual de usuario: Detalles de un componente (administración) . . . . .	89
7.18. Manual de usuario: Formulario para añadir un periodo . . . . .	89
7.19. Manual de usuario: Formulario para editar un periodo . . . . .	90
7.20. Manual de usuario: Formulario para añadir un componente . . . . .	90
7.21. Manual de usuario: Formulario para editar un componente . . . . .	91

# Índice de tablas

5.1. Especificación Caso de Uso 1 . . . . .	30
5.2. Especificación Caso de Uso 2 . . . . .	31
5.3. Especificación Caso de Uso 3 . . . . .	32
5.4. Especificación Caso de Uso 4 . . . . .	32
5.5. Especificación Caso de Uso 5 . . . . .	32
5.6. Especificación Caso de Uso 6 . . . . .	32
5.7. Especificación Caso de Uso 7 . . . . .	32
5.8. Especificación Caso de Uso 8 . . . . .	33
5.9. Especificación Caso de Uso 9 . . . . .	33
5.10. Especificación Caso de Uso 10 . . . . .	33
5.11. Especificación Caso de Uso 11 . . . . .	33
5.12. Análisis del Caso de Uso 1 . . . . .	34
5.13. Análisis del Caso de Uso 2 . . . . .	34
5.14. Análisis del Caso de Uso 3 . . . . .	35
5.15. Análisis del Caso de Uso 4 . . . . .	35
5.16. Análisis del Caso de Uso 5 . . . . .	36
5.17. Análisis del Caso de Uso 6 . . . . .	36
5.18. Análisis del Caso de Uso 7 . . . . .	37
5.19. Análisis del Caso de Uso 8 . . . . .	37
5.20. Análisis del Caso de Uso 9 . . . . .	38
5.21. Análisis del Caso de Uso 10 . . . . .	38
5.22. Análisis del Caso de Uso 11 . . . . .	39
5.23. Descripción de la clase Period . . . . .	42
5.24. Descripción de la interfaz MyComponent . . . . .	42
5.25. Descripción de la clase Cpu . . . . .	43
5.26. Descripción de la clase PeriodService (museo) . . . . .	43
5.27. Descripción de la clase CompService (museo) . . . . .	44

5.28. Descripción de la clase TimelineComponent . . . . .	44
5.29. Descripción de la clase PeriodDetailsComponent . . . . .	45
5.30. Descripción de la clase CompDetailsComponent . . . . .	45
5.31. Descripción de la clase UserService . . . . .	46
5.32. Descripción de la clase PeriodService (administración) . . . . .	46
5.33. Descripción de la clase CompService (administración) . . . . .	47
5.34. Descripción de la clase LoginComponent . . . . .	47
5.35. Descripción de la clase ListPeriodsComponent . . . . .	48
5.36. Descripción de la clase PeriodComponent . . . . .	48
5.37. Descripción de la clase AddPeriodComponent . . . . .	48
5.38. Descripción de la clase EditPeriodComponent . . . . .	49
5.39. Descripción de la clase MyCompComponent . . . . .	49
5.40. Descripción de la clase AddCompComponent . . . . .	49
5.41. Descripción de la clase EditCompComponent . . . . .	50
5.42. Pruebas unitarias: Caso de uso 1 . . . . .	58
5.43. Pruebas unitarias: Caso de uso 2 . . . . .	58
5.44. Pruebas unitarias: Caso de uso 3 . . . . .	58
5.45. Pruebas unitarias: Caso de uso 4 . . . . .	58
5.46. Pruebas unitarias: Caso de uso 5 . . . . .	59
5.47. Pruebas unitarias: Caso de uso 6 . . . . .	59
5.48. Pruebas unitarias: Caso de uso 7 . . . . .	59
5.49. Pruebas unitarias: Caso de uso 8 . . . . .	59
5.50. Pruebas unitarias: Caso de uso 9 . . . . .	60
5.51. Pruebas unitarias: Caso de uso 10 . . . . .	60
5.52. Pruebas unitarias: Caso de uso 11 . . . . .	60
7.1. Descripción de diseño de LoginScreen . . . . .	79
7.2. Descripción de diseño de HomeScreen . . . . .	79

# 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 7: DEFINICIÓN DE LA ARQUITECTURA TECNOLÓGICA

### 2.2.1. PSI 7.2: Selección de la Arquitectura Tecnológica

Al tratarse de una aplicación Angular, seguirá el patrón Modelo Vista Vista-Modelo (MVVM), una variación del Modelo Vista Controlador, patrón arquitectónico que separa los datos y la lógica de una aplicación de su representación. En la variación MVVM, la vista y el modelo son muy dependientes entre sí.

En este caso la vista estará compuesta por las *templates* de Angular, que son componentes HTML.

El modelo se corresponde con la base de datos MySQL y a la que se accede desde un servidor Apache que aloja los archivos PHP necesarios.

Por último, la vista-modelo es la propia aplicación de Angular.

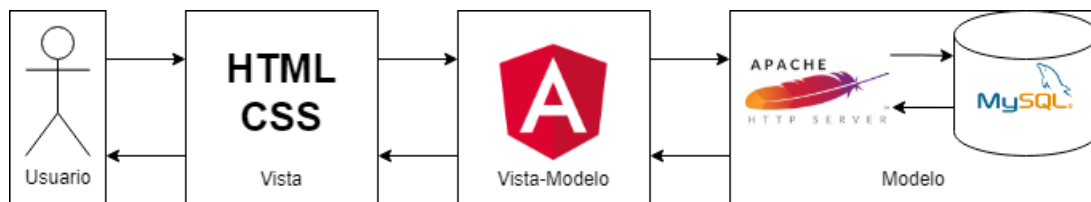


Figura 2.1: Diagrama de la arquitectura tecnológica

## Capítulo 3

# ESTUDIO DE VIABILIDAD DEL SISTEMA

### FASE DE DESARROLLO

## EVS

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

#### 3.1.1. Evaluación de alternativas de desarrollo

##### 3.1.1.1. 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 utilizar 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 3.1: Logos de JavaScript y Node.js

##### 3.1.1.2. 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 3.2: 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.

### 3.1.2. Evaluación de alternativas de gestor de bases de datos

#### 3.1.2.1. 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 3.3: Logo de MySQL

## Capítulo 4

# PLANIFICACIÓN Y GESTIÓN DEL TFG

## **4.1. PLANIFICACIÓN DEL PROYECTO**

### **4.1.1. Identificación de Interesados**

### **4.1.2. OBS y PBS**

### **4.1.3. Planificación Inicial. WBS**

### **4.1.4. Riesgos**

#### **4.1.4.1. Plan de Gestión de Riesgos**

#### **4.1.4.2. Identificación de Riesgos**

#### **4.1.4.3. Registro de Riesgos**

### **4.1.5. Presupuesto Inicial**

#### **4.1.5.1. Presupuesto de Costes**

#### **4.1.5.2. Presupuesto de Cliente**



## **4.2. EJECUCIÓN DEL PROYECTO**

### **4.2.1. Plan Seguimiento de Planificación**

### **4.2.2. Bitácora de Incidencias del Proyecto**

### **4.2.3. Riesgos**

## **4.3. CIERRE DEL PROYECTO**

### **4.3.1. Planificación Final**

### **4.3.2. Informe Final de Riesgos**

### **4.3.3. Presupuesto Final de Costes**

### **4.3.4. Informe de Lecciones Aprendidas**

## Capítulo 5

# ANÁLISIS DEL SISTEMA DE INFORMACIÓN

FASE DE DESARROLLO

ASI

## 5.1. ASI 2: ESTABLECIMIENTO DE REQUISITOS

### 5.1.1. Obtención de los Requisitos del Sistema

#### 5.1.1.1. Requisitos de interfaces externas

##### Interfaces de usuario

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

**RIE-IU-2.** El sistema estará disponible en diferentes idiomas.

**RIE-IU-2.1.** Español

**RIE-IU-2.2.** Inglés

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

**RIE-IU-3.1.** Google Chrome

**RIE-IU-3.2.** Mozilla Firefox

**RIE-IU-3.3.** Microsoft Edge

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

**RIE-IU-5.** El usuario no necesitará de conocimientos tecnológicos avanzados.

##### Interfaces hardware

**RIE-IH-1.** El sistema dispondrá de una base de datos para almacenar la información necesaria.

##### Interfaces de comunicaciones

**RIE-IC-1.** El sistema contendrá enlaces a diferentes sitios web.

**RIE-IC-2.** El sistema mostrará por defecto enlaces a los siguientes sitios web.

**RIE-IC-2.1.** Twitter oficial de la Escuela de Ingeniería Informática

**RIE-IC-2.2.** Página web de la Escuela de Ingeniería Informática

**RIE-IC-2.3.** Página web de la Universidad de Oviedo

#### 5.1.1.2. Requisitos funcionales

**RF-1.** El sistema estará constituido por dos aplicaciones web diferentes.

**RF-1.1.** El museo.

**RF-1.2.** La administración del museo.

## Museo

**RF-MU-1.** El sistema mostrará los periodos existentes.

**RF-MU-1.1.** Los periodos estarán ordenados por año de inicio.

**RF-MU-1.2.** Se presentarán en una línea temporal incluyendo los siguientes datos.

**RF-MU-1.2.1.** Nombre.

**RF-MU-1.2.2.** Año de inicio.

**RF-MU-1.2.3.** Año de fin.

**RF-MU-1.2.4.** Nombres de los componentes pertenecientes al periodo.

**RF-MU-2.** El sistema permitirá realizar búsquedas.

**RF-MU-2.1.** Se podrá buscar por los siguientes campos.

**RF-MU-2.1.1.** Por nombre.

**RF-MU-2.1.2.** Por un intervalo de años.

**RF-MU-2.2.** Al realizar la búsqueda se mostrará la línea temporal filtrada con los resultados.

**RF-MU-3.** El sistema mostrará los detalles de un periodo.

**RF-MU-3.1.** Nombre.

**RF-MU-3.2.** Características.

**RF-MU-3.3.** Listado de datos curiosos (Sabías qué...).

**RF-MU-3.4.** Eventos ocurridos durante el periodo.

**RF-MU-3.5.** Sistemas famosos que contienen componentes de este periodo.

**RF-MU-3.6.** Listado de los componentes pertenecientes al periodo.

**RF-MU-4.** El sistema mostrará los detalles de un componente.

**RF-MU-4.1.** Nombre.

**RF-MU-4.2.** Descripción.

**RF-MU-4.3.** Imágenes.

**RF-MU-4.4.** Año de inicio.

**RF-MU-4.5.** Año de fin.

**RF-MU-4.6.** Precio.

**RF-MU-4.7.** Tipo de dispositivos en los que se encuentra.

**RF-MU-4.7.1.** Portátiles.

**RF-MU-4.7.2.** De escritorio.

**RF-MU-4.8.** Detalles específicos del tipo de componente.

**RF-MU-4.8.1.** Detalles de CPU.

**RF-MU-4.8.1.1.** Memoria ROM (obligatorio).

**RF-MU-4.8.1.2.** Memoria RAM (obligatorio).

**RF-MU-4.8.1.3.** Velocidad de reloj (obligatorio).

**RF-MU-4.8.1.4.** Potencia (obligatorio).

- RF-MU-4.8.1.5.** Tamaño de palabra (obligatorio).
- RF-MU-4.8.1.6.** Nanómetros de los transistores (obligatorio).
- RF-MU-4.8.1.7.** Passmark (obligatorio).
- RF-MU-4.8.1.8.** Número de transistores (obligatorio).

### Administración del museo

- RF-ADM-1.** El sistema permitirá al usuario iniciar sesión mediante un formulario.
  - RF-ADM-1.1.** El formulario se solicitan los siguientes campos.
    - RF-ADM-1.1.1.** Correo electrónico (obligatorio).
    - RF-ADM-1.1.2.** Contraseña (obligatorio).
  - RF-ADM-1.2.** Si los campos no son correctos, se mostrará de nuevo el inicio de sesión.
  - RF-ADM-1.3.** Si los campos son válidos, se accederá al sistema como administrador.
- RF-ADM-2.** El sistema mostrará un listado de los periodos existentes.
- RF-ADM-3.** El sistema mostrará los detalles de un periodo.
  - RF-ADM-3.1.** Especificados en **RF-MU-3**.
- RF-ADM-4.** El sistema mostrará los detalles de un componente.
  - RF-ADM-4.1.** Especificados en **RF-MU-4**.
- RF-ADM-5.** El sistema permitirá añadir un periodo.
  - RF-ADM-5.1.** Se introducirán los datos mediante un formulario con los siguientes campos.
    - RF-ADM-5.1.1.** Nombre (obligatorio).
    - RF-ADM-5.1.2.** Características (obligatorio).
    - RF-ADM-5.1.3.** Listado de datos curiosos (Sabías qué...) (obligatorio).
    - RF-ADM-5.1.4.** Eventos ocurridos durante el periodo (obligatorio).
- RF-ADM-6.** El sistema permitirá añadir un componente a un periodo existente.
  - RF-ADM-6.1.** Se introducirán los datos mediante un formulario con los siguientes campos.
    - RF-ADM-6.1.1.** Nombre (obligatorio).
    - RF-ADM-6.1.2.** Tipo de componente (obligatorio).
      - RF-ADM-6.1.2.1.** CPU.
    - RF-ADM-6.1.3.** Descripción (obligatorio).
    - RF-ADM-6.1.4.** Familia de componente (obligatorio).
    - RF-ADM-6.1.5.** Imágenes.
    - RF-ADM-6.1.6.** Año de inicio (obligatorio).
    - RF-ADM-6.1.7.** Año de fin (obligatorio).
    - RF-ADM-6.1.8.** Precio (obligatorio).
    - RF-ADM-6.1.9.** Tipo de dispositivos en los que se encuentra.
      - RF-ADM-6.1.9.1.** Portátiles.

**RF-ADM-6.1.9.2.** De escritorio.

**RF-ADM-6.1.10.** Sistema famoso que lo contiene.

**RF-ADM-6.1.10.1.** Nombre del sistema.

**RF-ADM-6.1.10.2.** Imagen del sistema.

**RF-ADM-6.1.11.** Detalles específicos del tipo de componente.

**RF-ADM-6.1.11.1.** Detalles de CPU.

**RF-ADM-6.1.11.1.1.** Memoria ROM (obligatorio).

**RF-ADM-6.1.11.1.2.** Memoria RAM (obligatorio).

**RF-ADM-6.1.11.1.3.** Velocidad de reloj (obligatorio).

**RF-ADM-6.1.11.1.4.** Potencia (obligatorio).

**RF-ADM-6.1.11.1.5.** Tamaño de palabra (obligatorio).

**RF-ADM-6.1.11.1.6.** Nanómetros de los transistores (obligatorio).

**RF-ADM-6.1.11.1.7.** Passmark (obligatorio).

**RF-ADM-6.1.11.1.8.** Número de transistores (obligatorio).

**RF-ADM-7.** El sistema permitirá editar periodos.

**RF-ADM-7.1.** Se introducirán los datos mediante un formulario.

**RF-ADM-7.1.1.** Campos especificados en **RF-ADM-5.1.**

**RF-ADM-8.** El sistema permitirá editar componentes.

**RF-ADM-8.1.** Se introducirán los datos mediante un formulario.

**RF-ADM-8.1.1.** Campos especificados en **RF-ADM-6.1.**

**RF-ADM-9.** El sistema permitirá eliminar un periodo.

**RF-ADM-9.1.** Se pedirá confirmación antes de eliminarlo.

**RF-ADM-9.2.** Se eliminarán los componentes pertenecientes a dicho periodo.

**RF-ADM-10.** El sistema permitirá eliminar un componente.

**RF-ADM-10.1.** Se pedirá confirmación antes de eliminarlo.

### **5.1.1.3. Atributos del sistema**

#### **Seguridad**

**RNF-SEG-1.** La parte de administración del sistema se asegurará de que el usuario se identifica para acceder a ella.

**RNF-SEG-1.1.** El usuario se identificará mediante un email y una contraseña.

**RNF-SEG-2.** El sistema cifrará la contraseña para almacenarla en la base de datos.

## 5.1.2. Identificación de Actores del Sistema

### 5.1.2.1. Usuario estándar

Actor que interactúa con el sistema. Tiene acceso de lectura a la página web del museo. Solo debe tener un conocimiento básico para navegar por internet.

### 5.1.2.2. Usuario administrador

Actor que interactúa con el sistema. Debe iniciar sesión en la parte de administración del sistema, es el único actor con acceso a esta. Es responsable de gestionar el sistema y su mantenimiento. Debe tener amplios conocimientos sobre el sistema.

