



MEMORIA

DEL BACK AL END

Martín Ordoño Caro

77850032T

Trabajo de Fin de Grado

Tutora: Inmaculada Hernández Salmerón

Septiembre, segunda convocatoria, curso 2019/2020

Grado en Ingeniería Informática, Ingeniería del Software

Escuela Técnica Superior de Ingeniería Informática

Índice

1. Glosario de términos.....	4
2. Resumen.....	5
3. Introducción	6
4. Gestión del proyecto	7
4.1. Alcance.....	7
4.2. Objetivos	10
4.3. Planificación.....	11
4.4. Metodología	17
4.5. Herramientas	18
4.5.1. GitHub	18
4.5.2. Entornos de desarrollo	18
4.5.3. Frameworks	19
4.6. Relación con las asignaturas del Grado.....	21
5. Realización del proyecto.....	23
5.1. Análisis.....	23
5.2. Backend.....	25
5.2.1. Diseño.....	25
5.2.2. Implementación	26
5.2.3. Pruebas	32
5.3. Frontend	32
5.3.1. Diseño.....	32
5.3.2. Implementación	34

5.3.3. Pruebas	39
6. Manual de usuario.....	40
7. Conclusión	43
Anexo I – Control de cambios.	45

1. Glosario de términos.

Frontend: Parte de una aplicación web que contiene la interfaz gráfica con la que interactúa el usuario. Se corresponde con la parte *cliente*.

Backend: Parte de una aplicación web que se encarga de administrar la base de datos y donde se incluye la lógica de negocio. Se corresponde a la parte *servidor*.

Serie de datos: En el contexto de este documento, las series de datos son las series estadísticas que se quieren extraer de las distintas fuentes para ser almacenadas y mostradas por la aplicación. Estas series son conjuntos de datos recogidos por el Estado y que se encuentran públicos en internet.

API: Interfaz de programación de aplicaciones, llamada comúnmente API. Es el conjunto de endpoints que ofrece un software para ser usado por otro software como una capa de abstracción. En el contexto de la aplicación, la API supone el conjunto de endpoints expuestos por el backend para que el frontend interactúe con él.

Endpoint: Función concreta de una API. Cada endpoint debe tener una ruta única para ser diferenciados entre ellos dentro de una misma API.

Scraper: Programa que se encarga de extraer información específica de una página web, normalmente recorriendo el código HTML, manteniendo la información deseada y desechando la demás.

Crawler: Programa que inspecciona las páginas web de forma sistemática y ordenada, recuperando el contenido de cada una de las páginas.

CORS: “Intercambio de Recursos de Origen Cruzado”. Es un mecanismo característico de las aplicaciones web que restringe el acceso a recursos si son solicitados desde un dominio distinto. Esto sucede a menudo en la arquitectura desacoplada, por lo que hay que configurar el CORS para permitir que el frontend acceda a los recursos del backend.

NoSQL: Tipo de bases de datos que se caracteriza por su flexibilidad y por no contemplar relaciones entre tablas de datos. Son reconocidas por su fácil desarrollo y su rendimiento manejando grandes cantidades de datos.

Reactividad: Capacidad de un software de mantener un entorno actualizado de manera asíncrona.

2. Resumen

Este proyecto se centra en la extracción de series estadísticas de datos de varias fuentes para almacenar y mostrar la información de diferentes maneras. Las series estadísticas, en general, son conjuntos de datos asociados a una secuencia temporal y, en este proyecto, concretamente, las series que se tratan son series publicadas por el estado que recogen datos sobre la sociedad a lo largo de los años. Es interesante comparar estos datos ya que los usuarios serán capaces de mostrar las diferencias entre un periodo de tiempo y otro o entre un territorio y otro, entre otras cosas. Por tanto, los usuarios a los que está orientado este proyecto son usuarios pertenecientes a la administración pública o cualquier persona con interés en comparar series estadísticas de este tipo, los cuales encontrarán un gran valor en una aplicación que unifique distintas fuentes de datos en un mismo lugar y además permita realizar comparaciones entre las series.

Para la realización de este proyecto, se creará una aplicación con arquitectura desacoplada diferenciando entre frontend y backend. Contando además con los distintos scrapers necesarios para la extracción de los datos. Es cierto que, anteriormente, el alumno ha trabajado en aplicaciones con arquitectura desacoplada, pero nunca ha realizado por sí solo ningún proyecto completo, incluyendo además scrapers complejos que han de extraer grandes cantidades de información. Esto supone un gran avance tanto profesional como personal: profesional ya que realizar este proyecto implica aumentar los conocimientos en esta arquitectura muy usada en ámbito profesional, y personal al demostrar y consolidar los conocimientos adquiridos durante el grado.

Las características más importantes de este proyecto son los distintos scrapers personalizados que han de extraer los datos de las distintas páginas web junto al frontend, que deberá mostrar los datos de forma gráfica.

A lo largo de este documento se detallan las distintas tecnologías usadas para la realización del proyecto, de la cual se destaca Svelte, un framework web muy joven con buenas cualidades que puede llegar a dar mucho rendimiento.

Además, en este documento se encuentra toda la información acerca del alcance y la planificación del proyecto, las metodologías que se usan durante el desarrollo, se detalla la implementación del frontend y backend por separado, y se explican todos los problemas encontrados.

3. Introducción

El epicentro de este proyecto son las series temporales. Ampliando la explicación del apartado anterior, estas series son conjuntos de datos recogidos a lo largo de los años por distintos organismos del estado, y son publicados en distintas páginas webs. Pero estas series no son constituidas únicamente por sus conjuntos de valores, sino que las series pueden variar en el territorio, tipo de dato (valor real o porcentaje), fuente o periodicidad, la cual puede ser desde anual hasta diaria, aunque este proyecto no se centra en las series con una frecuencia tan alta. Pese a que estas son las características clave, según la fuente, la información de las series puede ser incluso más compleja.

Al tener tantos atributos distintos y proceder de organizaciones diferentes, el formato de las series varía en mayor o menor medida de una fuente a otra, lo que hace que la extracción de datos se vuelva una tarea de complejidad alta. Esta tarea no solo comprende la extracción de datos, sino que es necesario que se inserten las distintas series estadísticas con formatos diferentes dentro de una base de datos con una estructura única. Esto es necesario para realizar una comparación efectiva, lo cual es uno de los objetivos principales de este proyecto. No obstante, si se realiza un buen scraper, siempre que el formato de los datos no cambie, se podrán extraer los datos a voluntad con el único inconveniente del tiempo debido al volumen de datos.

Se quiere conseguir comparar series de datos de distintas fuentes, tanto de forma visual como los datos numéricos. Sin embargo, este objetivo se puede fragmentar en distintos apartados, todos ellos centrados en las series de datos: Extracción y almacenamiento; recuperación y entrega de datos; y, por último, visualización y exportación. Estos tres apartados se corresponderían a los scrapers, el backend y el frontend, respectivamente.

Por lo tanto, para llegar a la solución, se ha de conseguir una aplicación que cumpla con los tres apartados anteriores. Además, al optar por una arquitectura desacoplada, el backend puede ser usado como una interfaz con la que aplicaciones de terceros puedan interactuar.

El resultado de realizar este proyecto, siguiendo la solución descrita, sería un sistema que permita a los usuarios comparar las series de distintas fuentes de manera sencilla y rápida al encontrarse dentro de la misma aplicación. Además, al usar una API, se podrán

construir más aplicaciones de terceros que le den uso al backend desarrollado, aumentando el valor de este proyecto.

Por último, el desarrollo de este proyecto incluye tecnologías muy variadas, desde los scrapers hasta el frontend, pasando por la base de datos y el backend, por lo que el alumno ha de enfrentarse a todas las dificultades que aparezcan durante este desarrollo tan variado. Esto supone un aumento significativo en la madurez como desarrollador a nivel personal, además de que, al tratarse de tecnologías usadas en el mundo laboral, sobre todo la arquitectura desacoplada, el frontend basado en JavaScript, y una base de datos NoSQL especialmente útil para este proyecto, significa un aumento de capacidades y conocimientos que podrán suponer una ventaja para el alumno a la hora de encontrar trabajo.

4. Gestión del proyecto

4.1. Alcance

Objetivo del proyecto

El objetivo del proyecto es desarrollar una aplicación con diversas funcionalidades: la primera es extraer la información de distintas fuentes y unificarlos en una misma estructura de datos, ya que si no, las demás características serían inútiles; la segunda es crear un sistema de acceso a dicha estructura de datos que sea capaz de recuperar los datos según distintos criterios; y por último, una interfaz de usuario que permita navegar entre las series almacenadas, visualizarlas gráficamente o en tablas y además, permita la descarga de los datos.

La aplicación contará con distintos scrapers que sean los encargados de extraer y almacenar las series de datos de las distintas fuentes especificadas por el cliente. Una vez los datos estén almacenados, la aplicación deberá mandar las series de datos al frontend. Allí, los usuarios tendrán a su disposición el conjunto de series provenientes de distintas fuentes y serán capaces de elegir cuales comparar tanto gráficamente como numéricamente. Además, los usuarios tendrán la opción de exportar las comparaciones que realicen con solo pulsar un botón, cumpliendo así con los objetivos descritos.

Restricciones

A continuación, se definen las restricciones del proyecto a desarrollar. Estas restricciones limitan:

- **Tiempo:** es imprescindible que se cumpla la fecha de entrega.
- **Calidad:** el producto desarrollado tiene que cumplir con un mínimo de calidad que satisfaga tanto al cliente como al tutor de este proyecto.
- **Riesgos:** los riesgos contemplados en el proyecto deben tenerse en cuenta para su correcto desarrollo.
- **Costes:** al ser un proyecto sin presupuesto inicial, las herramientas y tecnologías usadas han de ser gratuitas. El único gasto que se contempla es el del hosting y dominio si se desplegase la aplicación y no fuese un coste alto.

Riesgos

Lista de riesgos posibles durante la vida del proyecto, cada riesgo ha de tenerse en cuenta para evitar contratiempos:

- **Planificación:** Si no se estima y planifica adecuadamente, es probable que no se cumplan los objetivos para la fecha acordada.
- **Tecnología:** Al usar un framework desconocido, es posible que el proyecto no se pueda desarrollar acorde a los requisitos establecidos o que las tareas de investigación se alarguen en exceso.
- **COVID-19:** Si se contrae el virus esto provocaría inevitablemente un retraso en el proyecto, como mínimo.
- **Pérdida de datos:** Existe la posibilidad de que se pierda información debido al malfuncionamiento de algún software o el fallo en el disco duro. Además, incidentes naturales o de caída de energía pueden provocar la pérdida de datos.

Riesgo	Impacto	Probabilidad	Nivel de riesgo
Planificación	4	4	8
Tecnología	3	7	10
COVID-19	10	1	11
Pérdida de datos	10	2	12

El nivel de riesgo es el resultado de la suma del impacto y la probabilidad, y representa lo grave que es el riesgo, por lo que, a mayor nivel, más se deberá tener en cuenta.

Medidas de control

- **Planificación:** Se realizará una planificación pesimista, dejando margen suficiente a las tareas hasta adquirir la capacidad de realizar estimaciones más aproximadas.
- **Tecnología:** Se realizarán tareas de investigación previas al desarrollo con las tecnologías desconocidas.
- **COVID-19:** Se tomarán las medidas de seguridad propuestas por el Estado, minimizando así la posibilidad de contraer el virus.
- **Pérdida de datos:** Se usará un repositorio privado en GitHub para almacenar todos los documentos y proyectos del producto, actualizándose en cada incremento. Esto reduce el impacto al mínimo, ya que únicamente se podrán perder las modificaciones realizadas al ocurrir la incidencia.

Criterios de aceptación

El proyecto será aceptado únicamente si se cumplen los siguientes criterios de aceptación:

- Es entregado dentro del plazo correspondiente.
- Se extraen series de datos, como mínimo, de dos fuentes distintas.
- La información extraída es almacenada en una única estructura de datos.
- Se desarrolla un backend con una API funcional.

- Se desarrolla un frontend que de uso a la API y cumpla con las funciones establecidas.

Perfil de usuario objetivo

Como se comenta anteriormente, los usuarios del sistema van a ser personas pertenecientes a la administración pública o con interés en comparar series estadísticas de este tipo. Por lo tanto, se puede asumir que el usuario final del producto domina los conceptos de las series de datos, por lo que no es necesario realizar una interfaz muy limitada para que los usuarios no hagan un mal uso de ella.

4.2. Objetivos

El Trabajo de Fin de Grado tiene como objetivo proporcionar al alumno una visión general y completa de la planificación, gestión y metodologías aplicables a un proyecto informático. Al tener una carga de 12 créditos ECTS, debe ser un trabajo con una complejidad considerable. Por lo tanto, uno de los objetivos principales de este proyecto es desarrollar un software completo de principio a fin, resumiendo todo lo aprendido a lo largo del grado.

Al ser un proyecto individual, entre los objetivos también se incluyen las metas personales del alumno, destacando un mayor interés por aprender tecnologías nuevas desconocidas por el mismo. Esto se tendrá en cuenta a la hora de elegir las tecnologías para el producto, a pesar de que usar tecnologías más novedosas puede acarrear inconvenientes.

Este es un trabajo desarrollado para un cliente, el cual quiere un producto útil y de una calidad mínima. Por tanto, es importante desarrollar un buen producto con el que impresionar al cliente, ya que puede llegar a ser un factor determinante para el comienzo de la vida laboral del alumno.

Al ser la última asignatura del grado, esta asignatura, junto con las prácticas, son el enlace entre la vida universitaria y la laboral, por lo que elaborar un producto de calidad puede ser un factor determinante a la hora de encontrar un buen puesto y dar a conocer las habilidades del alumno.

Por otro lado, se ha de desarrollar una aplicación completamente funcional, que cumpla con los requisitos establecidos, es decir, todos los scrapers desarrollados han de extraer la información y adaptarla a la estructura diseñada sin alterar su contenido; ha de desarrollarse también un backend que recupere dicha información de la base de datos y la exponga mediante una API; y, por último, ha de implementarse un frontend intuitivo y fácil de usar que permita a los usuarios realizar las distintas funciones especificadas en los requisitos. Es necesario que el backend esté bien implementado y que tenga un buen manejo de excepciones, ya que sería un gran fallo que los usuarios de la API reciban un error sin saber si lo están provocando ellos o si es un fallo interno del sistema. Además, también es necesario que la API esté bien documentada, ya que, como se comenta anteriormente, podrá llegar a ser usada por terceros, los cuales necesitarán saber los endpoints disponibles y sus usos.

4.3. Planificación

A pesar de no seguir una metodología ágil, el desarrollo de este proyecto se divide en tres fases principales. Cada una de las fases tiene un propósito distinto y está bien definida en el tiempo. En la primera fase se realizarán las tareas de investigación, el propósito de esta fase es determinar los frameworks y herramientas que se van a usar (especialmente en el frontend) y adquirir una base para poder comenzar a trabajar con ellos. El objetivo de la segunda fase es construir un prototipo con las funcionalidades clave, este prototipo debe ser completo para poder hacer una demostración al cliente y recibir buen feedback. El objetivo de la tercera y última fase es completar el producto, para ello se tendrá en cuenta el feedback recibido al final de la segunda fase. Además, se implementarán las funcionalidades extras y se mejorará tanto el apartado visual del producto como la API.

Primera fase: Inicio

Como se indica en el párrafo anterior, la finalidad de esta fase es determinar las tecnologías que se van a usar y adquirir cierto conocimiento básico en aquellas que sean desconocidas. Para ello se ha de realizar un análisis de las herramientas y framework propuestos y valorar cuáles son las mejores opciones.

Segunda fase: Prototipo

El objetivo de esta fase es implementar las funcionalidades principales. Para ello, se creará un scraper para una de las fuentes de datos, INDEA, y se implementará toda la funcionalidad desde el almacenamiento en base de datos hasta la visualización de las series de datos en el frontend.

Tercera fase: Finalización

A lo largo de esta fase, se completará el producto con las tareas detectadas en el feedback obtenido del cliente, además de cualquier funcionalidad extra propuesta. En el caso de que no diese tiempo a implementar alguna funcionalidad en la fase anterior, esta pasaría como tarea de esta fase con una prioridad superior a las demás.

Tareas

A continuación, se encuentra el listado de tareas. Aquí se definen los tiempos de inicio y finalización de las fases, las descripciones de las tareas, sus prioridades y sus tiempos estimados. Las tareas que se encuentran en este backlog son tareas generales que se dividirán en subtareas antes de ser desarrolladas.