## 5.1.3. Especificación de Casos de Uso

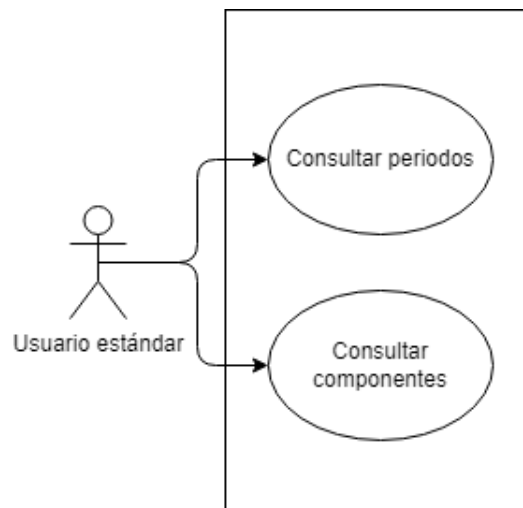


Figura 5.1: Diagrama de casos de uso del museo

Tabla 5.1: Especificación Caso de Uso 1

Nombre del caso de uso
Consultar periodos (museo)
Descripción
Un usuario estándar puede visualizar los periodos existentes en el museo.



Tabla 5.2: Especificación Caso de Uso 2

Nombre del caso de uso
Consultar componentes (museo)
Descripción
Un usuario estándar puede visualizar los componentes pertenecientes a cada periodo del museo.

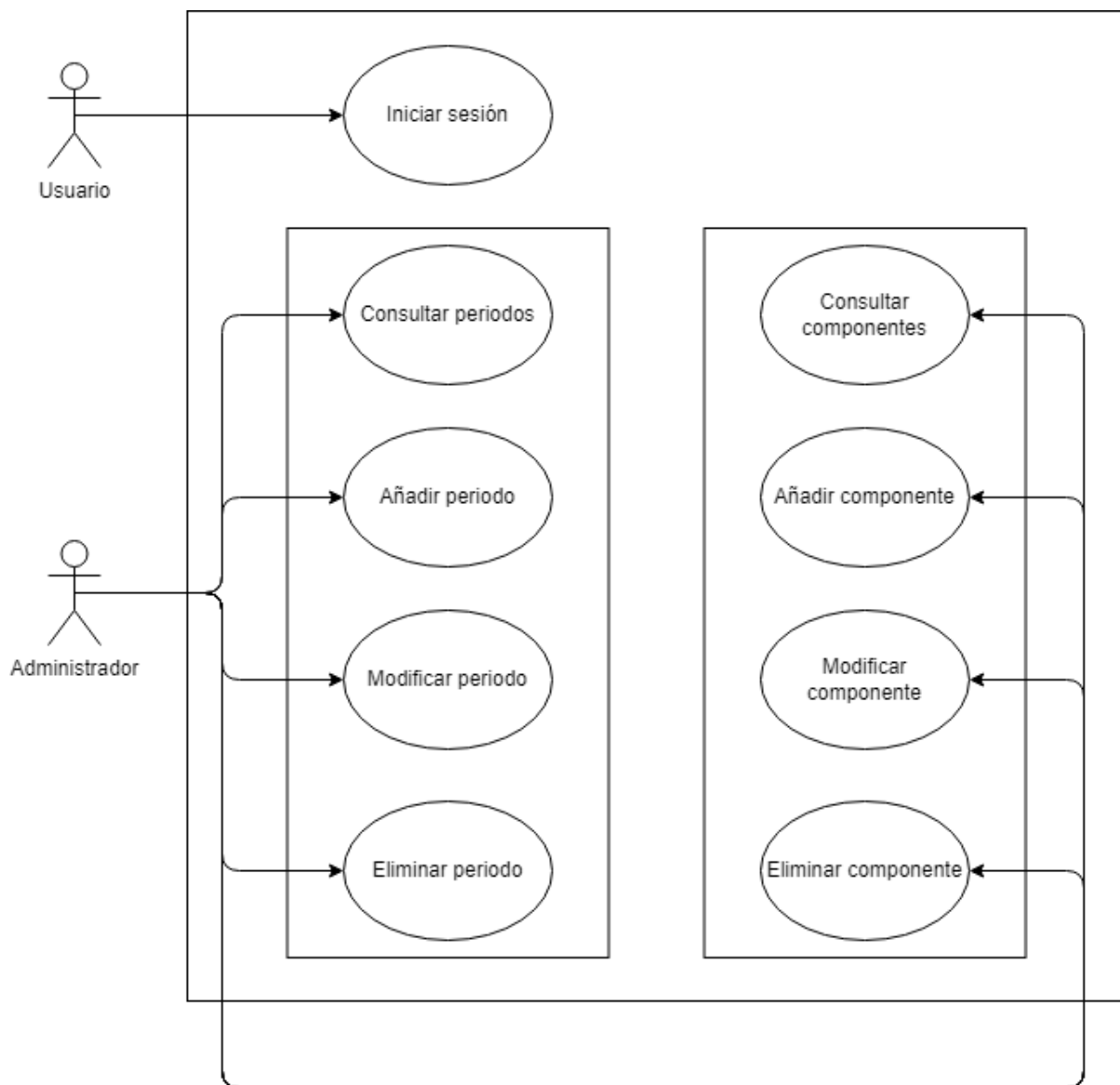


Figura 5.2: Diagrama de casos de uso de la administración del museo

Tabla 5.3: Especificación Caso de Uso 3

Nombre del caso de uso
Iniciar sesión
Descripción
Un usuario puede iniciar sesión en la aplicación para acceder a esta como administrador.

Tabla 5.4: Especificación Caso de Uso 4

Nombre del caso de uso
Consultar periodos (administración)
Descripción
El administrador puede consultar los periodos existentes en el sistema.

Tabla 5.5: Especificación Caso de Uso 5

Nombre del caso de uso
Añadir periodo
Descripción
El administrador puede añadir periodos a la base de datos del sistema.

Tabla 5.6: Especificación Caso de Uso 6

Nombre del caso de uso
Modificar periodo
Descripción
El administrador puede modificar los periodos existentes en la base de datos del sistema.

Tabla 5.7: Especificación Caso de Uso 7

Nombre del caso de uso
Eliminar periodo
Descripción
El administrador puede eliminar los periodos existentes en la base de datos del sistema.

Tabla 5.8: Especificación Caso de Uso 8

Nombre del caso de uso
Consultar componentes (administración)
Descripción
El administrador puede consultar los componentes existentes en el sistema.

Tabla 5.9: Especificación Caso de Uso 9

Nombre del caso de uso
Añadir componente
Descripción
El administrador puede añadir componentes a los periodos existentes en la base de datos del sistema.

Tabla 5.10: Especificación Caso de Uso 10

Nombre del caso de uso
Modificar componente
Descripción
El administrador puede modificar los componentes existentes en la base de datos del sistema.

Tabla 5.11: Especificación Caso de Uso 11

Nombre del caso de uso
Eliminar componente
Descripción
El administrador puede eliminar los componentes existentes en la base de datos del sistema.

## 5.2. ASI 4: ANÁLISIS DE LOS CASOS DE USO

### 5.2.1. Caso de Uso 1

Tabla 5.12: Análisis del Caso de Uso 1

<b>Consultar periodos (museo)</b>	
<b>Precondiciones</b>	-
<b>Postcondiciones</b>	-
<b>Actores</b>	Usuario estándar
<b>Descripción</b>	El usuario accederá a la vista principal del museo y podrá visualizar los periodos existentes. Podrá acceder a los periodos. Podrá acceder a los componentes de los periodos. Podrá realizar una búsqueda.
<b>Escenarios Secundarios</b>	

### 5.2.2. Caso de Uso 2

Tabla 5.13: Análisis del Caso de Uso 2

<b>Consultar componentes (museo)</b>	
<b>Precondiciones</b>	-
<b>Postcondiciones</b>	-
<b>Actores</b>	Usuario estándar
<b>Descripción</b>	El usuario accederá a la vista de un periodo y podrá visualizar los componentes pertenecientes al mismo. Podrá acceder a otros periodos. Podrá acceder a los otros componentes de ese periodo.
<b>Escenarios Secundarios</b>	

### 5.2.3. Caso de Uso 3

Tabla 5.14: Análisis del Caso de Uso 3

<b>Iniciar sesión</b>	
<b>Precondiciones</b>	El usuario no debe haber iniciado sesión.
<b>Postcondiciones</b>	-
<b>Actores</b>	Usuario
<b>Descripción</b>	El usuario accederá a la página principal de la aplicación de administración e introducirá su email y contraseña para iniciar sesión en el sistema.
<b>Escenarios Secundarios</b>	Los datos introducidos no se corresponden con los datos de un usuario con permiso de administrador. Se muestra un error y de nuevo se solicita iniciar sesión.

### 5.2.4. Caso de Uso 4

Tabla 5.15: Análisis del Caso de Uso 4

<b>Consultar periodos (administración)</b>	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión.
<b>Postcondiciones</b>	-
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accederá al listado de periodos. Podrá acceder a cada uno de ellos.
<b>Escenarios Secundarios</b>	Aún no existe ningún periodo en el sistema. Se muestra una tabla vacía.

### 5.2.5. Caso de Uso 5

Tabla 5.16: Análisis del Caso de Uso 5

Añadir periodo	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión.
<b>Postcondiciones</b>	El periodo añadido se guardará en la base de datos.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al formulario para añadir un periodo. Rellena los campos necesarios. Pulsa el botón de guardar.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"> <li>- Se pulsa el botón cancelar. El formulario se restablece.</li> <li>- Se intenta acceder a otra página de la aplicación sin haber guardado los cambios. Se avisa de la situación y se pide una confirmación para continuar.</li> <li>- No se puede añadir el periodo. Se mostrará un error avisando de la situación.</li> </ul>

### 5.2.6. Caso de Uso 6

Tabla 5.17: Análisis del Caso de Uso 6

Modificar periodo	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un periodo.
<b>Postcondiciones</b>	Los cambios realizados al periodo se guardarán en la base de datos.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al periodo deseado y selecciona la opción de editar. Se mostrará el formulario correspondiente. Se realizan los cambios en el formulario. Pulsa el botón de guardar.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"> <li>- Se pulsa el botón cancelar. El formulario se restablece.</li> <li>- Se intenta acceder a otra página de la aplicación sin haber guardado los cambios. Se avisa de la situación y se pide una confirmación para continuar.</li> <li>- No se puede modificar el periodo. Se mostrará un error avisando de la situación.</li> </ul>

## 5.2.7. Caso de Uso 7

Tabla 5.18: Análisis del Caso de Uso 7

Eliminar periodo	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un periodo.
<b>Postcondiciones</b>	El periodo eliminado y los componentes que pertenecen al mismo se borrarán de la base de datos y dejarán de mostrarse en la aplicación.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al periodo deseado y selecciona la opción de eliminar. Se pide confirmación para eliminarlo. Se acepta esta confirmación.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"> <li>- No se acepta la confirmación para eliminarlo. El periodo y sus componentes permanecen en la base de datos.</li> <li>- No se puede eliminar el periodo. Se mostrará un error avisando de la situación.</li> </ul>

## 5.2.8. Caso de Uso 8

Tabla 5.19: Análisis del Caso de Uso 8

Consultar componentes (administración)	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un periodo.
<b>Postcondiciones</b>	-
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accederá a un periodo existente y visualizará los componentes pertenecientes a este. Podrá acceder a cada uno de ellos.
<b>Escenarios Secundarios</b>	Aún no existen componentes para el periodo que se consulta. Se mostrará una tabla vacía.

### 5.2.9. Caso de Uso 9

Tabla 5.20: Análisis del Caso de Uso 9

Añadir componente	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un periodo.
<b>Postcondiciones</b>	El componente añadido se guardará en la base de datos y se asociará al periodo correspondiente.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al formulario para añadir un componente. Rellena los campos necesarios. Pulsa el botón de guardar.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"> <li>- Se pulsa el botón cancelar. El formulario se restablece.</li> <li>- Se intenta acceder a otra página de la aplicación sin haber guardado los cambios. Se avisa de la situación y se pide una confirmación para continuar.</li> <li>- No se puede añadir el componente. Se mostrará un error avisando de la situación.</li> </ul>

### 5.2.10. Caso de Uso 10

Tabla 5.21: Análisis del Caso de Uso 10

Modificar componente	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un componente.
<b>Postcondiciones</b>	Los cambios realizados en el componente se guardarán en la base de datos.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al componente deseado y selecciona la opción de editar. Se mostrará el formulario correspondiente. Se realizan los cambios en el formulario. Pulsa el botón de guardar.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"> <li>- Se pulsa el botón cancelar. El formulario se restablece.</li> <li>- Se intenta acceder a otra página de la aplicación sin haber guardado los cambios. Se avisa de la situación y se pide una confirmación para continuar.</li> <li>- No se puede modificar el componente. Se mostrará un error avisando de la situación.</li> </ul>



### 5.2.11. Caso de Uso 11

Tabla 5.22: Análisis del Caso de Uso 11

Eliminar componente	
<b>Precondiciones</b>	El usuario debe haber iniciado sesión. Debe existir al menos un componente.
<b>Postcondiciones</b>	El componente eliminado se borrará de la base de datos y dejará de mostrarse en la aplicación.
<b>Actores</b>	Usuario administrador
<b>Descripción</b>	El usuario accede al componente deseado y selecciona la opción de eliminar. Se pide confirmación para eliminarlo. Se acepta esta confirmación.
<b>Escenarios Secundarios</b>	<ul style="list-style-type: none"><li>- No se acepta la confirmación para eliminarlo. El componente permanece en la base de datos.</li><li>- No se puede eliminar el componente. Se mostrará un error avisando de la situación.</li></ul>