Primera fase: Inicio		
Fecha inicio: 01/07/2020		Fecha fin: 07/07/2020
TI.01	Determinar herramientas y frameworks a usar.	A
	<ul style="list-style-type: none">Investigación de los distintos frameworks	5h
TI.02	Obtener conocimientos básicos sobre las herramientas elegidas desconocidas.	A
	<ul style="list-style-type: none">Svelte	3h
	<ul style="list-style-type: none">MongoDB	1h
Segunda fase: Prototipo		

Fecha Inicio: 08/07/2020		Fecha fin: 28/07/2020
TP.01	Inicialización del proyecto backend	A
	<ul style="list-style-type: none"> Crear proyecto base con Spring Initializr. 	2h
	<ul style="list-style-type: none"> Instalar dependencias necesarias (POM). 	1h
TP.02	Tabla de series.	A
	<ul style="list-style-type: none"> Componente para mostrar las series en una tabla en Svelte. 	5h
	<ul style="list-style-type: none"> Paginación en la tabla. 	5h
	<ul style="list-style-type: none"> Endpoint con paginación en la API. 	1h
TP.03	Almacenamiento de datos del sistema sobre series y sus datos en MongoDB.	A
	<ul style="list-style-type: none"> Dominio java para <i>Serie</i> con la siguiente información: código, periodo, fuente y colección de datos. 	2h
	<ul style="list-style-type: none"> Dominio java para <i>Dato</i> con la siguiente información: momento y valor. 	2h
	<ul style="list-style-type: none"> Repositorios y servicios para <i>Serie</i> y <i>Dato</i> 	2h
	<ul style="list-style-type: none"> BDD MongoDB y conectar con los repositorios anteriores 	2h
TP.04	Scraper INDEA	A
	<ul style="list-style-type: none"> Scraper que recolecte series de INDEA y las inserte en el sistema. 	8h
TP.05	API backend	A
	<ul style="list-style-type: none"> Endpoints básicos para enviar las series de datos al frontend. 	3h
	<ul style="list-style-type: none"> CORS. 	2h
TP.06	Visualización de las series en forma de tabla.	A
	<ul style="list-style-type: none"> Componente tabla de datos Svelte. 	5h

TP.07	Implementación de e-chart para la visualización de series.	A
	<ul style="list-style-type: none"> • Componente e-chart en Svelte. 	5h
	<ul style="list-style-type: none"> • Visualización de varias series en el mismo gráfico. 	5h
TP.08	Botón de descarga del gráfico.	M
	<ul style="list-style-type: none"> • Botón para descargar el gráfico que se esté visualizando en el momento. 	2h
TP.09	Mejorar apartado visual.	M
	<ul style="list-style-type: none"> • Añadir estilo al frontend 	6h
TP.10	Filtrado de series	A
	<ul style="list-style-type: none"> • Formulario de búsqueda que debe incluir los siguientes criterios: palabra clave, fuente, origen, tipo de datos y periodicidad. 	10h
	<ul style="list-style-type: none"> • Endpoint en la API que devuelva un listado de series filtrado según los criterios descritos anteriormente. 	3h
TP.11	Scraper MINECO	A
	<ul style="list-style-type: none"> • Scraper que recolecte series del Ministerio de Economía e inserte las series en el sistema. 	6h
	<ul style="list-style-type: none"> • Datos de las fuentes coherentes y mostrados correctamente. 	3h
Tercera fase: Finalización		
Fecha Inicio: 29/07/2020		Fecha fin: 29/08/2020
TF.01	Scraper INE, EUROSTAT.	B
	<ul style="list-style-type: none"> • Scraper para fuente: INE. 	6h
	<ul style="list-style-type: none"> • Scraper para fuente: EUROSTAT. 	6h
TF.02	Mejorar frontend – tareas extra frontend	B

	• Menú principal.	5h
	• Mejorar estilo, apartado visual del sistema.	5h
	• Control de errores.	3h
TF.03	Mejorar backend – tareas extra backend	M
	• Control de excepciones.	8h
	• Documentación Swagger.	3h
TF.04	Avisos del sistema	M
	• Aviso (pop up) al visualizar series con unidades o periodicidad distinta.	4h

Prioridades: A – Alta, M – Media, B – Baja

En la siguiente tabla se puede ver la comparación entre el tiempo real y la estimación para cada tarea y fase de desarrollo, incluyendo la desviación, la cual sería más útil para un desarrollo iterativo, ya que, al realizar varias iteraciones, se mejorarían las estimaciones en cada una aprendiendo de las anteriores. Más adelante se explican en detalle las metodologías junto a sus ventajas y desventajas.

Tarea / Fase	Estimación	Real	Diferencia	Desviación
TI.01	5	5	0	0 %
TI.02	4	6	+2	+50 %
Inicio	9	11	+2	+22,22 %
TP.01	3	3	0	0 %
TP.02	11	25	+14	+127,27 %
TP.03	8	4	-4	-50 %
TP.04	8	10	+2	+25 %
TP.05	5	3	-2	-40 %

TP.06	5	6	+1	+20 %
TP.07	10	17	+7	+70 %
TP.08	2	6	+4	+200 %
TP.09	6	5	-1	-16,67 %
TP.10	13	15	+2	+15,38 %
TP.11	9	13	+4	+44,44 %
Prototipo	80	107	+27	+33,75 %
TF.02	13	6	-7	-53,85 %
TF.03	11	6	-5	-45,45 %
TF.04	4	5	+1	+25 %
Finalización	28	17	-11	-39,29 %
TOTAL	117	135	+18	+15,38 %

Costes

El coste más importante y el único coste directo que presenta este proyecto es el salario, como este proyecto se está desarrollando en una asignatura, para el cálculo del sueldo se han usado las 300 horas determinadas por la asignatura. Para este cálculo, también se ha usado [Glassdoor](#) para determinar el sueldo anual de un programador Junior en Sevilla.

La fórmula usada para calcular el coste es:

$$\frac{S*H}{JA} \rightarrow \frac{16.000*300}{1.688} = 2.843,60 \text{ €}$$

S: Salario anual. H: Horas de la asignatura. JA: Jornada anual

Además del salario, se tiene en cuenta el coste de amortización del equipo usado durante el desarrollo. Se ha calculado aplicando un porcentaje de pérdida de valor anual del 20% al precio original sin impuestos del equipo, lo que supone la siguiente fórmula:

$$\frac{P*0.79*H}{5*JA} \rightarrow \frac{1.000*0.79*300}{5*1.688} = 28,08 \text{ €}$$

Por lo que el coste total del proyecto sería **2.871,68 €**.

4.4. Metodología

Existen varias metodologías que se pueden usar a la hora de desarrollar un proyecto software. Estas metodologías se suelen separar entre tradicionales y ágiles. Los modelos tradicionales, llamados también modelos en cascada o lineales, se caracterizan por ser secuenciales, es decir, el proyecto pasa por las siguientes fases de manera sistemática: análisis, diseño, desarrollo, pruebas, mantenimiento. Por el contrario, los modelos ágiles o iterativos se caracterizan por fragmentar los requisitos en distintos grupos y realizar una iteración para implementar cada uno de dichos grupos. Es decir, un desarrollo iterativo es similar a realizar varios desarrollos en cascada con menos carga de trabajo.

Para este proyecto, se ha elegido una metodología tradicional, ya que los requisitos se conocían desde el principio, ser un proyecto llevado a cabo por una persona y disponer de un tiempo limitado para el desarrollo, el modelo iterativo no resultaría eficiente. No obstante, sí que se han usado ciertas características del modelo iterativo, de hecho, el planning se divide en fases que, aunque no se puedan llamar iteraciones, sí que guardan cierta relación. En concreto, la fase de prototipo es usada para crear, como su nombre indica, un primer prototipo funcional que enseñar al cliente, por lo que cumple con las características de un modelo iterativo. Además, es importante resaltar que a lo largo de esta fase se obtiene un aprendizaje muy importante para la siguiente fase. Asimismo, durante la finalización del prototipo se llevó a cabo una reunión con el cliente con el fin de obtener feedback y realizar las modificaciones pertinentes de cara a la última fase, lo cual encaja perfectamente con las metodologías ágiles. No obstante, todas las fases del producto están orientadas al desarrollo del producto completo, por lo que, a pesar de que se tomen prestadas ciertas características ágiles, este proyecto está realizado con una metodología tradicional.

Por otro lado, el backend y el frontend van a ser desarrollados de manera individual, lo que se conoce como arquitectura desacoplada. Posiblemente, la ventaja principal de esta arquitectura es la fluidez que adquiere el sistema, ya que el intercambio de información entre backend y frontend es extremadamente ligero, en cada petición y respuesta entre el cliente y servidor solo se intercambian archivos *Json*. Además, el backend puede ser utilizado por otras interfaces web o aplicaciones, lo que añade valor al producto.

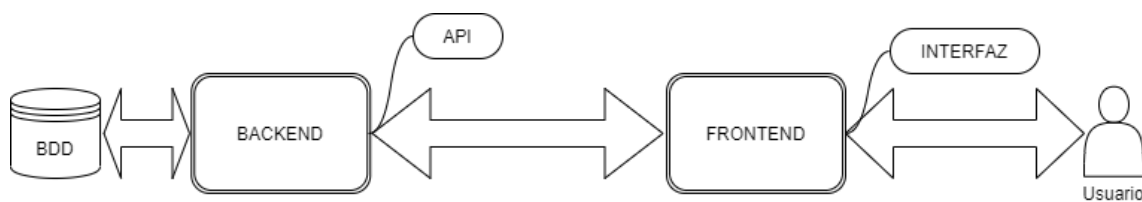


Ilustración 1: Arquitectura del sistema

4.5. Herramientas

4.5.1. GitHub

GitHub es una plataforma de desarrollo colaborativo en la que se pueden almacenar proyectos usando Git. En este proyecto, a pesar de que no sea colaborativo, al ser desarrollado por una persona, se va a usar GitHub para el control de versiones y la gestión de las tareas.