## 5.3. ASI 5: ANÁLISIS DE CLASES

### 5.3.1. Diagrama de Clases

#### 5.3.1.1. Museo

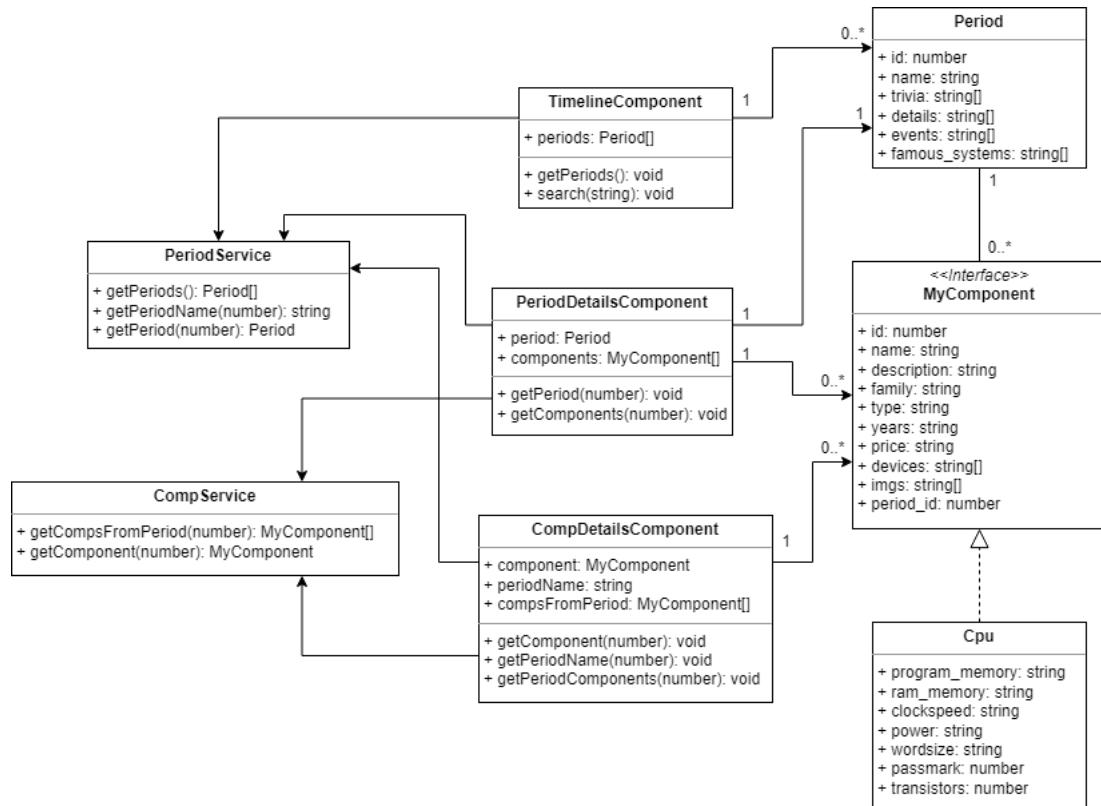


Figura 5.3: Análisis de clases: diagrama de clases del museo

### 5.3.1.2. Administración del museo

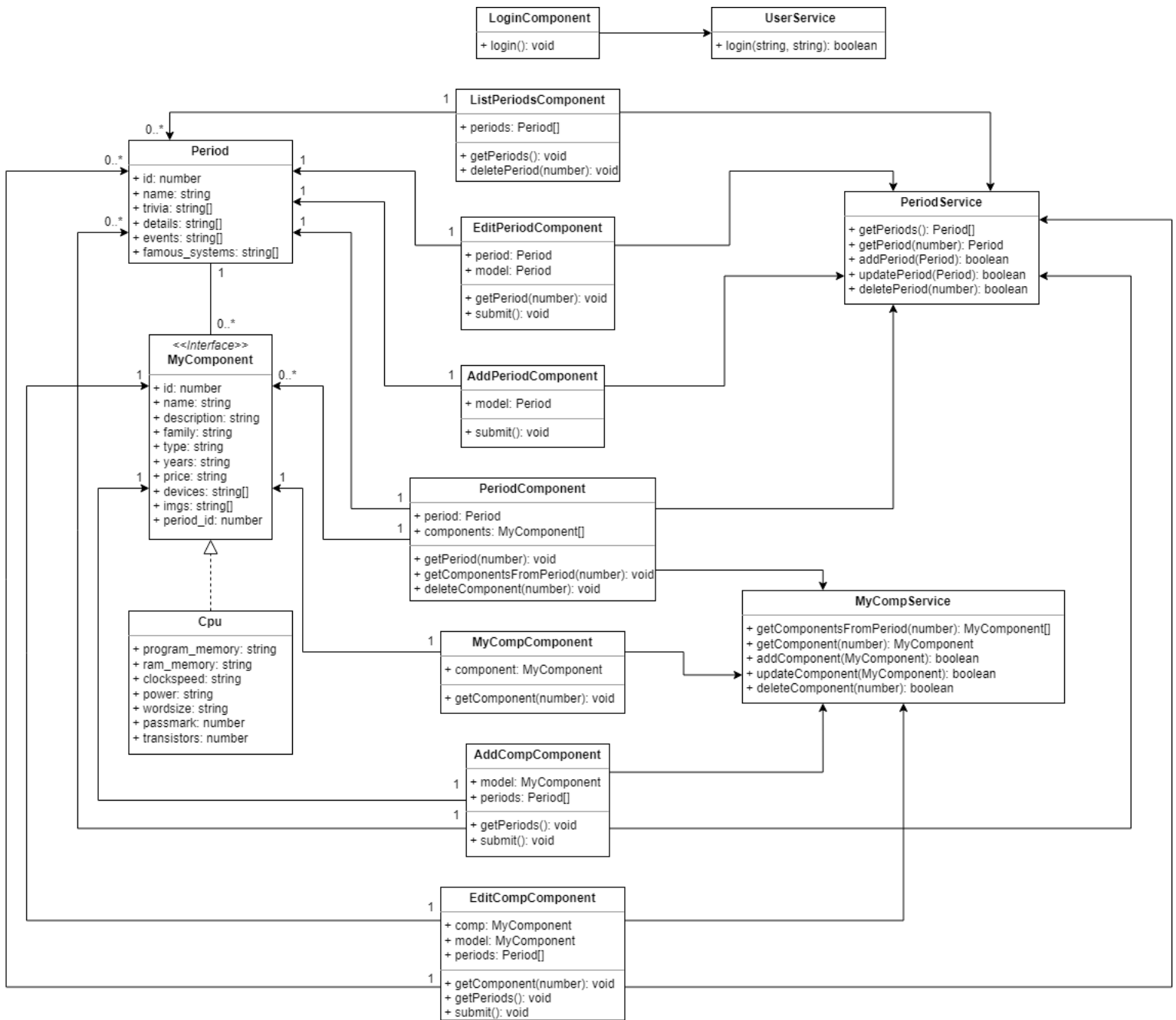


Figura 5.4: Análisis de clases: diagrama de clases de la administración

### 5.3.2. Descripción de las Clases

*Period*, *MyComponent* y *Cpu* son iguales en el proyecto del museo y en el de la administración, por lo tanto se describen una única vez a continuación:

Tabla 5.23: Descripción de la clase Period

Periodo		
Descripción		
Clase que modela un periodo.		
Atributos propuestos		
<b>id</b>	number	Identificador del periodo
<b>name</b>	string	Nombre del periodo
<b>trivia</b>	string[]	Curiosidades del periodo
<b>details</b>	string[]	Detalles del periodo
<b>events</b>	string[]	Eventos ocurridos durante este periodo
<b>famous_systems</b>	string[]	Sistemas famosos que llevaban componentes pertenecientes al periodo
Métodos propuestos		
-		

Tabla 5.24: Descripción de la interfaz MyComponent

MyComponent		
Descripción		
Interfaz que modela los atributos genéricos de un componente.		
Atributos propuestos		
<b>id</b>	number	Identificador del componente
<b>name</b>	string	Nombre del componente
<b>description</b>	string	Descripción del componente
<b>family</b>	string	Familia a la que pertenece
<b>type</b>	string	Tipo de componente (CPU, genérico...)
<b>years</b>	string	Rango de años en los que se utilizó
<b>price</b>	string	Precio de venta del componente
<b>devices</b>	string[]	Tipo de dispositivos en los que se usaba el componente (portátiles o de escritorio)
<b>imgs</b>	string[]	Nombres de las imágenes del componente
<b>period_id</b>	number	Identificador del periodo al que pertenece el componente
Métodos propuestos		
-		

Tabla 5.25: Descripción de la clase Cpu

Cpu		
Descripción		
Clase que implementa la interfaz MyComponent. Modela una CPU, con sus atributos específicos correspondientes.		
Atributos propuestos		
<b>program_memory</b>	string	Memoria ROM de la CPU
<b>ram_memory</b>	string	Memoria RAM de la CPU
<b>clockspeed</b>	string	Velocidad de reloj de la CPU
<b>power</b>	string	Potencia de la CPU
<b>wordsize</b>	string	Tamaño de palabra de la CPU
<b>passmark</b>	number	Passmark de la CPU
<b>transistors</b>	number	Número de transistores de la CPU
Métodos propuestos		
-		

#### 5.3.2.1. Museo

Tabla 5.26: Descripción de la clase PeriodService (museo)

PeriodService	
Descripción	
Servicio que conecta con el back-end de la aplicación para realizar las operaciones relacionadas con los periodos.	
Atributos propuestos	
-	
Métodos propuestos	
<b>getPeriods</b>	Devuelve todos los periodos existentes.
<b>getPeriodName</b>	Devuelve el nombre del periodo correspondiente al identificador pasado por parámetro.
<b>getPeriod</b>	Devuelve el periodo cuyo identificador se pasa como parámetro.

Tabla 5.27: Descripción de la clase CompService (museo)

CompService	
Descripción	
Servicio que conecta con el back-end de la aplicación para realizar las operaciones relacionadas con los componentes.	
Atributos propuestos	
-	
Métodos propuestos	
<b>getCompsFromPeriod</b>	Devuelve los componentes pertenecientes al periodo cuyo identificador se pasa como parámetro.
<b>getComponent</b>	Devuelve el componente cuyo identificador se pasa como parámetro.

Tabla 5.28: Descripción de la clase TimelineComponent

TimelineComponent		
Descripción		
Clase asociada a la vista 5.4.1.1.		
Atributos propuestos		
<b>periods</b>	Period[]	Listado de todos los periodos existentes.
Métodos propuestos		
<b>getPeriods</b>	Obtiene los periodos y los asigna a <i>periods</i> .	
<b>search</b>	Filtra los periodos según el texto introducido en la búsqueda.	

Tabla 5.29: Descripción de la clase PeriodDetailsComponent

PeriodDetailsComponent		
Descripción		
Clase asociada a la vista 5.4.1.1.		
Atributos propuestos		
<b>period</b>	Period	Periodo del que se muestran los detalles.
<b>components</b>	MyComponent[]	Componentes pertenecientes al periodo.
Métodos propuestos		
<b>getPeriod</b>	Obtiene el periodo y lo asigna a <i>period</i> .	
<b>getComponents</b>	Obtiene los componentes del periodo y los asigna a <i>components</i> .	

Tabla 5.30: Descripción de la clase CompDetailsComponent

CompDetailsComponent		
Descripción		
Clase asociada a la vista 5.4.1.1.		
Atributos propuestos		
<b>component</b>	MyComponent	Componente del que se muestran los detalles.
<b>periodName</b>	string	Nombre del periodo al que pertenece el componente.
<b>components</b>	MyComponent[]	Otros componentes pertenecientes al periodo.
Métodos propuestos		
<b>getComponent</b>	Obtiene el componente y lo asigna a <i>component</i> .	
<b>getPeriodName</b>	Obtiene el nombre del periodo y lo asigna a <i>periodName</i> .	
<b>getPeriodComponents</b>	Obtiene los componentes del periodo y los asigna a <i>components</i> .	

## 5.3.2.2. Administración del museo

Tabla 5.31: Descripción de la clase UserService

UserService	
Descripción	
Servicio que conecta con el back-end de la aplicación para realizar las operaciones relacionadas con el usuario administrador.	
Atributos propuestos	
-	
Métodos propuestos	
<b>login</b>	Comprueba si el usuario y la contraseña introducidos se corresponden con los existentes en la base de datos.

Tabla 5.32: Descripción de la clase PeriodService (administración)

PeriodService	
Descripción	
Servicio que conecta con el back-end de la aplicación para realizar las operaciones relacionadas con los periodos.	
Atributos propuestos	
-	
Métodos propuestos	
<b>getPeriods</b>	Devuelve todos los periodos existentes.
<b>getPeriod</b>	Devuelve el periodo cuyo identificador se pasa como parámetro.
<b>addPeriod</b>	Añade el periodo pasado como parámetro a la base de datos.
<b>updatePeriod</b>	Actualiza el periodo pasado como parámetro en la base de datos.
<b>getPeriod</b>	Elimina de la base de datos el periodo cuyo identificador se pasa como parámetro.



Tabla 5.33: Descripción de la clase CompService (administración)

CompService	
Descripción	
Servicio que conecta con el back-end de la aplicación para realizar las operaciones relacionadas con los componentes.	
Atributos propuestos	
-	
Métodos propuestos	
<b>getCompsFromPeriod</b>	Devuelve los componentes pertenecientes al periodo cuyo identificador se pasa como parámetro.
<b>getComponent</b>	Devuelve el componente cuyo identificador se pasa como parámetro.
<b>addComponent</b>	Añade el componente pasado como parámetro a la base de datos.
<b>updateComponent</b>	Actualiza el componente pasado como parámetro en la base de datos.
<b>getComponent</b>	Elimina de la base de datos el componente cuyo identificador se pasa como parámetro.

Tabla 5.34: Descripción de la clase LoginComponent

LoginComponent	
Descripción	
Clase asociada a la vista 5.4.1.2.	
Atributos propuestos	
-	
Métodos propuestos	
<b>login</b>	Comprueba los datos introducidos para iniciar sesión.

Tabla 5.35: Descripción de la clase ListPeriodsComponent

ListPeriodsComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>periods</b>	Period[]	Listado de todos los periodos existentes.
Métodos propuestos		
<b>getPeriods</b>	Obtiene los periodos y los asigna a <i>periods</i> .	
<b>deletePeriod</b>	Elimina el periodo seleccionado.	

Tabla 5.36: Descripción de la clase PeriodComponent

PeriodComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>period</b>	Period	Periodo del que se muestran los detalles.
<b>components</b>	MyComponent[]	Componentes pertenecientes al periodo.
Métodos propuestos		
<b>getPeriod</b>	Obtiene el periodo y lo asigna a <i>period</i> .	
<b>getComponentsFromPeriod</b>	Obtiene los componentes del periodo y los asigna a <i>components</i> .	
<b>deleteComponent</b>	Elimina el componente seleccionado.	

Tabla 5.37: Descripción de la clase AddPeriodComponent

AddPeriodComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>model</b>	Period	Periodo asociado al formulario en el que se introducen los datos.
Métodos propuestos		
<b>submit</b>	Añade el periodo con los datos introducidos en el formulario.	

Tabla 5.38: Descripción de la clase EditPeriodComponent

EditPeriodComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>period</b>	Period	Periodo que se va a editar, con los datos iniciales.
<b>model</b>	Period	Periodo asociado al formulario en el que se editan los datos.
Métodos propuestos		
<b>getPeriod</b>	Obtiene el periodo y lo asigna a <i>period</i> .	
<b>submit</b>	Actualiza el periodo con los datos introducidos en el formulario.	

Tabla 5.39: Descripción de la clase MyCompComponent

MyCompComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>component</b>	MyComponent	Componente del que se muestran los detalles.
Métodos propuestos		
<b>getComponent</b>	Obtiene el componente y lo asigna a <i>component</i> .	

Tabla 5.40: Descripción de la clase AddCompComponent

AddCompComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>model</b>	MyComponent	Componente asociado al formulario en el que se introducen los datos.
<b>periods</b>	Period[]	Listado de periodos existentes.
Métodos propuestos		
<b>getPeriods</b>	Obtiene los periodos y los asigna a <i>periods</i> .	
<b>submit</b>	Añade el componente con los datos introducidos en el formulario.	

Tabla 5.41: Descripción de la clase EditCompComponent

EditCompComponent		
Descripción		
Clase asociada a la vista 5.4.1.2.		
Atributos propuestos		
<b>comp</b>	MyComponent	Componente que se va a editar, con los datos iniciales.
<b>model</b>	MyComponent	Componente asociado al formulario en el que se editan los datos.
<b>periods</b>	Period[]	Listado de periodos existentes.
Métodos propuestos		
<b>getComponent</b>	Obtiene el componente y lo asigna a <i>comp</i> .	
<b>getPeriods</b>	Obtiene los periodos y los asigna a <i>periods</i> .	
<b>submit</b>	Actualiza el componente con los datos introducidos en el formulario.	

## 5.4. ASI 8: DEFINICIÓN DE INTERFACES DE USUARIO

### 5.4.1. Definición del aspecto de la interfaz

#### 5.4.1.1. Museo

A continuación se presentan los prototipos de interfaces diseñados para la página web del museo. Todas ellas tienen en común la barra de navegación, que contiene el logo de la EII, un enlace a la vista general del museo y un selector de idioma.

#### Inicio

En la página de inicio del museo se muestra un mensaje de bienvenida y un botón que conduce a la vista general del museo.

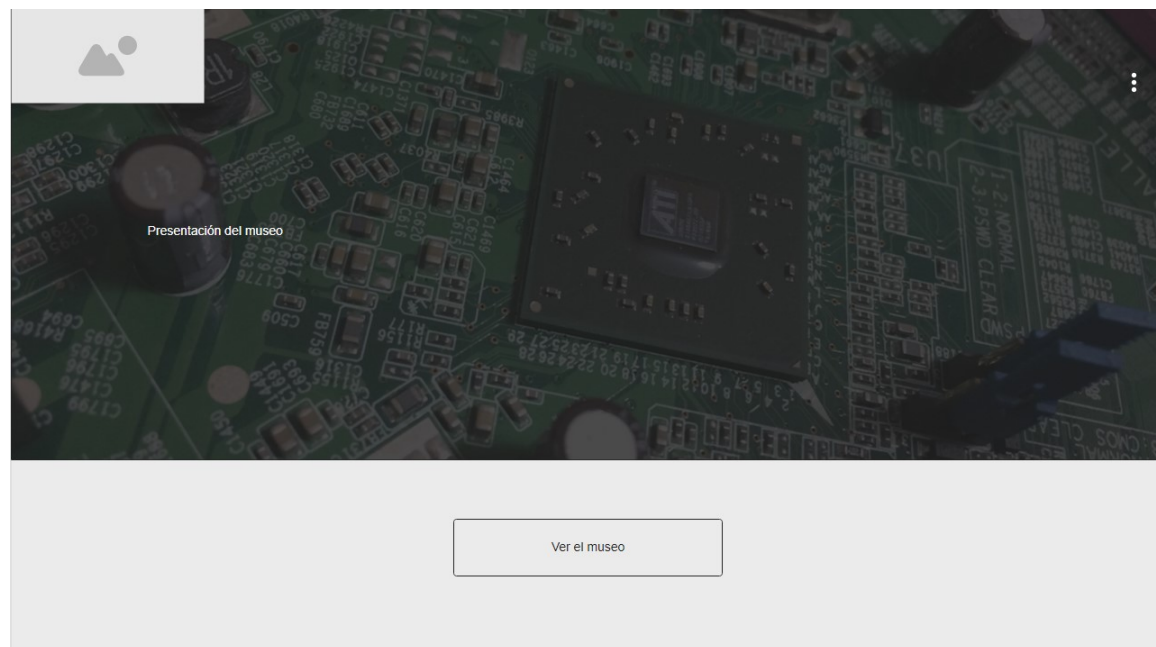


Figura 5.5: Prototipo: Página de inicio

#### Vista general del museo

En la vista general del museo encontramos un menú lateral con filtros de búsqueda, y una sección principal que contiene una línea temporal con los periodos en los que se divide la historia de las CPUs.

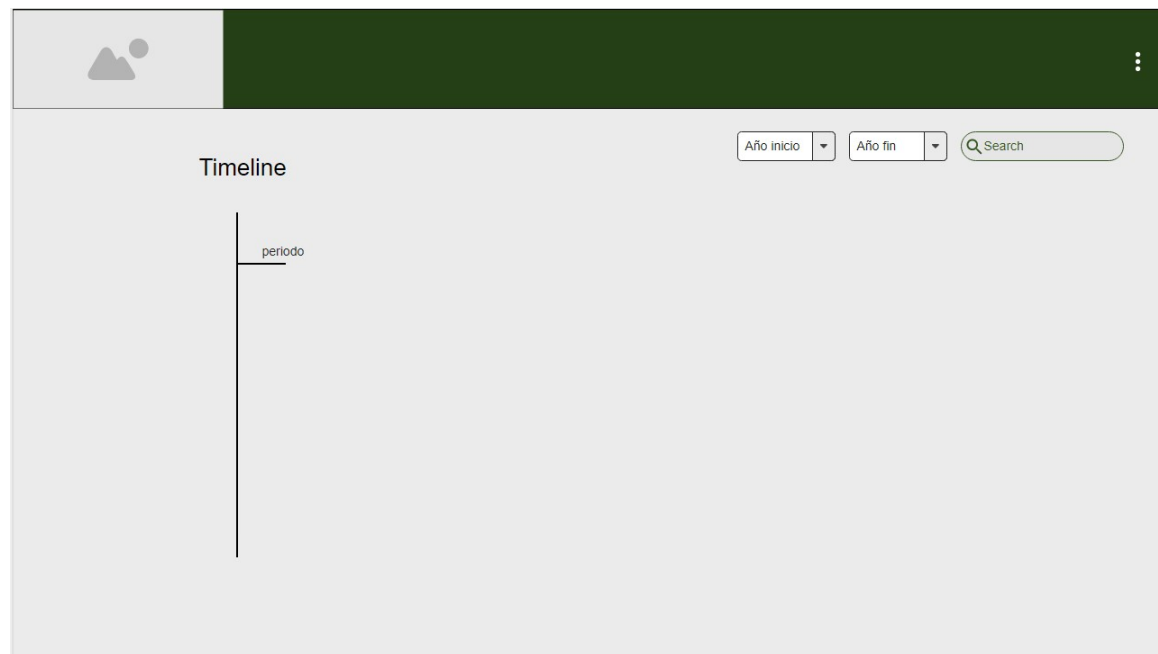


Figura 5.6: Prototipo: Página de la vista general del museo

### Detalles del periodo

En esta página hay un menú para volver a la vista general, y se muestran todos los detalles de un periodo (nombre, características, sistemas famosos de dicho periodo, etc.) y los componentes que pertenecen al mismo.

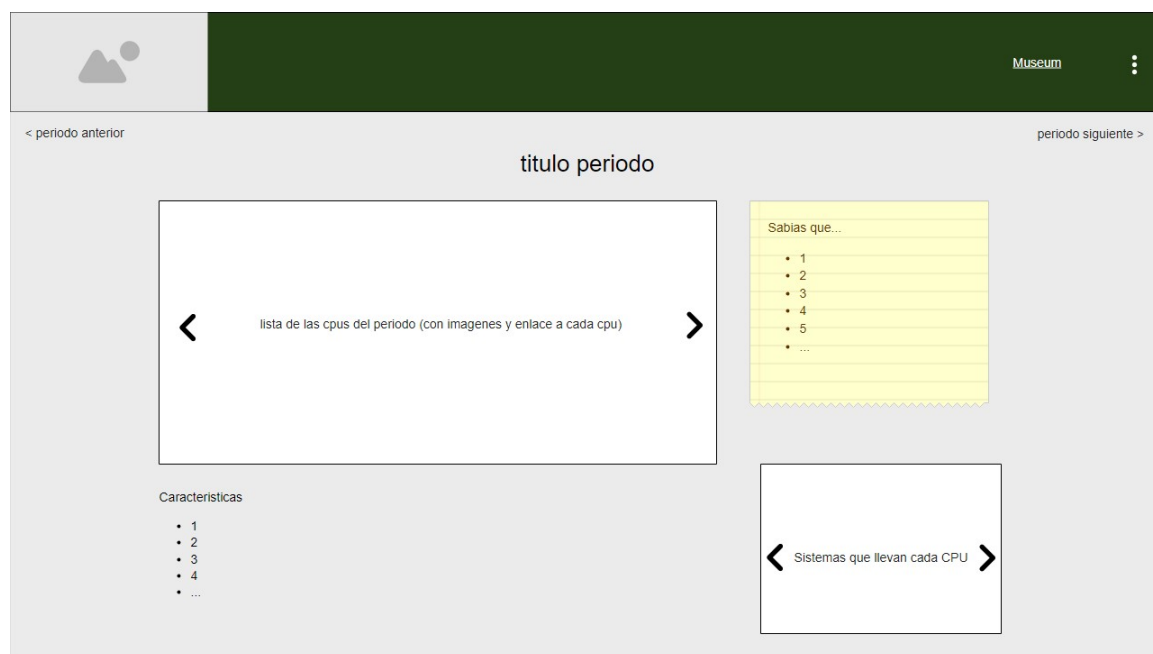


Figura 5.7: Prototipo: Página de detalles del periodo (museo)

## Detalles del componente

En esta página se muestra una galería de fotos del componente, la descripción del mismo, y un listado de características. En el menú de esta página hay un listado de componentes pertenecientes al mismo periodo.

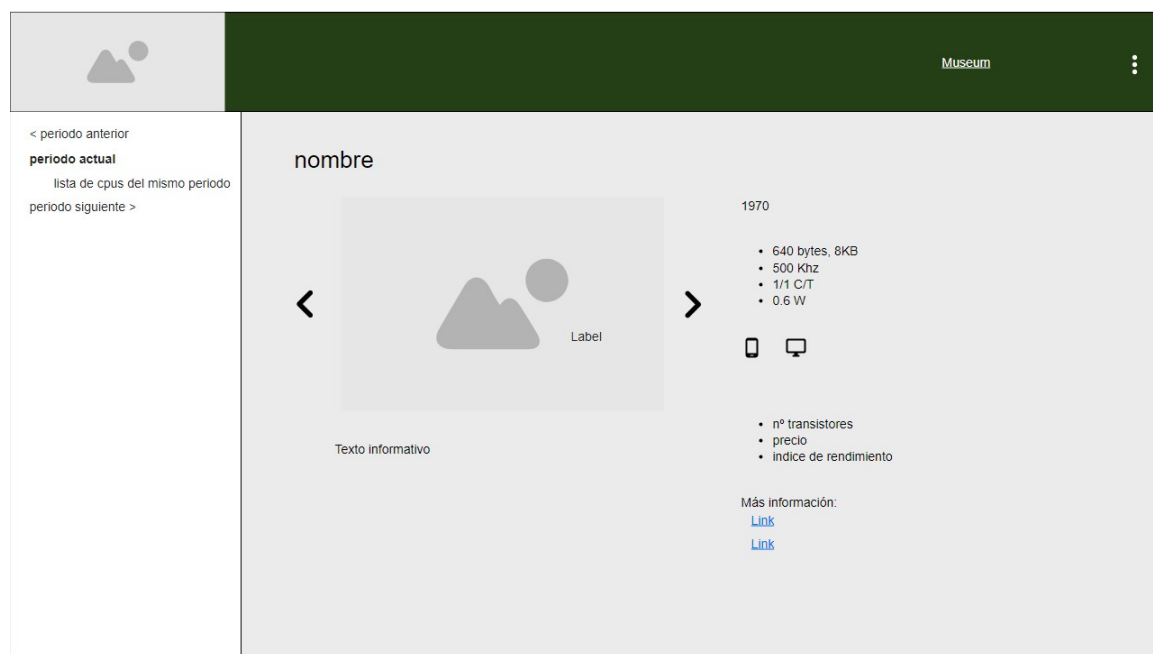


Figura 5.8: Prototipo: Página de detalles del componente (museo)

### 5.4.1.2. Administración del museo

A continuación, se muestran los prototipos iniciales para la aplicación de administración del museo. En todas ellas, salvo en la de inicio de sesión, hay un menú lateral de navegación.

#### Iniciar sesión

En esta página el administrador del sistema deberá introducir su usuario y contraseña para acceder al mismo.



Figura 5.9: Prototipo: Página de inicio de sesión

## Listado de periodos

En esta página se muestra un listado de los periodos existentes, con las opciones de acceder a cada uno, editarlo o eliminarlo.

[Listado de periodos](#)[Añadir periodo](#)[Añadir componente](#)

Periodo 1	Editar	Eliminar
Periodo 2	Editar	Eliminar
Periodo 3	Editar	Eliminar
Periodo 4	Editar	Eliminar

Figura 5.10: Prototipo: Página de listado de periodos

## Periodo

Esta página contiene los detalles de un periodo así como un listado de los componentes pertenecientes al mismo, ofreciendo la opción de acceder a ellos, editarlos o eliminarlos.

[Listado de periodos](#)[Añadir periodo](#)[Añadir componente](#)

### Periodo 1

Características

Sabías qué...

Eventos

Sistemas famosos

Componente 1	Editar	Eliminar
Componente 2	Editar	Eliminar
Componente 3	Editar	Eliminar
Componente 4	Editar	Eliminar

Figura 5.11: Prototipo: Página de detalles de un periodo (administración)



## Componente

En esta página se muestran los detalles correspondientes al componente.

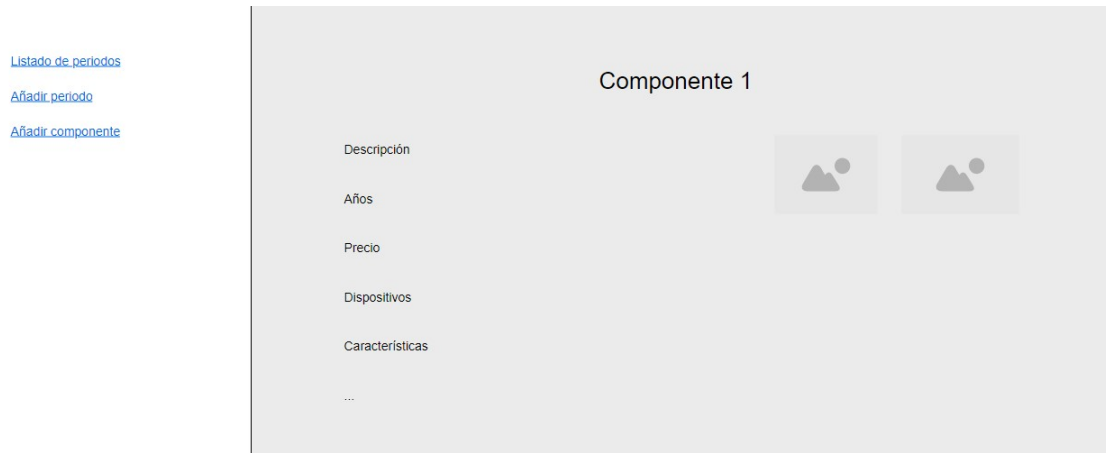


Figura 5.12: Prototipo: Página de detalles de un componente (administración)

## Añadir/editar periodo

Los formularios para añadir o editar un periodo son idénticos, con la única diferencia de que el formulario para editar ya tiene los campos completados con los valores existentes del periodo, por tanto solo se muestra una captura representando ambos.



Figura 5.13: Prototipo: Formulario para añadir o editar un periodo

## Añadir/editar componente

Con los formularios para añadir o editar un componente ocurre igual que con los del periodo ya mencionados.

[Añadir periodo](#)  
[Añadir componente](#)  
[Editar periodo](#)  
[Editar componente](#)

Figura 5.14: Prototipo: Formulario para añadir o editar un componente

### 5.4.2. Diagrama de Navegabilidad

A continuación se presentan dos diagramas de navegabilidad, correspondientes a las dos aplicaciones web que constituyen el sistema.

#### 5.4.2.1. Museo

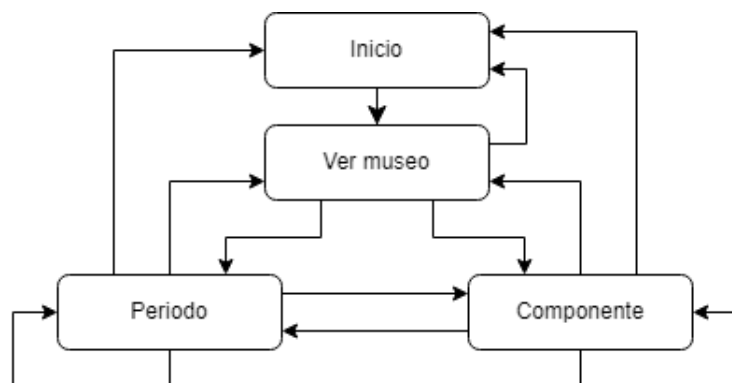


Figura 5.15: Diagrama de navegabilidad del museo

#### 5.4.2.2. Administración del museo

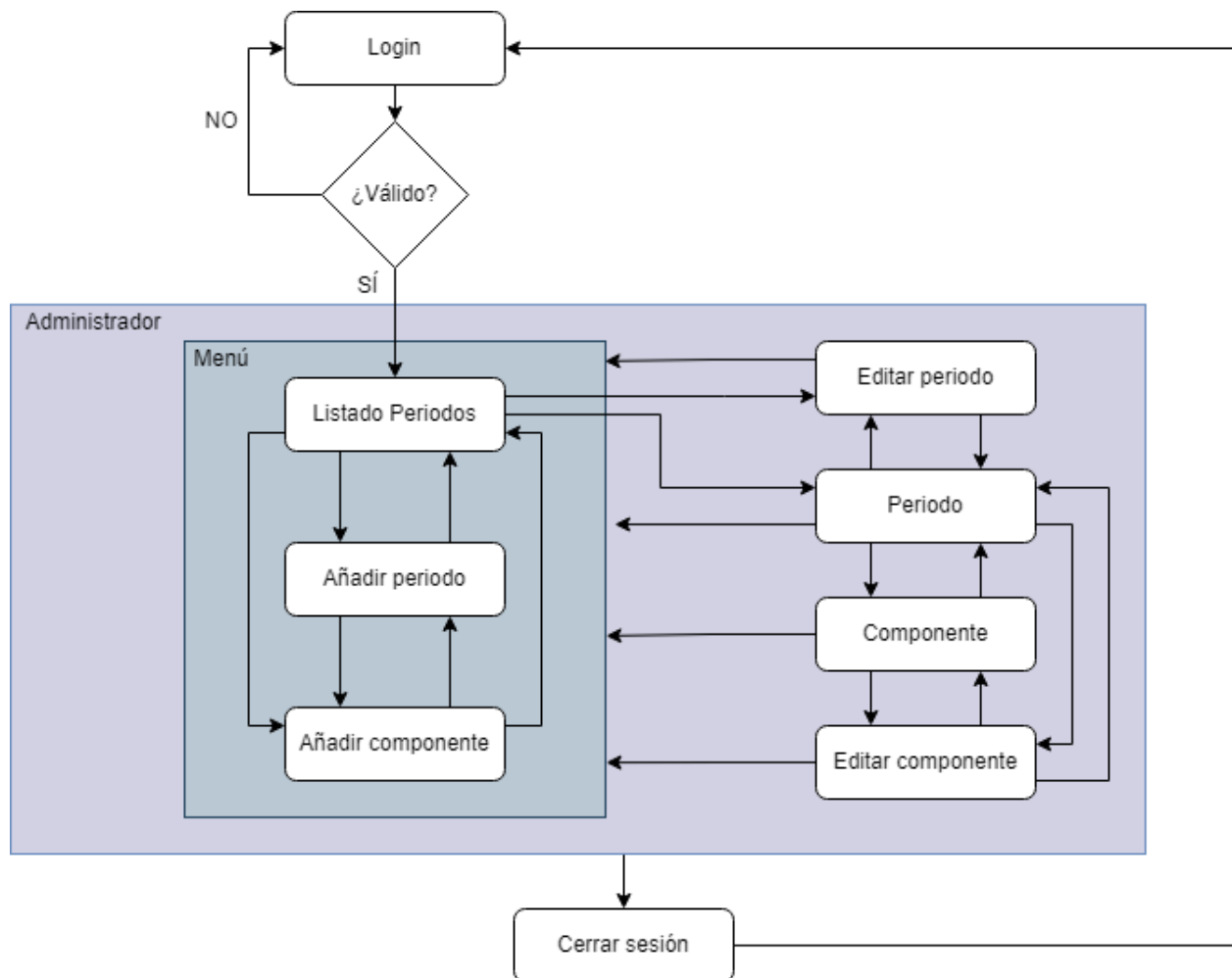


Figura 5.16: Diagrama de navegabilidad de la administración del museo

## 5.5. ASI 10: ESPECIFICACIÓN DEL PLAN DE PRUEBAS

### 5.5.1. Pruebas unitarias

Se realizarán pruebas unitarias del sistema utilizando Jasmine y Karma, herramientas incluidas en Angular para probar el correcto funcionamiento de los diferentes componentes.