Además, al usar GitHub para almacenar el proyecto y para la gestión de tareas, se van a usar funcionalidades más avanzadas, como asociar cada subida de código o de archivos a una tarea concreta.

4.5.2. Entornos de desarrollo

Se van a usar dos entornos de desarrollo distintos: IntelliJ para el backend y Visual Studio Code para el frontend.

IntelliJ es un buen IDE que ha sido seleccionado entre otros, como por ejemplo Eclipse, debido a que ofrece muchas funcionalidades útiles para desarrollar en Spring Boot. En concreto, se puede conectar directamente con Spring Initializr, lo cual es muy

eficiente a la hora de crear proyectos nuevos. Además, es un IDE usado habitualmente por el desarrollador del proyecto, por lo que no hay alternativa mejor.

Se ha elegido Visual Studio Code para el frontend por ser usado habitualmente por el desarrollador del proyecto, al igual que IntelliJ, además de ser bastante popular para el desarrollo web.

4.5.3. Frameworks

Backend

Respecto al backend, no ha habido debate para elegir la tecnología a usar, Spring Boot es un framework de desarrollo de proyectos en Java muy conocido tanto por la comunidad de desarrolladores como por el desarrollador del proyecto. Al ser tan conocido, se minimiza el riesgo del desarrollo del backend, y permite que se comience a trabajar lo antes posible.

Frontend

Respecto al frontend, ha sido necesario realizar un estudio previo para elegir la tecnología adecuada: las candidatas eran Svelte, propuesta por la tutora de este proyecto, y React, un framework bastante reconocido por la comunidad y con el que el alumno ya ha adquirido cierta experiencia. Svelte posee ciertas características útiles para este proyecto, como puede ser la reactividad. Adicionalmente, el uso de esta herramienta implica cumplir algunos objetivos del proyecto al ser una tecnología nueva desconocida para el alumno.

React, por otro lado, ya es conocida por el alumno, esta tecnología es muy conocida y con una comunidad grande, lo que sin duda es una gran ventaja. Este es un framework aplicable a todo tipo de aplicaciones web, que se centra mucho en la creación y reutilización de componentes.

Finalmente, la tecnología decidida para el desarrollo ha sido Svelte debido a que es la tecnología que más se adecua a los objetivos del proyecto. Este es un framework de interfaces de usuario muy reciente que tiene varias características interesantes. No obstante, al ser tan novedoso, hay menos documentación disponible y la comunidad de desarrolladores es reducida.

MongoDB

Para la base de datos se va a usar un sistema NoSQL, es decir, en vez de guardar los datos en tablas, como se hace tradicionalmente, se guardan en un esquema dinámico sin relaciones. MongoDB es capaz de devolver los datos de manera más fácil y rápida que las bases de datos relacionales, por lo que, al no tener un sistema complejo de relaciones, es el sistema más adecuado para este proyecto.

Apache E-Charts

Apache ECharts es una librería muy potente para la visualización de gráficos en el navegador. Esta librería incluye gran multitud de tipos de gráficos, desde simples diagramas de barras hasta gráficos tridimensionales. Además, permite una alta personalización de los gráficos, dejando incluso crear gráficos propios. La siguiente imagen muestra un gráfico extraído de la página oficial con los componentes más usados de la librería.

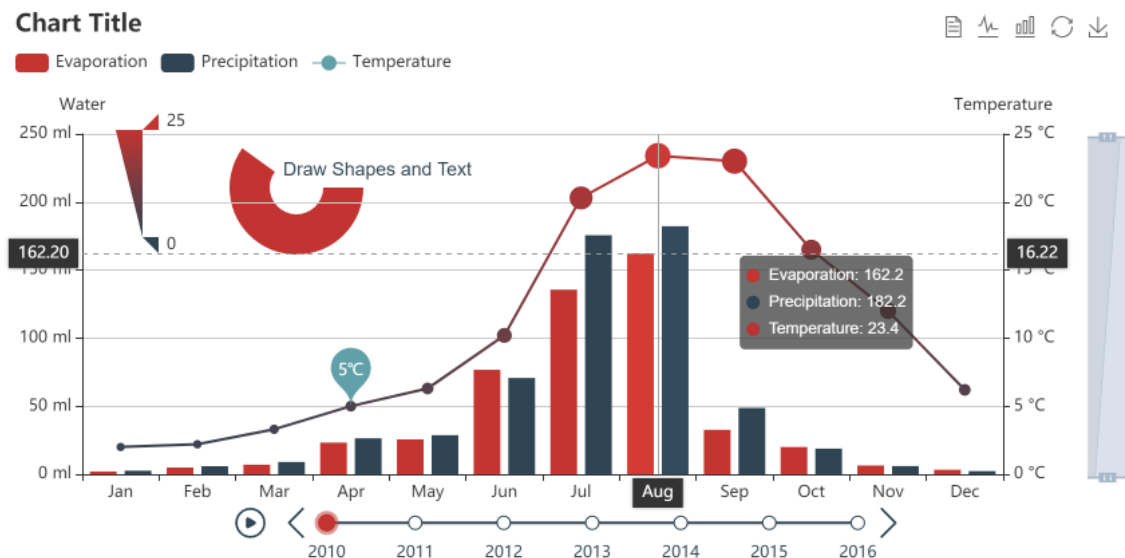


Ilustración 2: Apache ECharts.

Swagger

Swagger es una herramienta muy fácil de usar que permite documentar APIs entre otras cosas. La finalidad de usar esta herramienta, para este proyecto, es generar una documentación clara y completa de cómo usar la API que ha sido desarrollada. Al no existir ningún estándar de diseño ni de documentación de APIs, Swagger arregla este problema al crear, automáticamente, una descripción completa de la API documentada.

Además, Swagger ofrece una interfaz gráfica llamada Swagger UI para poder visualizar toda esta información, a continuación, se muestra una imagen de ejemplo:

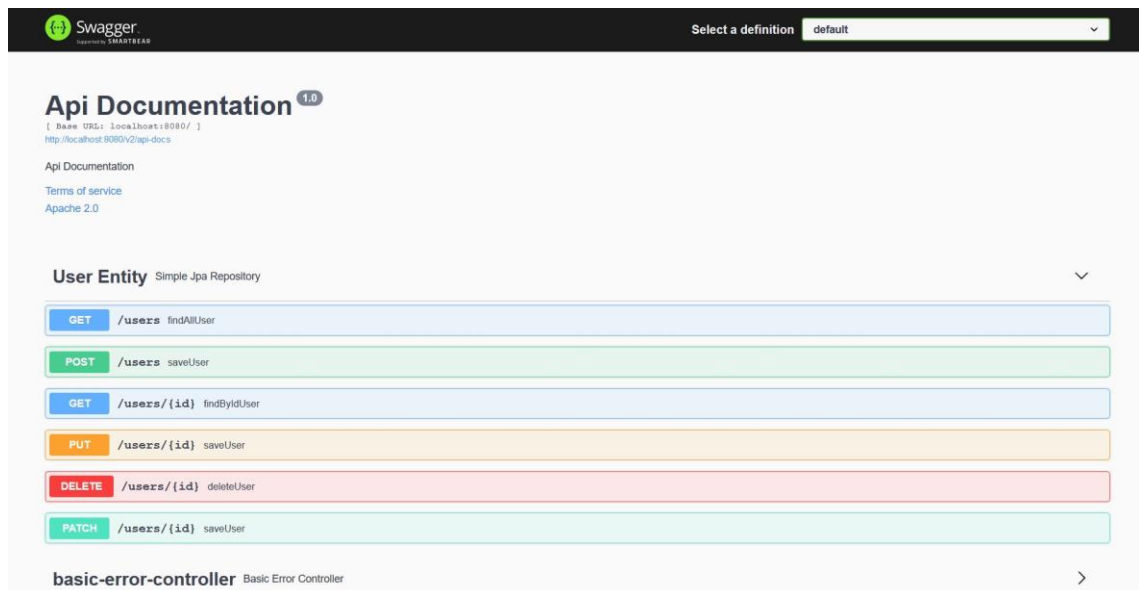


Ilustración 3: Swagger UI

4.6. Relación con las asignaturas del Grado

Diseño y pruebas

Diseño y pruebas son dos asignaturas (DP 1 y DP 2) de tercer curso en las cuales se forma al alumno para ser capaz de diseñar e implementar sistemas web a partir de unos requisitos establecidos por el cliente. Estas asignaturas, a pesar de ser muy intensas, son muy importantes para aprender conceptos fundamentales para un Ingeniero del Software que se dedique a sistemas web. A lo largo de esta asignatura no solo se aprende a diseñar, implementar, probar y desplegar aplicaciones web, sino que el alumno tendrá que pasar por todas las dificultades que supone realizar una aplicación web con muy poca experiencia.