Tabla 5.42: Pruebas unitarias: Caso de uso 1

Caso de uso 1: Consultar periodos (museo)	
Prueba	Resultado esperado
Obtener periodos existentes	El sistema devolverá una lista de todos los periodos existentes.
Obtener periodo por nombre	El sistema devolverá una lista de los periodos cuyo nombre contenga el texto introducido.
Obtener periodo por años	El sistema devolverá una lista de los periodos cuyos años coincidan con los introducidos.

Tabla 5.43: Pruebas unitarias: Caso de uso 2

Caso de uso 2: Consultar componentes (museo)	
Prueba	Resultado esperado
Obtener componentes de un periodo	El sistema devolverá una lista de todos los componentes pertenecientes al periodo.

Tabla 5.44: Pruebas unitarias: Caso de uso 3

Caso de uso 3: Iniciar sesión	
Prueba	Resultado esperado
Iniciar sesión con datos válidos	El sistema permitirá el acceso a la página de administración.
Iniciar sesión con datos incorrectos	El sistema no permitirá el acceso y se mostrará un error.

Tabla 5.45: Pruebas unitarias: Caso de uso 4

Caso de uso 4: Consultar periodos (administración)	
Prueba	Resultado esperado
Obtener periodos existentes	El sistema devolverá una lista de todos los periodos existentes.

Tabla 5.46: Pruebas unitarias: Caso de uso 5

Caso de uso 5: Añadir periodo	
Prueba	Resultado esperado
Añadir nuevo periodo	El sistema tendrá un periodo más.
Añadir periodo que ya existe	El sistema no añadirá el periodo y responderá con un error.
Añadir periodo con campos vacíos	El sistema no añadirá el periodo y responderá con un error.

Tabla 5.47: Pruebas unitarias: Caso de uso 6

Caso de uso 6: Modificar periodo	
Prueba	Resultado esperado
Modificar periodo existente	El sistema actualizará los datos del periodo.
Modificar un periodo que no existe	El sistema responderá con un error.
Modificar periodo dejando campos vacíos	El sistema no actualizará el periodo y responderá con un error.

Tabla 5.48: Pruebas unitarias: Caso de uso 7

Caso de uso 7: Eliminar periodo	
Prueba	Resultado esperado
Eliminar un periodo existente	El sistema tendrá un periodo menos.
Eliminar un periodo que no existe	El sistema responderá con un error.

Tabla 5.49: Pruebas unitarias: Caso de uso 8

Caso de uso 8: Consultar componentes (administración)	
Prueba	Resultado esperado
Obtener componentes de un periodo	El sistema devolverá una lista de todos los componentes pertenecientes al periodo.

Tabla 5.50: Pruebas unitarias: Caso de uso 9

Caso de uso 9: Añadir componente	
Prueba	Resultado esperado
Añadir nuevo componente	El sistema tendrá un componente más.
Añadir componente que ya existe	El sistema no añadirá el componente y responderá con un error.
Añadir componente con campos obligatorios vacíos	El sistema no añadirá el componente y responderá con un error.

Tabla 5.51: Pruebas unitarias: Caso de uso 10

Caso de uso 10: Modificar componente	
Prueba	Resultado esperado
Modificar componente existente	El sistema actualizará los datos del componente.
Modificar un componente que no existe	El sistema responderá con un error.
Modificar componente dejando campos obligatorios vacíos	El sistema no actualizará el componente y responderá con un error.

Tabla 5.52: Pruebas unitarias: Caso de uso 11

Caso de uso 11: Eliminar componente	
Prueba	Resultado esperado
Eliminar un componente existente	El sistema tendrá un componente menos.
Eliminar un componente que no existe	El sistema responderá con un error.

### 5.5.2. Pruebas del sistema

Se probará el conjunto completo del sistema, es decir, si la aplicación, el servidor y la base de datos se conectan de forma correcta, sin errores en tiempo de ejecución y obteniendo las respuestas esperadas. Estas pruebas se realizarán de forma manual sobre la aplicación.

### 5.5.3. Pruebas de usabilidad

Para realizar estas pruebas se hará uso de una serie de cuestionarios, cuyos resultados se obtendrán tras observar como usuarios con diferentes perfiles interactúan con el sistema.

## Capítulo 6

# DISEÑO DEL SISTEMA DE INFORMACIÓN

### FASE DE DESARROLLO

## DSI

## 6.1. DSI 4: DISEÑO DE CLASES

### 6.1.1. Diagrama de Clases

#### 6.1.1.1. Museo

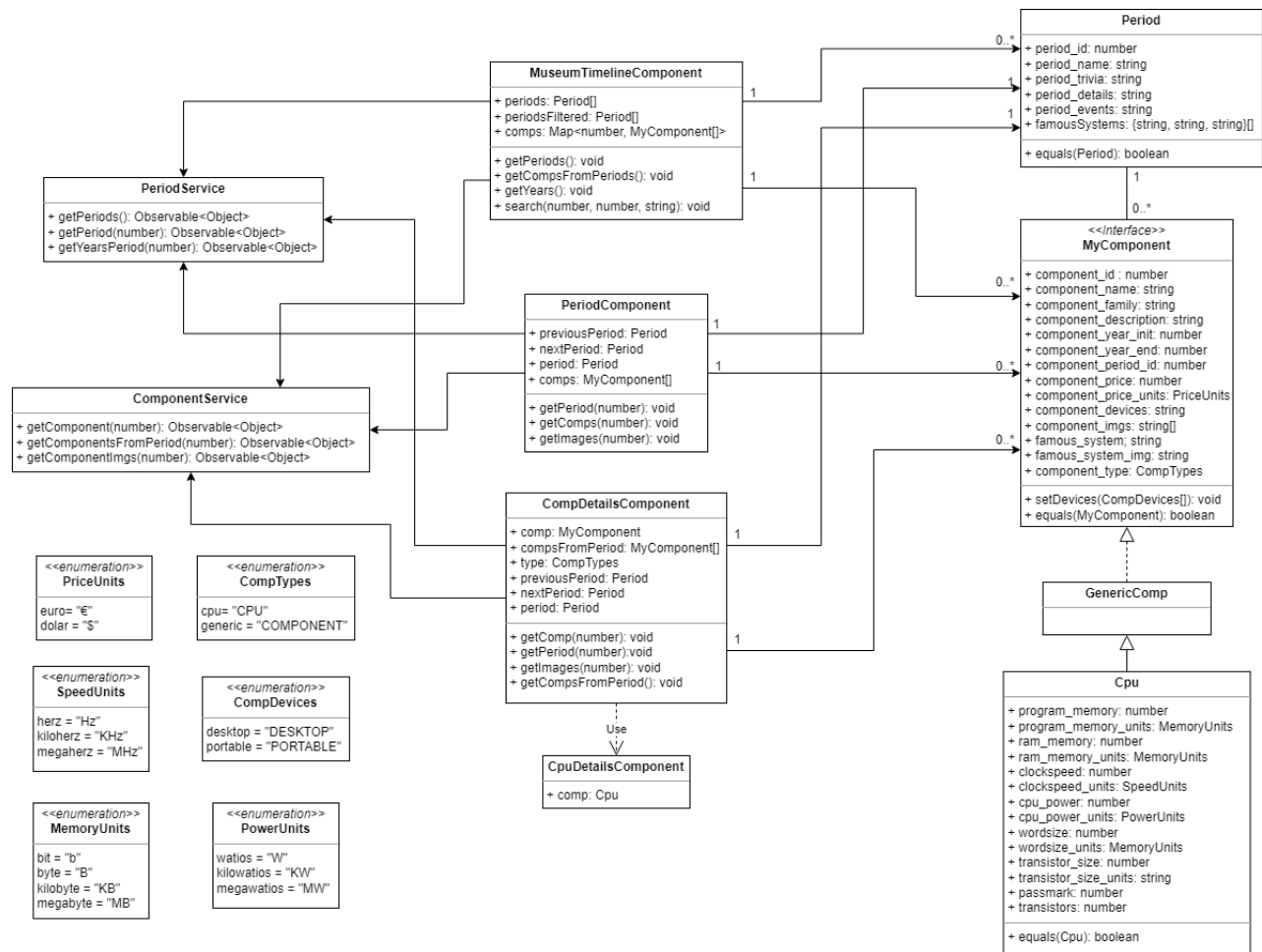


Figura 6.1: Diseño de clases: diagrama de clases del museo



## 6.1.1.2. Administración del museo

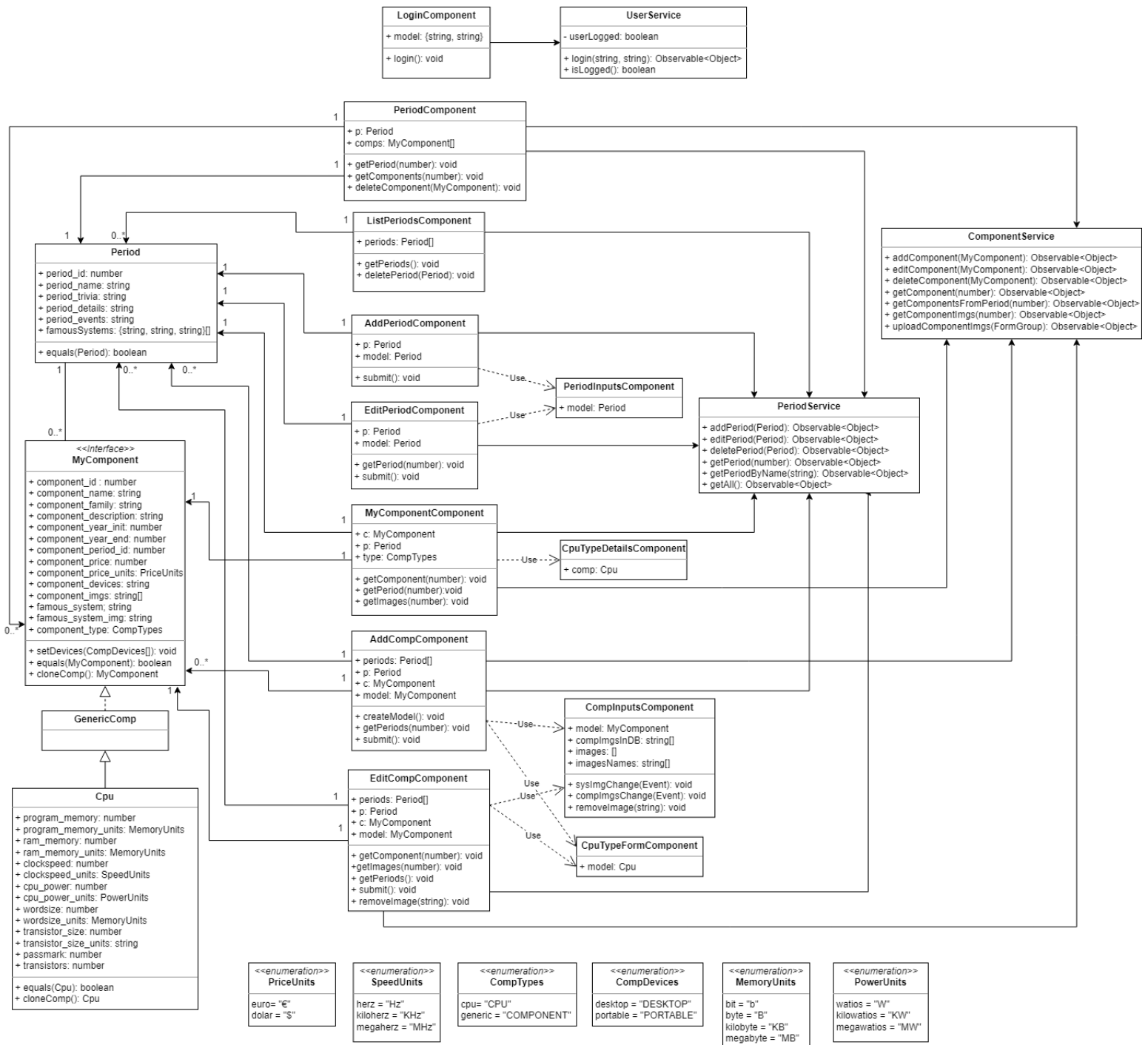


Figura 6.2: Diseño de clases: diagrama de clases de la administración del museo

## 6.2. DSI 5: DISEÑO DE LA ARQUITECTURA DE MÓDULOS DEL SISTEMA

### 6.2.1. DSI 5.1 Diseño de Módulos del Sistema

### 6.2.2. DSI 5.2 Diseño de Comunicaciones entre Módulos

### 6.2.3. DSI 5.3 Revisión de la Interfaz de Usuario

A continuación, se muestran las interfaces definitivas de la aplicación web.