Adicionalmente, el framework usado para esta asignatura es una versión antigua del mismo que se usa para el backend de este proyecto: Spring.

Ingeniería del Software y Práctica Profesional

Esta asignatura, al igual que Diseño y Pruebas, está fuertemente ligada a este proyecto. ISPP es una asignatura de último curso en el que los grupos de alumnos tienen que crear

un producto desde cero, esto implica que la idea del proyecto ha de ser propia de los alumnos.

Esta asignatura es más completa que la anterior, sin embargo, el alumnado tiene más experiencia por lo que se le puede exigir más. A pesar de no haberse utilizado en este proyecto ninguna de las tecnologías usadas en la asignatura, la arquitectura del sistema desarrollado también es desacoplada. Esto ha supuesto que ya se dominase el concepto de backend y frontend, y lo que conlleva desarrollar un producto de estas características.

Planificación y Gestión de Proyectos Informáticos

Esta asignatura se centra completamente en las fases de planificación y gestión de proyectos software, como su nombre indica, por lo que se complementa con la asignatura ISPP y DP. Esta asignatura, también realizada en grupo, tiene como objetivo proporcionar al alumno una visión clara de cómo se gestiona y dirige un proyecto software, sus contenidos trata sobre la organización, planificación y documentación de un proyecto teniendo en cuenta la normativa y legislación aplicables. Esta asignatura se destaca bastante ya que trata aspectos que no se tratan en otras asignaturas del grado.

Complementos de Bases de Datos

Esta es una asignatura optativa de cuarto curso cuyo objetivo es consolidar y ampliar los conocimientos sobre bases de datos adquiridos a lo largo del grado. A lo largo de esta asignatura, se estudian todos los tipos y variantes de bases de datos, en concreto, las bases de datos no relacionales se estudian por primera vez en CBD. De hecho, en esta asignatura se explica por qué las bases de datos NoSQL son mejores para tratar grandes cantidades de datos y tienen más escalabilidad horizontal, lo cual ha sido el factor determinante para elegir el tipo de base de datos para este proyecto.

Acceso Inteligente a la Información

Acceso Inteligente a la Información es otra asignatura optativa de último curso, en la que, además de aprender a implementar interfaces de usuario y a usar Django, se enseña a extraer información desde páginas HTML estáticas en Python.

A pesar de ya haber trabajado con Crawlers en el pasado, en esta asignatura se enseña a localizar y extraer los elementos deseados de una página HTML. Estos conocimientos son extremadamente útiles a la hora de construir Scrapers, por lo que esta asignatura también guarda relación con este proyecto.

5. Realización del proyecto

5.1. Análisis

Requisitos de información

RI.01 - Se quiere guardar la siguiente información acerca de cada serie:

- Nombre.
- Periodo (fecha de inicio y fin de datos disponibles).
- Fuente (IECA/INE/EUROSTAR).
- Periodicidad (Mensual/Trimestral/Anual).

En el caso de encontrar más información útil mediante los Scrapers, se podrán añadir campos adicionales.

RI.02 - Para cada serie, hay que guardar los datos correspondientes a cada momento en el tiempo, los cuales constan de:

- Año.
- Valor.
- Periodo.

Es necesario que se guarde la información del momento en el tiempo en dos campos distintos debido a que la periodicidad de los datos puede cambiar. Por tanto, si los datos fuesen trimestrales, el año puede no variar de un dato a otro, pero el conjunto de año y periodo debe ser único para cada dato de la serie.

Requisitos funcionales

RF.01 – El backend deberá poder conectarse con la base de datos para almacenar, modificar o eliminar los datos.

Para la inserción de datos desde los scrapers y la recuperación de estos para mostrarlos en el frontend, es necesario que el backend sea capaz de acceder a la base de datos y realizar las operaciones necesarias.

RF.02 – El backend deberá disponer de scrapers para las distintas fuentes especificadas para poder extraer los datos e insertarlos en la base de datos.

Han de implementarse distintos scrapers que extraigan las series de datos de diversas fuentes, para lo que es necesario que estos programas estén contenidos en el backend y así obtener acceso a la base de datos e insertar la información recolectada.

RF.03 – El backend deberá de disponer de una API con la que poder comunicarse con el frontend.

Para que el frontend funcione correctamente, es necesario que exista un canal de comunicación con el backend, desde el que poder recuperar las series de datos entre otras cosas. Para ello, ha de definirse una API que de soporte al frontend, permitiéndole realizar las llamadas necesarias para funcionar adecuadamente.

RF.04 – El sistema permitirá al usuario listar las series almacenadas.

Dicha lista ha de contener la descripción de la serie, su código, territorio, unidad y periodicidad. Además, para evitar una lista extremadamente grande, ha de estar paginada.

RF.05 – El sistema permitirá al usuario filtrar la lista de series.

Al tener una gran cantidad de series almacenadas en el sistema, es necesario que el usuario sea capaz de filtrarlas por palabra clave, territorio, unidad o periodicidad para poder visualizar únicamente las series en las que esté interesado.

RF.06 – El sistema permitirá al usuario realizar una selección de series.

Esto es necesario para que el usuario pueda elegir que series mostrar en el gráfico o en la tabla.

RF.07 – El sistema permitirá al usuario mostrar una o varias series en un gráfico.

Las series seleccionadas por el usuario deberán poder visualizarse en un diagrama de líneas mediante un botón o similar. Adicionalmente, si el usuario ha seleccionado varias

series para compararlas, el sistema mostrará un aviso de que las series no son compatibles en el caso de que tengan unidades o periodicidades distintas.

RF.08 – El sistema permitirá al usuario mostrar una o varias series en una tabla.

Las series seleccionadas por el usuario deberán poder visualizarse en una tabla mediante un botón o similar. Adicionalmente, si el usuario ha seleccionado varias series para compararlas, el sistema mostrará un aviso de que las series no son compatibles en el caso de que tengan unidades o periodicidades distintas.

RF.09 – El sistema permitirá al usuario descargar la gráfica que esté visualizando.

Cuando el usuario se encuentre en la pantalla de visualización del diagrama, debe aparecer un botón de descarga que exporte una imagen con el gráfico.

RF.10 – El sistema permitirá al usuario descargar la tabla que este visualizando.

Cuando el usuario se encuentre en la pantalla de visualización de la tabla, debe aparecer un botón de descarga que exporte un archivo en formato csv de la tabla.

5.2. Backend

5.2.1. Diseño

A continuación, se muestra el modelo de dominio, el cual contiene las entidades que han de ser almacenadas por el sistema:

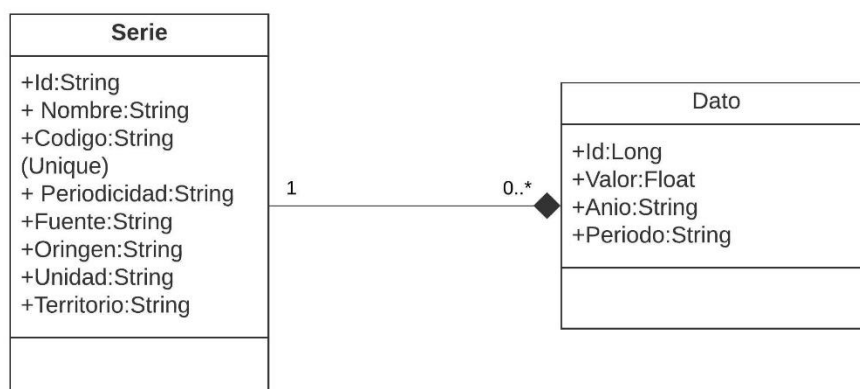


Ilustración 4: modelo conceptual

Como podemos observar, la relación entre *Serie* y *Dato* no es una relación estándar, sino que es una composición. Esta es una relación especial, indica que las series están compuestas de datos, es decir, no tiene cabida una serie sin datos ni un conjunto de datos sin serie.

5.2.2. Implementación

Scraper IECA

El primer scraper realizado tiene como objetivo el Instituto de Estadística y Cartografía de Andalucía (IECA). Este es un organismo perteneciente a la Junta de Andalucía en cuya web se recopilan series estadísticas a lo largo del tiempo sobre distintos aspectos de la sociedad. Las principales series consultadas según la propia web son: índice de población, índice del producto interior bruto (PIB), índice de precios al consumo (IPC), número de turistas, sociedades mercantiles constituidas y tasa de paro.

El primer acercamiento para crear el Scraper fue intentar crear un Crawler que recorriese las páginas de las series de datos extrayendo los datos del propio código fuente, pero esto resultó imposible ya que la página está construida con muchas funciones JavaScript y los datos no aparecen en el código fuente.