#### 6.2.3.1. Museo

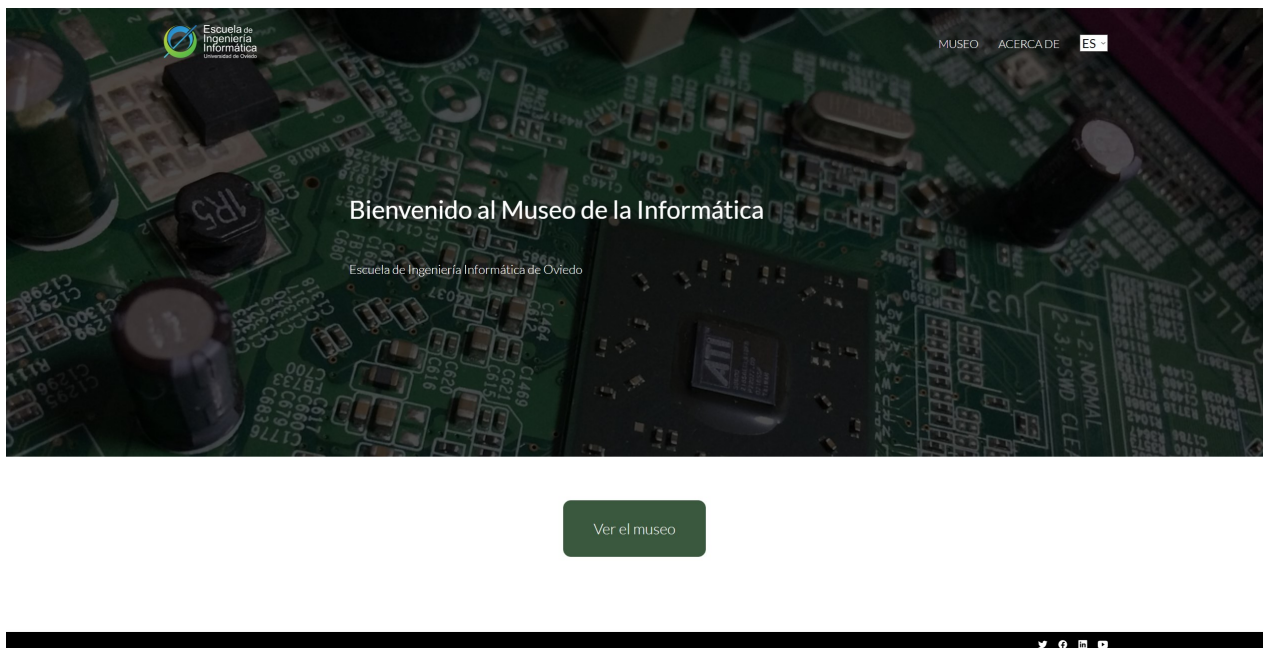


Figura 6.3: Página de inicio

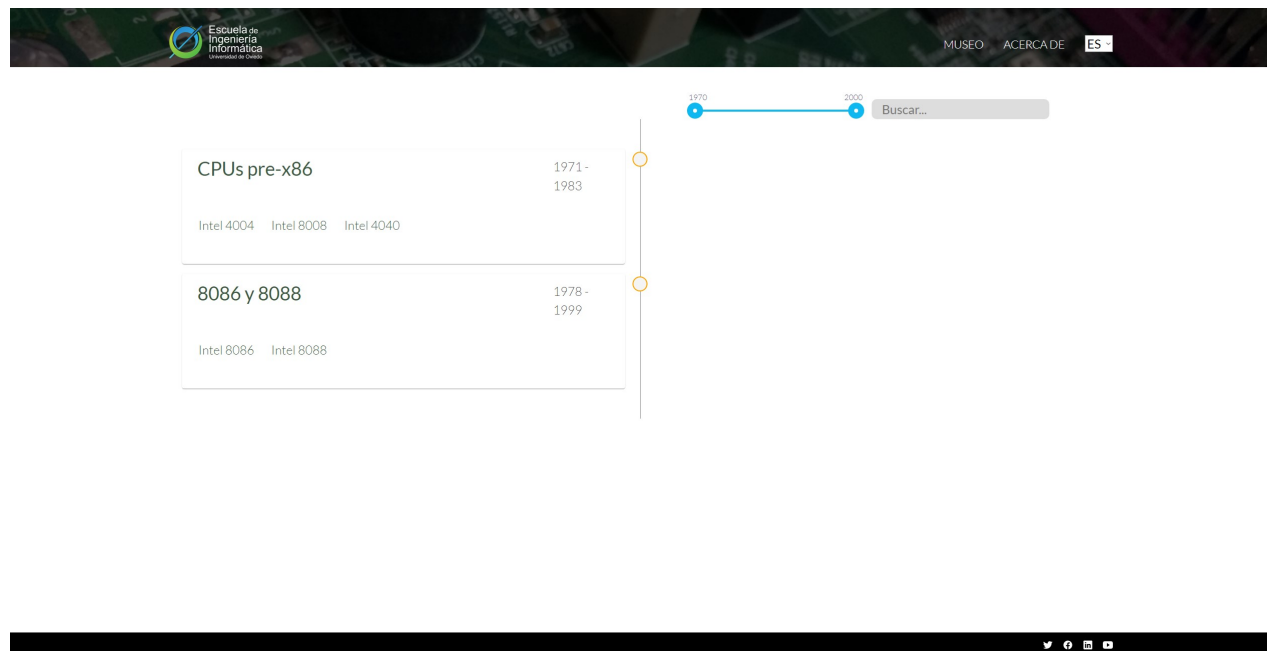


Figura 6.4: Página de la vista general del museo

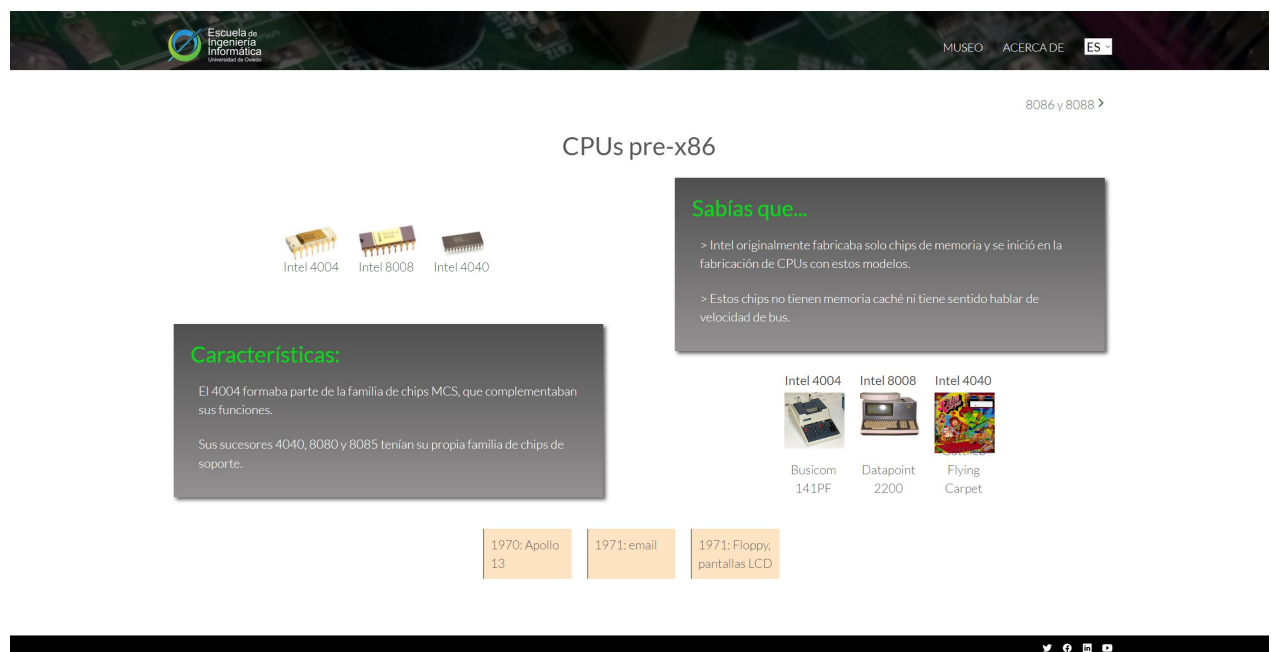


Figura 6.5: Página de detalles del periodo (museo)

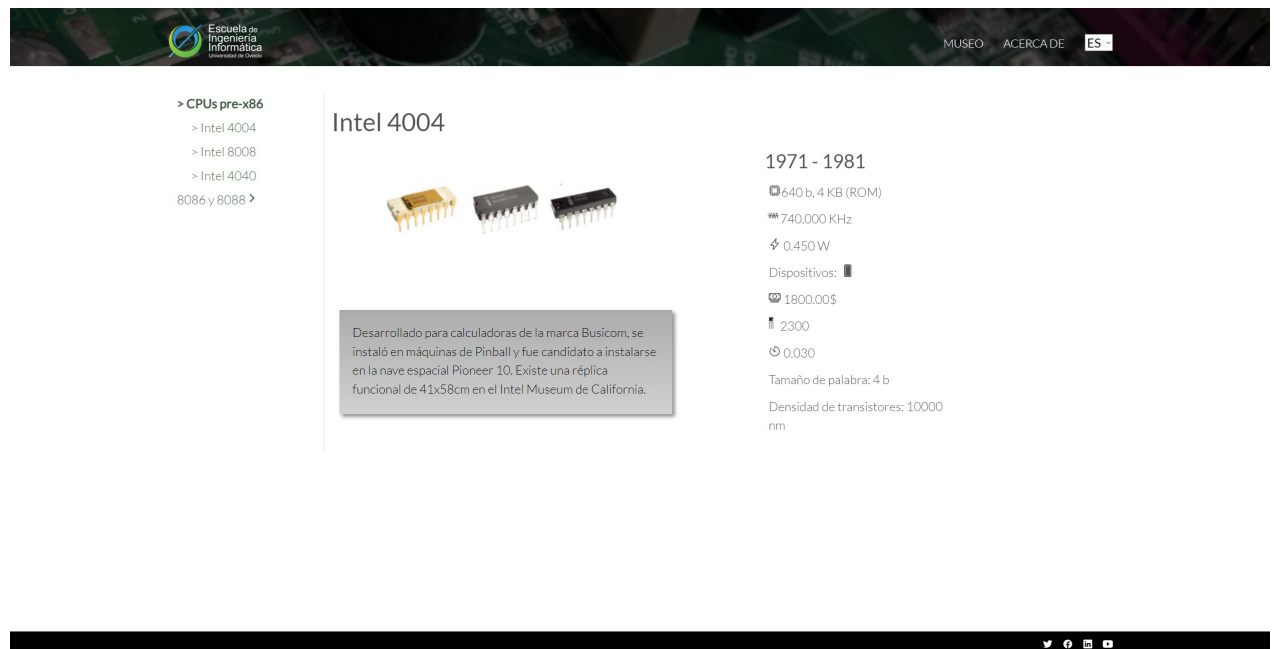


Figura 6.6: Página de detalles del componente (museo)

#### 6.2.3.2. Administración del museo

## Museo de la Informática - Administrador

### Iniciar sesión

**INICIAR SESIÓN**

Figura 6.7: Página de inicio de sesión

> Listado de periodos  
 > Añadir periodo  
 > Añadir componente a periodo existente

## Listado de periodos

Nombre	Editar	Eliminar
CPUs pre-x86		
8086 y 8088		

Figura 6.8: Página de listado de periodos

> Listado de periodos  
 > Añadir periodo  
 > Añadir componente a periodo existente

## CPUs pre-x86

Editar

### Detalles

El 4004 formaba parte de la familia de chips MCS, que complementaban sus funciones.

Sus sucesores 4040, 8080 y 8085 tenían su propia familia de chips de soporte.

### Sabías qué...

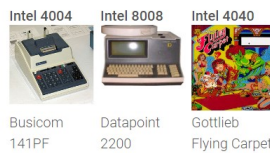
Intel originalmente fabricaba solo chips de memoria y se inició en la fabricación de CPUs con estos modelos.

Estos chips no tienen memoria caché ni tiene sentido hablar de velocidad de bus.

### Eventos

1970: Apollo 13 1971: email 1971: Floppy, pantallas LCD

### Sistemas famosos



### Componentes

Nombre	Editar	Eliminar
Intel 4004		
Intel 8008		
Intel 4040		

Figura 6.9: Página de detalles de un periodo (administración)

> Listado de periodos

> Añadir periodo

> Añadir componente a periodo existente

←

Intel 4004

Editar

<b>Periodo</b> CPUs pre-x86 <b>Familia</b> Intel <b>Años</b> 1971 - 1981 <b>Descripción</b> Desarrollado para calculadoras de la marca Basicom, se instaló en máquinas de Pinball y fue candidato a instalarse en la nave espacial Pioneer 10. Existe una réplica funcional de 41x58cm en el Intel Museum de California.	<b>Precio</b> 1800.00\$ <b>Dispositivos</b> Portátiles, De escritorio.	
<b>Memoria ROM</b> 4 KB <b>Memoria RAM</b> 640 b <b>Frecuencia de reloj</b> 740.000 KHz <b>Consumo energético</b> 0.450 W	<b>Tamaño de palabra</b> 4 b <b>Nanómetros de los transistores</b> 10000 <b>Passmark</b> 0.030 <b>Número de transistores</b> 2300	

Figura 6.10: Página de detalles de un componente (administración)

> Listado de periodos

> Añadir periodo

> Añadir componente a periodo existente

Añadir periodo

Nombre:

Nombre

Detalles:

Detalles

Sabías que...  
Introduzca cada característica en una línea

Sabías que...

Eventos:  
Introduzca cada evento en una línea

Eventos

Cancelar

Guardar y continuar

Figura 6.11: Formulario para añadir un periodo

> Listado de periodos  
 > Añadir periodo  
 > Añadir componente a periodo existente

### Editar periodo

←

Nombre: CPUs pre-x86

Detalles: El 4004 formaba parte de la familia de chips MCS, que complementaban sus funciones. Sus sucesores 4040, 8080 y 8085 tenían su propia familia de chips de soporte.

Sabías que...  
 Introduzca cada característica en una línea  
 Intel originalmente fabricaba solo chips de memoria y se inició en la fabricación de CPUs con estos modelos. Estos chips no tienen memoria caché ni tiene sentido hablar de velocidad de bus.

Eventos:  
 Introduzca cada evento en una línea  
 1970: Apollo 13  
 1971: email  
 1971: Floppy, pantallas LCD

Cancelar Guardar

Figura 6.12: Formulario para editar un periodo

> Listado de periodos  
 > Añadir periodo  
 > Añadir componente a periodo existente

### Añadir componente

Seleccione el periodo al que pertenece: CPUs pre-x86 Tipo: CPU

Nombre: Nombre

Descripción: Descripción

Familia: Familia

Año de salida: 1970 Año de fin: 1990 Precio: 100 \$

Utilizado en dispositivos: ☐ Portátiles ☐ De escritorio

Sistema famoso que lo utiliza: Nombre del sistema Subir imagen

Subir imágenes del componente

Memoria ROM: 0 Memoria RAM: 0

Frecuencia de reloj: 0 Consumo energético: 0

Tamaño de palabra: 0 Nanómetros de los transistores: 0 nm

Passmark: 0 Número de transistores: 0

Cancelar Guardar

Figura 6.13: Formulario para añadir un componente

> Listado de periodos  
> Añadir periodo  
> Añadir componente a periodo existente

### Editar componente

Selecione el periodo al que pertenece: CPUs pre-x86

Nombre: Intel 4004


Descripción: Desarrollado para calculadoras de la marca Busicom, se instaló en máquinas de Pinball y fue candidato a instalarse en la nave espacial Pioneer 10. Existe una réplica funcional de 41x58cm en el Intel Museum de California.

Familia: Intel

Año de salida: 1971 Año de fin: 1981 Precio: 1800,00 \$

Utilizado en dispositivos: ☒ Portátiles ☒ De escritorio

Sistema famoso que lo utiliza: Busicom 141PF



Memoria ROM:	4	KB	Memoria RAM:	640	b
Frecuencia de reloj:	740,000	KHz	Consumo energético:	0,450	W
Tamaño de palabra:	4	b	Nanómetros de los transistores:	10000	nm
Passmark:	0,030		Número de transistores:	2300	

Figura 6.14: Formulario para editar un componente



## 6.3. DSI 6: DISEÑO FÍSICO DE DATOS

### 6.3.1. 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:** almacena toda la información relativa a un periodo.
- **components:** contiene los datos que serían comunes a cualquier tipo de componente independientemente 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.

### 6.3.2. 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.

### 6.3.3. Diagrama E-R

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

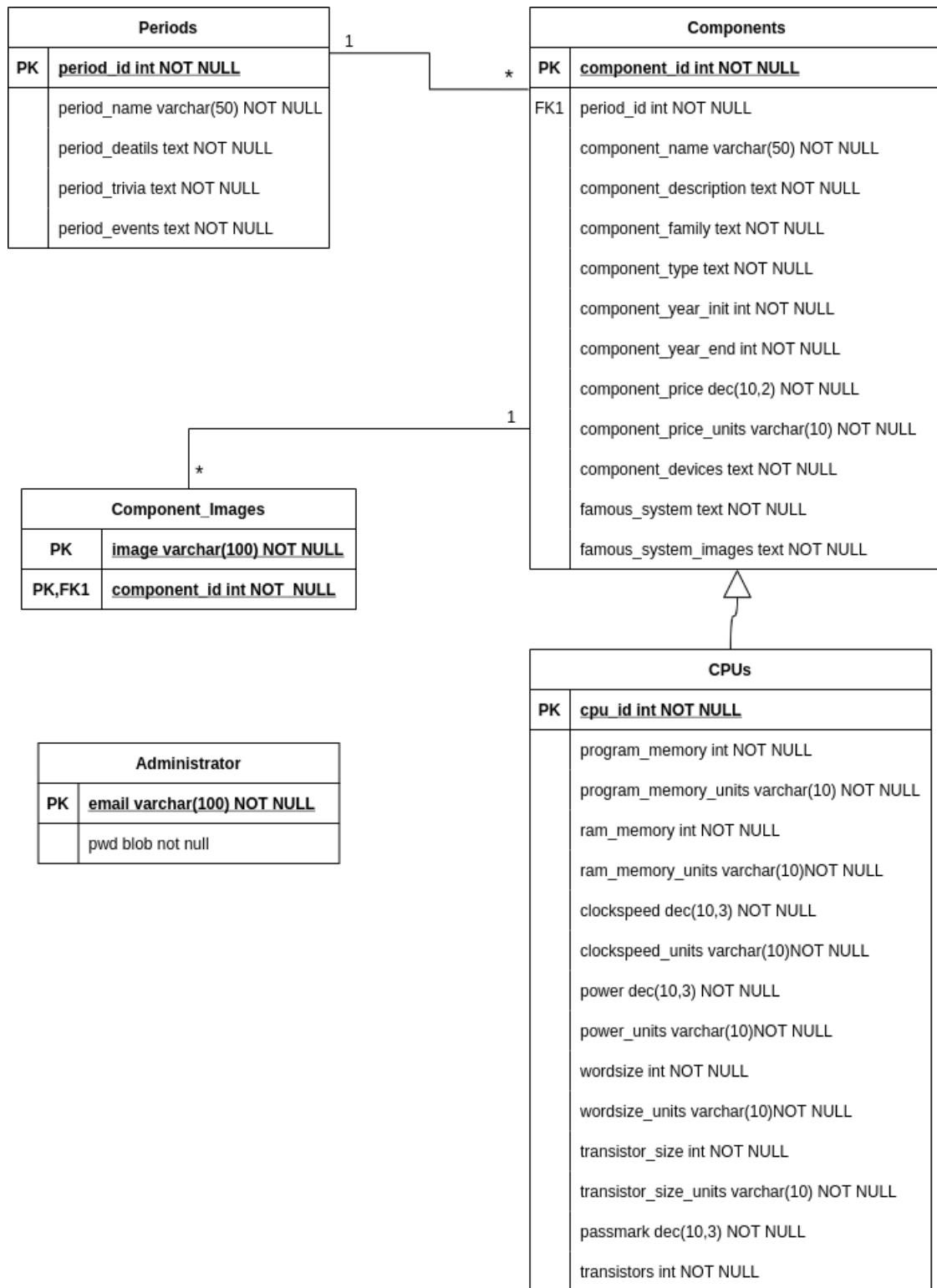


Figura 6.15: Diagrama Entidad-Relación de la base de datos creada

## **6.4. DSI 9: DISEÑO DE LA MIGRACIÓN Y CARGA INICIAL DE DATOS**

## **6.5. DSI 10: ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS**

### **6.5.1. Pruebas Unitarias**

### **6.5.2. Pruebas de Integración y del Sistema**

### **6.5.3. Pruebas de Usabilidad y Accesibilidad**

#### **6.5.3.1. Diseño de Cuestionarios**

#### **6.5.3.2. Actividades de las Pruebas de Usabilidad**

### **6.5.4. Pruebas de Accesibilidad**

## Capítulo 7

# CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN

### FASE DE DESARROLLO

## CSI

## **7.1. CSI 1: PREPARACIÓN DEL ENTORNO DE GENERACIÓN Y CONSTRUCCIÓN**

### **7.1.1. Estándares y normas seguidos**

#### **7.1.1.1. 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.

#### **7.1.1.2. HTML5**

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

#### **7.1.1.3. CSS**

Hojas de estilos estandarizadas por el W3C.

#### **7.1.1.4. PHP Code Style Guide**

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

### **7.1.2. Lenguajes de programación**

#### **7.1.2.1. TypeScript**

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

#### **7.1.2.2. HTML**

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

#### **7.1.2.3. CSS**

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

#### **7.1.2.4. PHP**

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

#### 7.1.2.5. 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.

### 7.1.3. Herramientas y programas usados para el desarrollo

#### 7.1.3.1. 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 7.1: Logo de Visual Studio Code

#### 7.1.3.2. XAMPP

XAMPP es una distribución de Apache gratuita que contiene MariaDB, PHP y Perl[10]. Fue usado inicialmente para trabajar con la base de datos y los PHP necesarios, pero una vez configurado el servidor Ubuntu 20.04 dejó de ser necesario.



Figura 7.2: Logo de XAMPP

#### 7.1.3.3. MobaXTerm

MobaXTerm permite trabajar con herramientas de red remotas, como SSH, utilizando una terminal Unix desde Windows. Se ha usado para configurar el servidor Ubuntu 20.04 que aloja el servidor MySQL con la base de datos del sistema, y el servidor Apache con los PHP que se utilizan para trabajar con esta base de datos.



Figura 7.3: Logo de MobaXTerm

#### 7.1.3.4. Git

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



Figura 7.4: Logo de Git



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

### Ejemplos de tablas descripción de clases

Tabla 7.1: 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	
<b>signInWithGoogle</b>	Hace una llamada al objeto Fire para el inicio de sesión con Firebase authentication mediante una cuenta de Google.
render	

Tabla 7.2: 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	
<b>emergencyCalling</b>	Es el método encargado de redirigir la aplicación hacia el marcador con el 112 marcado.
<b>warnProtectors</b>	[Falta implementar] Es el encargado de generar un mensaje de aviso a los protectores creando notificaciones push.
render	

### **7.3. CSI 3: EJECUCIÓN DE LAS PRUEBAS UNITARIAS**

## **7.4. CSI 4: EJECUCIÓN DE LAS PRUEBAS DE INTEGRACIÓN**

## **7.5. CSI 5: EJECUCIÓN DE LAS PRUEBAS DEL SISTEMA**

### **7.5.1. Prueba de Usabilidad**

### **7.5.2. Pruebas de Accesibilidad**

#### **7.5.2.1. Revisión Preliminar**

#### **7.5.2.2. Evaluación de Conformidad**

#### **7.5.2.3. Checklist del WCAG 2.1**

#### **7.5.2.4. Accesibilidad con Dispositivos Móviles**

## 7.6. CSI 6: ELABORACIÓN DE LOS MANUALES DE USUARIO

### 7.6.1. Manual de Instalación

En este manual se detallarán los pasos necesarios para realizar las instalaciones necesarias para la ejecución del sistema.

En primer lugar, es necesario instalar NodeJS (se puede descargar en <https://nodejs.org/en/download/>) y reiniciar el sistema, ya que con esta instalación se ha cambiado la configuración de variables del PATH.

Para los siguientes pasos, es necesario el uso de la terminal del sistema.

Usando npm, el gestor de paquetes de NodeJS, hay que instalar Angular CLI. Para ello hay que ejecutar el comando *npm install -g @angular/cli*.

```
C:\Users\isabe>npm install -g @angular/cli
```

Figura 7.5: Instalación de Angular CLI

Por último, desde la carpeta que ubica tanto el proyecto del museo (museo-eii) como el de la administración (museo-eii-admin), se ejecuta el comando *npm install* para instalar los paquetes necesarios.

```
C:\Users\isabe\Documents\TFG\museo-eii\museo-eii>npm install_
```

Figura 7.6: Instalación de los paquetes del proyecto del museo

```
C:\Users\isabe\Documents\TFG\museo-eii\museo-eii-admin>npm install_
```

Figura 7.7: Instalación de los paquetes del proyecto de administración

### 7.6.2. Manual de Ejecución

Una vez completada la instalación siguiendo los pasos descritos en el apartado anterior, se pueden ejecutar ambas aplicaciones utilizando el comando *ng serve -o*, *npm start* o *npm run ng serve -o*. Esto hará que la aplicación esté disponible en <http://localhost:4200>.

```

C:\Users\isabe\Documents\TFG\museo-eii\museo-eii>ng serve -o
10% building 3/3 modules 0 active(node:14092) [DEP0111] DeprecationWarning: Access to process.binding('http_parser') is
deprecated.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:14092) [DEP0148] DeprecationWarning: Use of deprecated folder mapping "." in the "exports" field module resolutio
n of the package at C:\Users\isabe\Documents\TFG\museo-eii\museo-eii\node_modules\tslib\package.json.
Update this package.json to use a subpath pattern like ".*".

chunk {main} main.js, main.js.map (main) 142 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 150 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 964 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.38 MB [initial] [rendered]
Date: 2022-04-27T12:24:42.301Z - Hash: f4defa2709d07765124e - Time: 6678ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.

```

Figura 7.8: Ejecución de la aplicación del museo

```

C:\Users\isabe\Documents\TFG\museo-eii\museo-eii-admin>ng serve -o
10% building 3/3 modules 0 active(node:56048) [DEP0111] DeprecationWarning: Access to process.binding('http_parser') is
deprecated.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:56048) [DEP0148] DeprecationWarning: Use of deprecated folder mapping "." in the "exports" field module resolutio
n of the package at C:\Users\isabe\Documents\TFG\museo-eii\museo-eii-admin\node_modules\tslib\package.json.
Update this package.json to use a subpath pattern like ".*".

chunk {main} main.js, main.js.map (main) 255 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 150 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 1.11 MB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 5 MB [initial] [rendered]
Date: 2022-04-27T12:25:28.123Z - Hash: 6832c2cf5609987bce1d - Time: 8925ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.

```

Figura 7.9: Ejecución de la aplicación de administración

### 7.6.3. Manual de Usuario

#### 7.6.3.1. Museo

Al acceder a la web observamos la página de inicio. La parte superior de esta página está presente en toda la aplicación web y, por orden de izquierda a derecha, observamos:

- El logo de la escuela, que redirige a esta página de inicio.
- Museo, que redirige a la vista general del museo.
- Acerca de.
- Un selector de idioma, que permite cambiar entre inglés y español.

La parte inferior, que contiene enlaces a las redes sociales de la escuela, también está presente en toda la aplicación web.

En la parte central se encuentra el contenido específico de la página de inicio: una bienvenida a la página web y un botón que nos dirige a la vista general del museo.

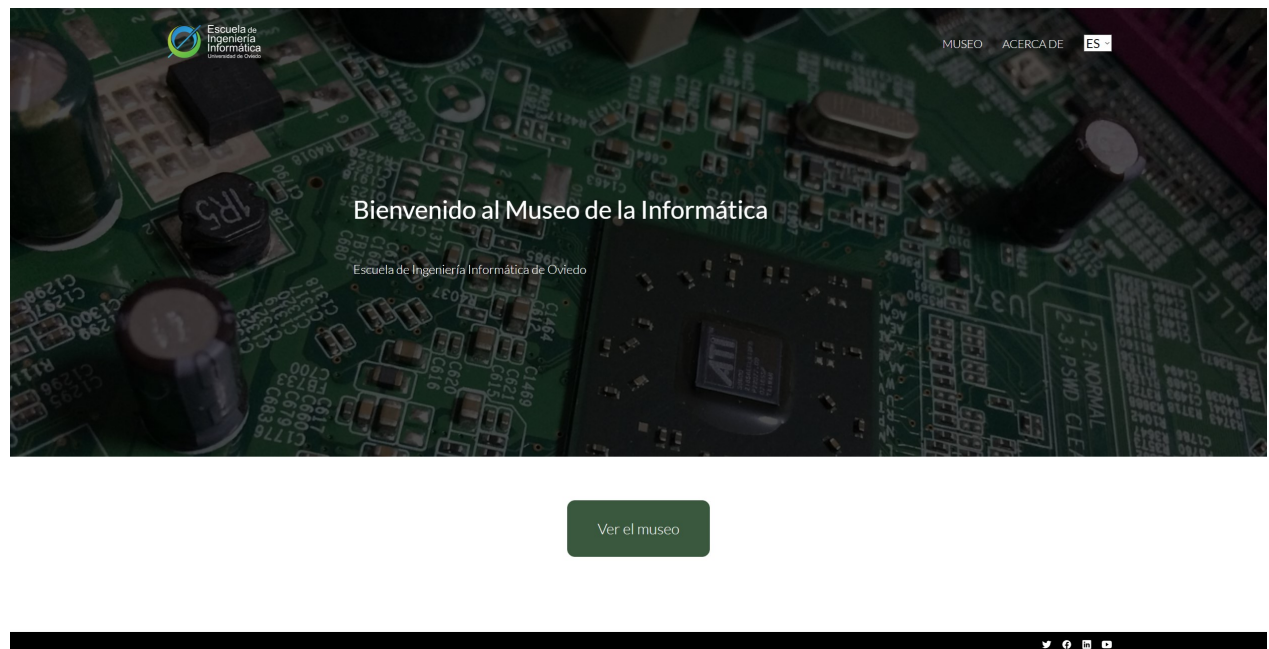


Figura 7.10: Manual de usuario: Inicio

En la vista general del museo hay una línea temporal y filtros de búsqueda.

En cada elemento de la línea temporal se muestra el nombre del periodo con un enlace al mismo, los años que comprende dicho periodo, y los nombres de los componentes pertenecientes al periodo, también con enlaces a cada uno de ellos.

La búsqueda puede realizarse por años o por nombre. Se puede filtrar por años mediante la barra deslizadora, y se mostrarán entonces todos aquellos periodos que, parcialmente o en su totalidad, tengan componentes pertenecientes a esos años. La búsqueda por nombre se realiza tras escribir en el recuadro de búsqueda y pulsar la tecla *Enter*, y el resultado será aquellos periodos cuyo nombre o el nombre de alguno de sus componentes contenga el texto buscado.

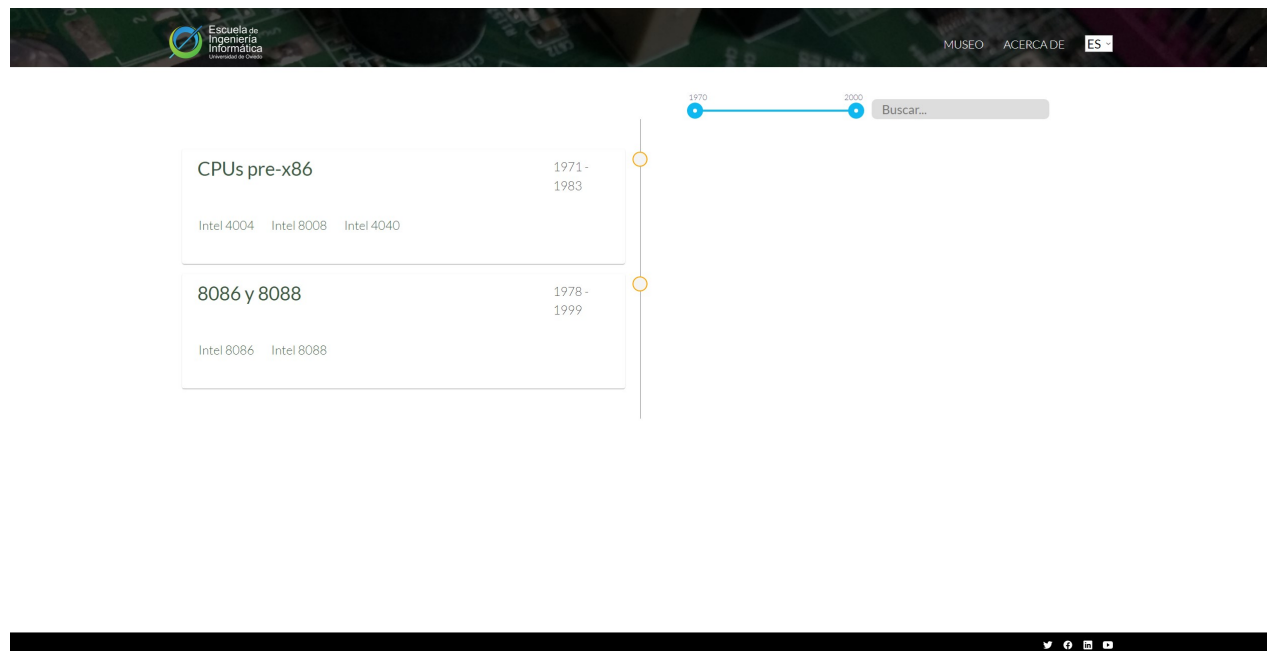


Figura 7.11: Manual de usuario: Vista general del museo

Al entrar en un periodo, en la parte superior podemos ver un menú, en la izquierda se mostraría el periodo anterior cronológicamente, y en la izquierda el periodo siguiente (si existen). El contenido principal de la página son los detalles del periodo: nombre, características, una lista de curiosidades (sabías que...), eventos informáticos ocurridos en esos años, los componentes pertenecientes al periodo (mostrando una imagen y el nombre, con un enlace al componente), y una serie de sistemas famosos que llevan esos componentes.

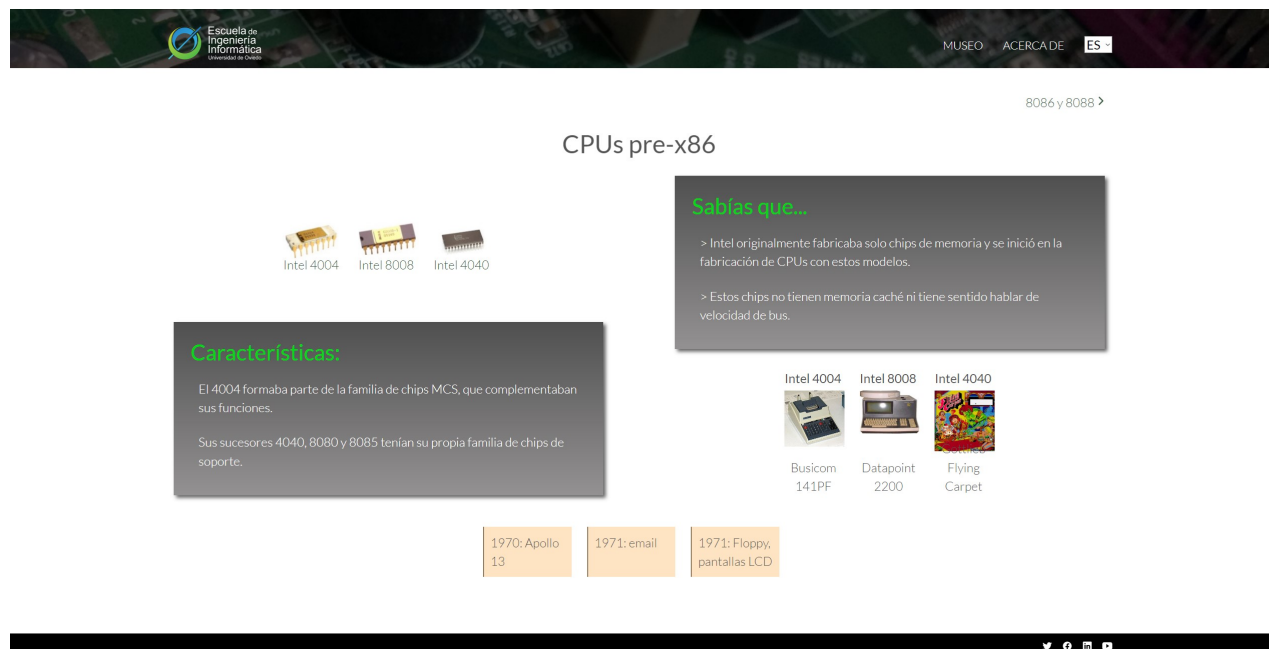


Figura 7.12: Manual de usuario: Detalles del periodo (museo)



Por último, al acceder a un componente, podemos ver una galería de fotos que se abrirán en grande al pulsar sobre ellas, una descripción del componente y un listado de características. Además, en el lateral izquierdo hay un menú que permite navegar entre periodos (ver el anterior, el actual y el siguiente) y entre los componentes del periodo actual.