Afortunadamente, la página permite guardar una cesta con mucha información sobre las series seleccionadas. Finalmente, el Scraper funciona recorriendo esta cesta y realizando peticiones a la web con los datos de la propia cesta. A continuación, se expone un ejemplo básico:

En la información de la cesta de la Serie X se indica que su código es: 1234, por lo que para acceder a sus datos se construye la siguiente URL:

`www.ieca.es/obtenerSerie?cod=1234`

Así, tras recorrer la cesta en la que se incluyen todas las series disponibles, tenemos los datos de todas las series de esta fuente.

Scraper MINECO

El segundo scraper implementado recopila las series de datos almacenados en la web del Ministerio de Economía de España, el cual almacena series del mismo tipo que IECA, pero aplicadas a toda España.

Al igual que en la web de la Junta de Andalucía, a pesar de intentar recorrer las páginas de datos con un Crawler, estas páginas no existen realmente, sino que son mostradas a través funciones JavaScript, lo que hace imposible recorrerlas de manera automática.

La solución encontrada finalmente es parecida a la de IECA: se descargan los datos completos en un directorio zip para, posteriormente, recorrer esta carpeta y se extraer todos los datos de cada serie.

Se han encontrado distintas dificultades respecto a la fuente anterior:

La primera de ellas viene dada por el formato de los datos, pues al contrario que IECA (que permite la descarga en distintos formatos conocidos), el Ministerio de Economía tiene su propio formato y solo permite la descarga de todas las series en ese formato, lo que ha provocado un esfuerzo adicional a la hora de extraer los datos y estructurarlos a nuestra manera. Por otra parte, la cantidad de series es muy elevada (22.000 aproximadamente), lo que provoca una gran caída del rendimiento al extraer todos los ficheros de datos. Este último problema se solventó adaptando el algoritmo del scraper para que extraiga la información de los ficheros individualmente.

MongoDB

La implementación de la base de datos es importante a pesar de su sencillez. El primer paso es incluir las dependencias de MongoDB en el POM de la aplicación. Gracias a Spring Initializr, este paso se realizó automáticamente en la creación de la aplicación base.

El siguiente paso es adaptar las clases del dominio a MongoDB, lo que también es sencillo, pues únicamente tendremos que usar la anotación `@Document(collection = "examples")` para especificar un nombre para la colección. Sin embargo, este paso se puede omitir, lo que provocará que el nombre de la colección sea el mismo del de la clase en cuestión. Además, queremos que MongoDB guarde un identificador único para cada

objeto, por lo que hay que especificar ese atributo en la clase y anotarlo con `@Id`. MongoDB se encargará de gestionar el valor para este atributo internamente.

Otra característica de MongoDB en Spring Boot que hay que tener en cuenta es que no hace falta la existencia previa de la base de datos, pues la misma aplicación la generará en el caso de que no la encuentre. Por defecto, el nombre de la base de datos autogenerada es `"test"`, pero se puede modificar especificando un nombre en las propiedades de la aplicación mediante el siguiente parámetro:

```
spring.data.mongodb.database = databasename
```

Sin embargo, este no es el único parámetro que se ha tenido que incluir para que los índices especificados en el dominio se generen en la base de datos hay que añadir este otro parámetro:

```
spring.data.mongodb.auto-index-creation = true
```

Esta opción permite añadir índices de MongoDB desde el dominio java y, en concreto, se ha usado para añadir la restricción de unicidad a los códigos de las series. Esta restricción va a asegurar que no haya series duplicadas en el sistema, pues si se intentase insertar una serie con un mismo código, MongoDB lanzará una excepción tipo `"DuplicateKeyException"`. A continuación, se muestra el resultado de dicha excepción controlada mediante el manejo de excepciones explicado más adelante:

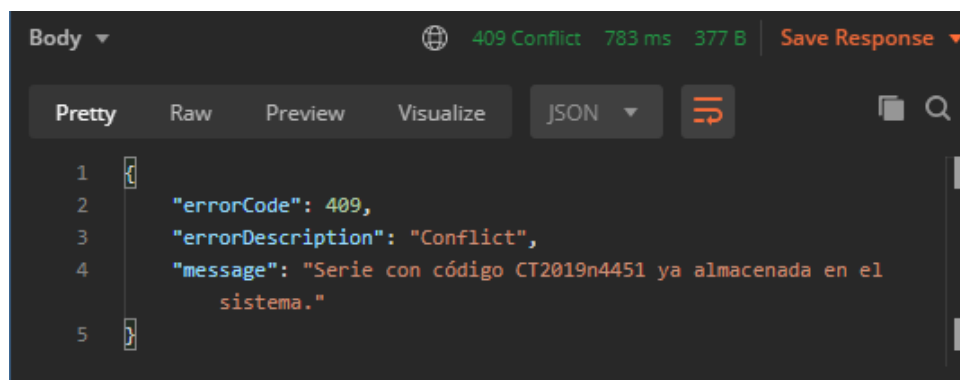


Ilustración 5: Excepción por código duplicado.

API

Al tener una aplicación con arquitectura desacoplada, es necesario que el frontend y el backend se comuniquen mediante algún mecanismo. Para conseguir este objetivo, el backend se ha realizado en formato API, es decir, desde el punto de vista del frontend, va a haber una serie de endpoints disponibles con los que obtener o enviar datos al backend. Esto es muy fácil de realizar en Spring Boot, ya que únicamente hace falta poner las siguientes anotaciones a los controladores que vayan a formar parte de la API:

@RestController: Identifica al controlador como parte de la API.

@RequestMapping: Indica la ruta hacia el controlador. p.e: (*@RequestMapping("/serie")*).

@GetMapping: Se colocan sobre los métodos del controlador para especificar el tipo de solicitud HTTP del endpoint y especificar la ruta si es necesario. Existen distintos tipos de solicitudes, los más usados son *Get*, *Post*, *Put* y *Delete*. p.e: (*@GetMapping("/{id}")*).

Una vez configurada la aplicación con estas anotaciones ya está lista para interactuar con el frontend, pero para que las dos aplicaciones se puedan comunicar es necesario habilitar el CORS desde el backend hacia el frontend, ya que si no las peticiones serán bloqueadas por este mecanismo.

Hay distintas alternativas para habilitar el CORS, la más simple es especificar mediante la anotación *@CrossOrigin*, la cual se puede colocar en los métodos del controlador o en todo el controlador para aplicar los ajustes a todos los métodos a la vez. No obstante, se ha optado por aplicar una configuración global para toda la aplicación. Esto se ha conseguido modificando la clase de configuración de la aplicación, añadiéndole la siguiente línea de código:

```
registry.addMapping("/serie/**").allowedOrigins("http://localhost:5000");
```

Esto permite el acceso desde localhost:5000 al recurso serie y todos sus endpoints.

Control de excepciones

Para que los errores que ocurran durante la ejecución de la aplicación estén controlados y los mensajes de error sean comprensibles, se ha hecho uso del manejo de excepciones que ofrece Spring Boot, además de crear excepciones propias de la aplicación y mensajes de error personalizados.

Sin embargo, hay que especificar qué excepciones queremos controlar a Spring, lo cual se puede hacer a nivel de controlador, pero como ocurre también con el CORS, es más interesante solucionar este problema a nivel de aplicación, lo que se consigue especificando todas las excepciones en una clase aparte marcada con la anotación `@ControllerAdvice`.

En la clase de configuración hay que crear un método para cada excepción que se quiera tratar y marcar el método con la siguiente anotación:

`@ExceptionHandler({ ExampleException.class})`

Esto permite capturar la excepción y tratarla de forma individual, así se puede especificar el código de error HTTP correspondiente a la excepción y crear una respuesta más explicativa.

A continuación, se pueden observar las diferencias entre controlar una excepción o no; en concreto, este caso es el error provocado por intentar obtener una serie inexistente:

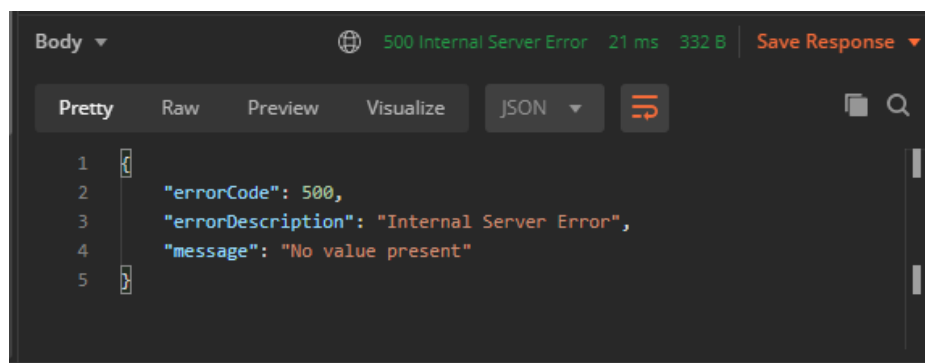
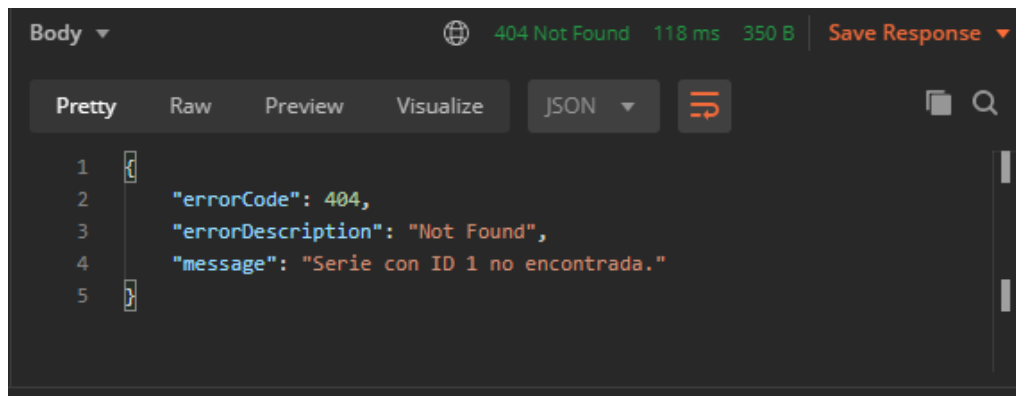


Ilustración 6: Excepción no controlada

Como se puede observar en la imagen, el error es poco descriptivo y el estatus es “500 Internal Server Error”, cuando debería ser “404 Not Found” debido a que no es un error del sistema, si no que no se ha encontrado el recurso solicitado.

A continuación, se muestra el comportamiento de la aplicación con el manejo de excepciones implementado, y se puede observar que el mensaje es mucho más explicativo y que el status mostrado es el correcto.



```
Body ▾ 404 Not Found 118 ms 350 B Save Response ▾  
Pretty Raw Preview Visualize JSON ↺  
1 {  
2   "errorCode": 404,  
3   "errorDescription": "Not Found",  
4   "message": "Serie con ID 1 no encontrada."  
5 }
```

Ilustración 7: Excepción controlada

Swagger

Swagger da uso de las anotaciones vistas anteriormente, como `@RestController`, para identificar los controladores y métodos que van a ser documentados.

Para configurar Swagger en Spring Boot, únicamente hay que crear un método *Docket* en la clase de configuración. En este punto, podemos definir algunos parámetros para controlar el comportamiento de Swagger, pero en este proyecto se ha dejado la configuración básica ya que se quiere documentar toda la API.

Una vez está configurado, Swagger nos permitirá usar comentarios en el código para documentar los métodos de la API y su funcionamiento, es decir, a través de esta herramienta, las personas que vayan a dar uso de la API, serán capaces de visualizar un listado de todos los endpoints que ofrece (sus definiciones, restricciones y especificaciones), adquiriendo así una idea bastante buena de cómo usar la API. Esto es bastante útil ya que normalmente se da el caso de querer usar APIs de terceros, pero su documentación puede estar mal hecha o incompleta.

Calidad del código

Para controlar la calidad del código se han usado dos plugins para IntelliJ: CheckStyle y SonarLint.

CheckStyle es una herramienta que analiza el código y avisa al desarrollador si este no cumple con las reglas de codificación establecidas. Esta herramienta es muy útil ya que es común dejar pasar pequeñas faltas en el código que, al compilar la aplicación, son

imperceptibles a la hora de ejecutar la aplicación. Además, esta herramienta ayuda a mantener el código limpio y legible.

SonarLint es una herramienta que cumple una función similar a CheckStyle, sin embargo, esta herramienta se centra más en los bugs y problemas de seguridad que puedan aparecer.

Al ser plugins de IntelliJ, ambas herramientas pueden analizar el código mientras es escrito, lo cual es una gran ventaja para el desarrollador.

5.2.3. Pruebas

Para asegurar el correcto funcionamiento de los servicios y controladores se han realizado tests unitarios para cada método que aparece en dichas clases para comprobar que funcionan según se espera.

5.3. Frontend

5.3.1. Diseño

A continuación, se muestran los mockups de las pantallas más importantes de la aplicación. Estos representan una imagen aproximada que puede ser usada como guía de la información y estructura que tendrá el frontend una vez se desarrolle:

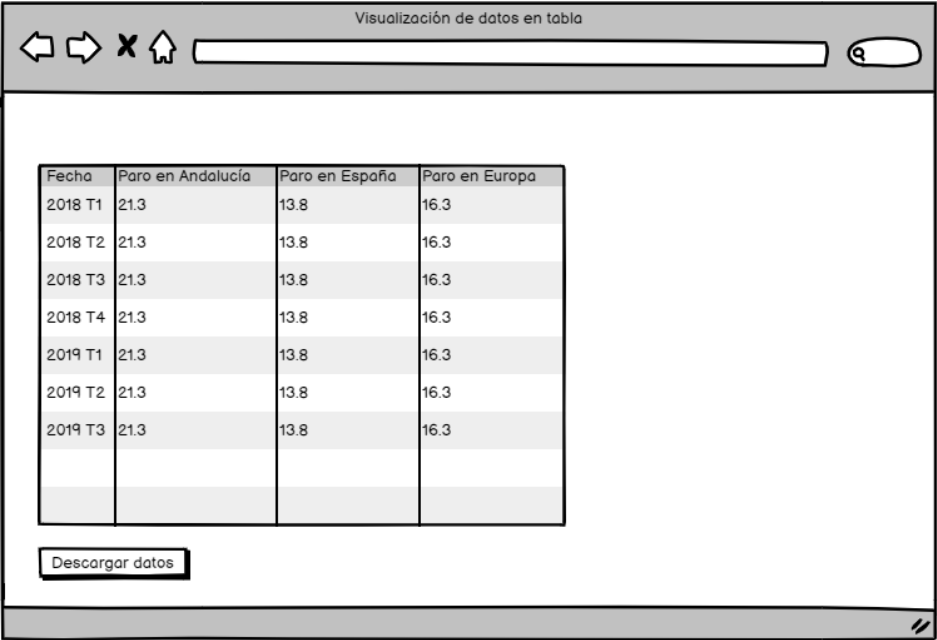
Mockup of the 'Listado de series' screen. The interface includes a header with navigation icons (back, forward, close, home) and a search bar. Below the header is a filter section with dropdown menus for 'Fuente', 'Periodicidad', and 'Territorio', and a 'Buscar' button. A table displays the following data:

Nombre	Fuente	Periodicidad	Territorio	Unidad	Nº de datos
Paro en andalucía	IECA	Trimestral	Andalucía	Porcentaje	124
Paro en españa	MINECO	Trimestral	España	Porcentaje	124
Paro en europa	EUROSTAT	Trimestral	Europa	Porcentaje	124

At the bottom of the screen are two buttons: 'Ver tabla' and 'Ver grafico'.

Ilustración 8: Mockup listado de series.

Esta es una de las pantallas principales del sistema, en la cual los usuarios deberán poder visualizar las series almacenadas en el sistema. Además, los usuarios deben ser capaces de realizar una selección de series para visualizarlas en una tabla o un gráfico. Las pantallas siguientes son correspondientes a las visualizaciones.



The mockup shows a web browser window titled "Visualización de datos en tabla". It features a navigation bar with back, forward, close, and home icons, a search bar, and a refresh icon. The main content area displays a table with the following data:

Fecha	Paro en Andalucía	Paro en España	Paro en Europa
2018 T1	21.3	13.8	16.3
2018 T2	21.3	13.8	16.3
2018 T3	21.3	13.8	16.3
2018 T4	21.3	13.8	16.3
2019 T1	21.3	13.8	16.3
2019 T2	21.3	13.8	16.3
2019 T3	21.3	13.8	16.3

Below the table is a button labeled "Descargar datos".