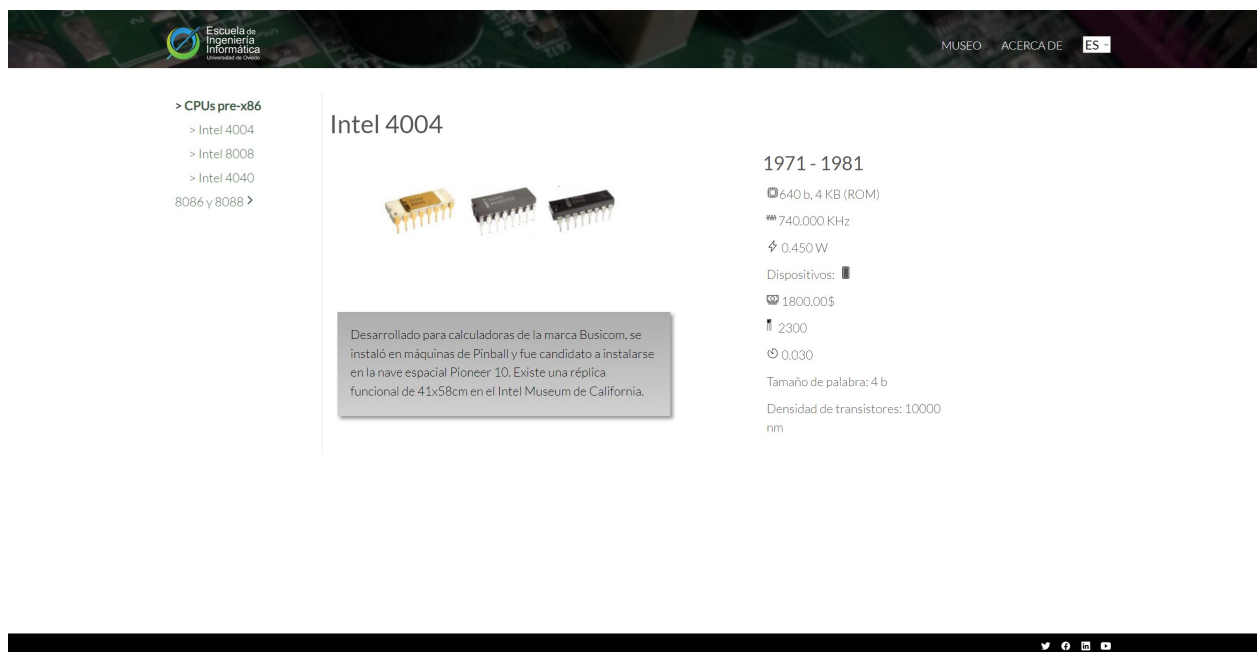


Figura 7.13: Manual de usuario: Detalles del componente (museo)

### 7.6.3.2. Administración del museo

Al entrar a la web de administración del museo nos encontramos con el inicio de sesión. Es necesario indicar el correo electrónico y la contraseña para acceder.

**Museo de la Informática - Administrador**

**Iniciar sesión**

Figura 7.14: Manual de usuario: Inicio de sesión

Lo primero que se muestra una vez iniciada la sesión es un listado de los periodos existentes, mostrando sus nombres con un enlace a cada uno de ellos, y permitiendo editar y eliminar cada periodo. Eliminar un periodo borrará también los componentes asociados al mismo, para ello se mostrará un aviso y se pedirá confirmación. En el lateral izquierdo hay un menú que se incluye en todas las páginas de la aplicación, desde el que se puede acceder a este listado de periodos, y a los formularios para añadir periodos y componentes.

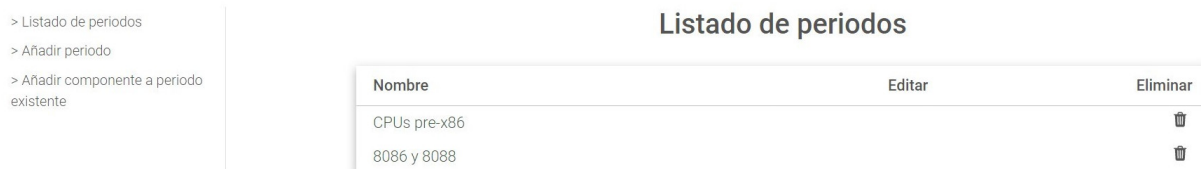


Figura 7.15: Manual de usuario: Listado de periodos

Los detalles de un periodo y del componente muestran los mismos datos explicados anteriormente en el manual de usuario del museo, con la diferencia de en que cada una de estas páginas se muestra una opción para editar el periodo o el componente que estemos visualizando, y en el listado de componentes del periodo también se da la opción de editar o eliminar cada uno de ellos.

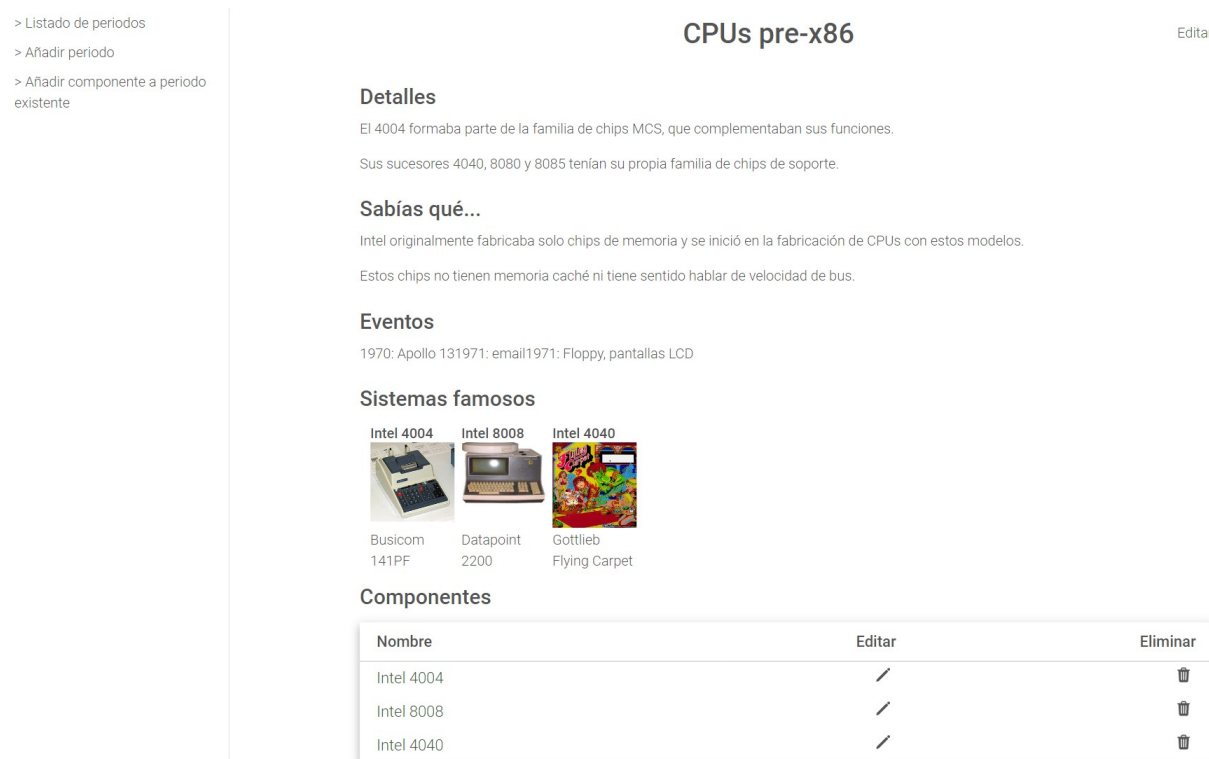


Figura 7.16: Manual de usuario: Detalles de un periodo (administración)

> Listado de periodos

> Añadir periodo

> Añadir componente a periodo existente

## Intel 4004

Editar

**←**

**Periodo**  
CPUs pre-x86

**Familia**  
Intel

**Años**  
1971 - 1981

**Descripción**  
Desarrollado para calculadoras de la marca Basicom, se instaló en máquinas de Pinball y fue candidato a instalarse en la nave espacial Pioneer 10. Existe una réplica funcional de 41x58cm en el Intel Museum de California.

**Memoria ROM**  
4 KB

**Memoria RAM**  
640 b

**Frecuencia de reloj**  
740.000 KHz

**Consumo energético**  
0.450 W

**Precio**  
1800.00\$

**Dispositivos**  
Portátiles. De escritorio.



**Tamaño de palabra**  
4 b

**Nanómetros de los transistores**  
10000

**Passmark**  
0.030

**Número de transistores**  
2300

Figura 7.17: Manual de usuario: Detalles de un componente (administración)

En el formulario de añadir un periodo hay cuatro entradas de texto para nombre, detalles, curiosidades y eventos del periodo. Todos ellos deben rellenarse obligatoriamente para poder guardar el periodo. Si se pulsa el botón *Cancelar*, el formulario se vaciará de nuevo. Al pulsar *Guardar y continuar* con el formulario completo, se añadirá el periodo a la base de datos del sistema y nos redigirá al formulario para añadir componentes. En cambio, si el formulario no es válido se mostrará un error y no se añadirá.

> Listado de periodos

> Añadir periodo

> Añadir componente a periodo existente

## Añadir periodo

Nombre:

Detalles:

Sabías que...  
Introduzca cada característica en una línea

Eventos:  
Introduzca cada evento en una línea

Cancelar

Guardar y continuar

Figura 7.18: Manual de usuario: Formulario para añadir un periodo

A la hora de editar un periodo, el formulario funcionará igual que al añadirlo, con la diferencia de que los valores iniciales serán los del periodo que se está editando.

> Listado de periodos  
> Añadir periodo  
> Añadir componente a periodo existente

### Editar periodo

←

Nombre: CPUs pre-x86

Detalles: El 4004 formaba parte de la familia de chips MCS, que complementaban sus funciones. Sus sucesores 4040, 8080 y 8085 tenían su propia familia de chips de soporte.

Sabías que...  
Introduzca cada característica en una línea Intel originalmente fabricaba solo chips de memoria y se inició en la fabricación de CPUs con estos modelos. Estos chips no tienen memoria caché ni tiene sentido hablar de velocidad de bus.

Eventos:  
Introduzca cada evento en una línea  
1970: Apollo 13  
1971: email  
1971: Floppy, pantallas LCD

Cancelar Guardar

Figura 7.19: Manual de usuario: Formulario para editar un periodo

Los formularios para añadir y editar componentes funcionan de la misma forma que los de añadir y editar periodos, pero en este caso, hay campos que no son obligatorios, como la subida de imágenes y el sistema famoso. Además, al añadir o editar componentes se puede seleccionar su tipo: CPU o componente genérico. Al seleccionar CPU se muestran los campos de memoria ROM, memoria RAM, frecuencia de reloj, consumo energético, tamaño de palabra, nanómetros de transistores, passmark y número de transistores.

### Añadir componente

Seleccione el periodo al que pertenece: CPUs pre-x86 Tipo: CPU

Nombre: Nombre

Descripción: Descripción

Familia: Familia

Año de salida: 1970 Año de fin: 1990 Precio: 100 \$

Utilizado en dispositivos: ☐ Portátiles ☐ De escritorio

Sistema famoso que lo utiliza: Nombre del sistema Subir imagen

Subir imágenes del componente

Memoria ROM: 0 Memoria RAM: 0

Frecuencia de reloj: 0 Consumo energético: 0

Tamaño de palabra: 0 Nanómetros de los transistores: 0 nm

Passmark: 0 Número de transistores: 0

Cancelar Guardar

Figura 7.20: Manual de usuario: Formulario para añadir un componente

- > Listado de periodos
- > Añadir periodo
- > Añadir componente a periodo existente

## Editar componente

Seleccione el periodo al que pertenece: CPUs pre-x86

Nombre: Intel 4004

Descripción: Desarrollado para calculadoras de la marca Busicom, se instaló en máquinas de Pinball y fue candidato a instalarse en la nave espacial Pioneer 10. Existe una réplica funcional de 41x58cm en el Intel Museum de California.





Familia: Intel

Año de salida: 1971 Año de fin: 1981 Precio: 1800,00 \$

Utilizado en dispositivos: ☒ Portátiles ☒ De escritorio

Sistema famoso que lo utiliza: Busicom 141PF Subir imagen

Subir imágenes del componente

Memoria ROM: 4 KB

Frecuencia de reloj: 740,000 KHz

Tamaño de palabra: 4 b

Passmark: 0,030

Memoria RAM: 640 b

Consumo energético: 0,450 W

Nanómetros de los transistores: 10000 nm

Número de transistores: 2300

Cancelar
Guardar

Figura 7.21: Manual de usuario: Formulario para editar un componente

### 7.6.4. Manual del Programador

En este manual se explicará cómo ampliar la aplicación añadiendo nuevos tipos de componentes además de CPUs. Primero habría que crear una nueva clase para cada tipo que se desee añadir. Cada una de estas clases implementarán la interfaz *MyComponent* y heredarán de *GenericComp*. También habría que actualizar la enumeración *CompTypes*. Estos tres elementos mencionados se encuentran en el archivo *comp.ts*, que forma parte tanto del proyecto del museo (museo-eii) como de la administración (museo-eii-admin). Una vez realizado esto, común a ambos proyectos, se explicará qué debe añadirse a cada uno de ellos en específico, así como a la base de datos.

#### 7.6.4.1. Museo

En el proyecto del museo (museo-eii) deberá generarse un componente de Angular para cada tipo añadido, se llamará *'new type'DetailsComponent* y será hijo de *CompDetailsComponent*, del que recibirá como input el atributo *comp*. Este solo se mostrará cuando *comp* sea una instancia del tipo correspondiente a *'new type'*. En *'new type'-details.component.html* se listarán las características de *comp*.

Además, en el método *getComp* de *CompDetailsComponent* habrá que añadir las comprobaciones necesarias para mostrar los nuevos tipos definidos.

#### 7.6.4.2. Administración del museo

En el proyecto de la administración (museo-eii-admin) habrá que generar dos componentes de Angular nuevos por cada tipo añadido:

- ‘new type’*FormComponent*, hijo de *AddCompComponent* y de *EditCompComponent*. De ambos recibe como input el atributo *model*. En ‘new type’-*form.component.html* se incluirán los campos del formulario que se correspondan con las características del tipo creado. Se mostrará cuando *model* sea una instancia del tipo correspondiente a ‘new type’.

En el método *createModel* de *AddCompComponent* habrá que añadir la opción de crear un objeto de este nuevo tipo, y también se añadirán las comprobaciones necesarias en los métodos *isValid* y *cloneComp* de *AddCompComponent* y *EditCompComponent*.

- ‘new type’*DetailsComponent*, hijo de *MyComponentComponent*, del que recibe como input el atributo *c*. En este caso, se hará exactamente lo mismo que lo mencionado anteriormente al añadir ‘new type’*DetailsComponent* en el proyecto del museo, ya que ambos componentes son para mostrar las características de cada tipo.

#### 7.6.4.3. Base de datos

En la base de datos habría que crear una tabla por cada nuevo tipo de componente, con los campos necesarios para este que no estén ya incluidos en la tabla *components*. La clave primaria de esta tabla sería también una clave foránea, el identificador del componente en la tabla *components*. Una vez creadas las tablas correspondientes, habría que modificar las comprobaciones y consultas realizadas en los archivos *getComp.php*, *updateComp.php* y *postComp.php* para incluir los nuevos tipos creados.

## **7.7. CSI 8: CONSTRUCCIÓN DE LOS COMPONENTES Y PROCEDIMIENTOS DE MIGRACIÓN Y CARGA INICIAL DE DATOS**

## Capítulo 8

# IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA

### FASE DE DESARROLLO

## IAS



## **8.1. IAS 1: ESTABLECIMIENTO DEL PLAN DE IMPLANTACIÓN**

## **8.2. IAS 4: CARGA DE DATOS AL ENTORNO DE OPERACIÓN**

### **8.3. IAS 5: PRUEBAS DE IMPLANTACIÓN DEL SISTEMA**

## **8.4. IAS 7: PREPARACIÓN DEL MANTENIMIENTO DEL SISTEMA**

## **8.5. IAS 8: ESTABLECIMIENTO DEL ACUERDO DE NIVEL DE SERVICIO**

## **8.6. IAS 9–10: PRESENTACIÓN Y APROBACIÓN DEL SISTEMA Y PASO A PRODUCCIÓN**

## Capítulo 9

# CONCLUSIONES Y AMPLIACIONES

## **9.1. CONCLUSIONES**



## **9.2. 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 encuentran 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](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.