Ilustración 10:Mockup tabla de datos

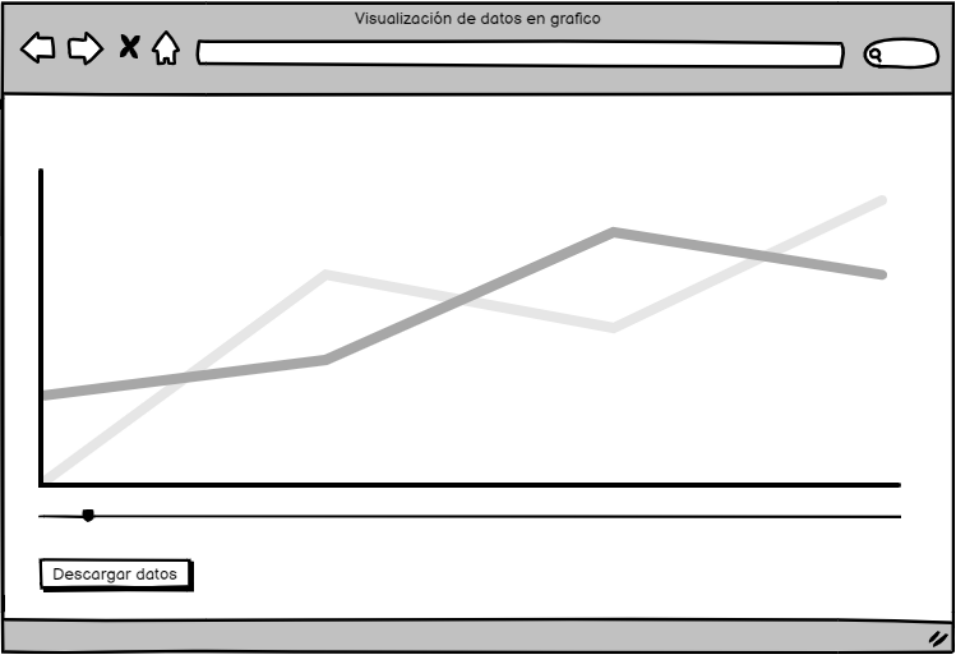


Ilustración 9:Mockup diagrama de líneas

5.3.2. Implementación

Svelte

Svelte, como se explica anteriormente, es un framework nuevo sin mucha documentación aún en internet más allá de la oficial. Esto ha supuesto, junto con la falta de experiencia en frontend del desarrollador, que se inviertan muchas horas en investigación. Sin embargo, una vez se aprenden las nociones básicas de Svelte, este resulta ser muy potente, ya que se pueden implementar tareas con menos líneas de código de las que harían falta con otro framework como React. Además, Svelte permite añadir estilo a los componentes desde el mismo fichero sin ser necesario crearlo aparte, lo cual es muy útil si se quiere aplicar un estilo concreto solo a los componentes de ese mismo fichero, ya que el resto de la aplicación quedará fuera del alcance.

Por otro lado, Svelte permite realizar declaraciones reactivas. Estas declaraciones provocan una actualización del DOM al cambiar su valor, lo que simplifica mucho el desarrollo de componentes que se actualizan a menudo ya que únicamente habría que controlar los elementos reactivos de manera adecuada. Por otro lado, además de variables, también se pueden declarar sentencias como puede ser un *console.log* o un bloque *if*.

Pero la reactividad de Svelte no acaba ahí, aunque este tipo de declaraciones sean útiles no han sido necesarias, ya que la reactividad de Svelte también es puesta en movimiento a través de las asociaciones. Esto quiere decir que, al asociar un valor a una variable también provocará que se recargue la página, visualizando así los cambios pertinentes. Sin embargo, al ser disparada por las asociaciones, el hecho de añadir un elemento a un *array* o realizar cualquier cambio sobre un objeto que no sea una asignación, la reactividad no se activará. No obstante, hay una alternativa muy simple a este problema, si, por ejemplo, queremos añadir un elemento a un *array* y que el sistema se comporte como si fuese una asociación, podemos asociar el *array* en cuestión a sí mismo, lo que no provocará ningún cambio en los datos, pero solucionará el problema.

No cabe duda de que con el tiempo Svelte se convertirá en un gran framework de interfaces de usuario, pero aún es demasiado joven como para dale uso en el ámbito laboral. Esto se debe a dos razones principalmente: la primera, es la falta de documentación (la cual puede ser muy limitante en un proyecto importante) y la segunda es la falta de componentes y herramientas de soporte existentes (lo que supone un gran

aumento en la carga de trabajo al tener que implementar todos los componentes desde cero). Aún en contraposición a las ventajas que ofrece Svelte, estos dos factores resultan determinantes, por lo que a la hora de realizar un proyecto a nivel empresarial, elegiría otro lenguaje como React o Angular. Pese a ello, sigue siendo un gran framework que recomendaría usar a cualquier desarrollador de frontend, Svelte proporciona otro punto de vista más novedoso frente a los frameworks usados actualmente

Componentes externos

Para mantener un estilo de componentes coherente y de calidad, se ha optado por utilizar los componentes de Svelte Material UI. Este último es una colección de componentes ya hechos que tiene diferentes versiones para utilizarse en los distintos frameworks (React, Angular, Svelte). Utilizar componentes ya existentes supone un ahorro de tiempo considerable teniendo en cuenta la escasa experiencia desarrollando componentes en frontend y en svelte, aunque al ser svelte un framework tan joven, estos componentes apenas han llegado a su primera versión estable, son bastante limitados y apenas están documentados.

El uso de estos componentes es bastante simple pues únicamente hay que instalarlo en la aplicación mediante `npm install`, importarlos en el código y ya podrían ser empleados como cualquier otro componente.

Estos componentes se han utilizado para los botones, tablas y selectores de la página, la siguiente imagen muestra la tabla de series, la cual usa todos estos componentes.

Listado de series

Palabra clave Origen Fuente Periodicidad Territorio

Código	Descripción	Fuente	Nº de Datos	Periodicidad	Datos	Territorio	Selección
CT2019n4451	pib a precios de mercado. índices de volumen encadenados	Instituto de Estadística y Cartografía de Andalucía (IECA)	98	Trimestral	Tasa interanual (porcentaje)	Andalucía	<input type="checkbox"/>
IDBn3860	esperanza de vida al nacer. mujeres	Instituto de Estadística y Cartografía de Andalucía (IECA)	24	Anual	BASE (años)	Andalucía	<input type="checkbox"/>
CT2019n4455	vab ramas industriales. índices de volumen encadenados	Instituto de Estadística y Cartografía de Andalucía (IECA)	98	Trimestral	Tasa interanual (porcentaje)	Andalucía	<input type="checkbox"/>
CT2019n4453	vab ramas primarias. índices de volumen encadenados	Instituto de Estadística y Cartografía de Andalucía (IECA)	98	Trimestral	Tasa interanual (porcentaje)	Andalucía	<input type="checkbox"/>
CT2019n4457	vab construcción. índices de volumen encadenados	Instituto de Estadística y Cartografía de Andalucía (IECA)	98	Trimestral	Tasa interanual (porcentaje)	Andalucía	<input type="checkbox"/>

Numero de series
5 « < ANTERIOR 0 SIGUIENTE > »

Ilustración 11: Listado de series con componentes

E-Charts

Para la visualización de los datos en gráficos, se tuvieron en cuenta varias librerías. Finalmente, se optó por Apache E-Charts, una librería muy potente al ofrecer una gran cantidad de personalización de los gráficos.

Aunque el gráfico que se muestra en la aplicación es relativamente sencillo porque solo han de mostrarse los datos de las distintas series en un diagrama de líneas, pero su implementación se ha llevado a cabo con varias dificultades. Para usar esta librería es necesario importarla como cualquier otro componente y, una vez instalada, para mostrar los gráficos, es necesario utilizar un elemento SVG o Canvas como contenedor.

Llegados a este punto, debemos inicializar el echart y especificar sus opciones. La inicialización es sencilla, solamente hace falta una línea de código usando la importación de echarts y usando el método *init*. No obstante, la especificación de opciones es más compleja pues se realiza con el método *setOptions*, el cual recibe como parámetro un objeto JavaScript con todas las especificaciones del gráfico, incluyendo los datos. A continuación, se explican todas las propiedades del objeto que se han usado:

- **Legend:** Atributo que habilita el componente de la leyenda del diagrama. La leyenda consta del título de cada serie mostrada y su indicador de color.
- **DataZoom:** Atributo que habilita el componente que permite hacer zoom al gráfico. Esta característica es especialmente útil ya que se trabaja con series compuestas de multitud de datos.
- **Toolbox:** Atributo que habilita la barra de herramientas del diagrama. Hay varias herramientas disponibles, pero en este caso únicamente se ha habilitado la de descarga del gráfico.
- **Tooltip:** Atributo que habilita el tooltip. Esto es una descripción emergente en la localización del ratón, en este caso se usa para mostrar los datos de las series en el momento correspondiente a la localización del ratón.
- **xAxis:** Atributo en el que se definen las características de eje x del gráfico. Según la implementación realizada, se especifica que los datos de este eje son de tipo *categoría* y se introducen los datos correspondientes, es decir, el listado de momentos que componen las series de datos mostradas.

- **yAxis:** Atributo en el que se especifica el tipo de datos que se va a mostrar en el diagrama. Al ser valores numéricos, se utiliza la opción *value*. Al contrario que en el eje x, en este atributo no se introducen datos.
- **Series:** Este es el atributo más importante pues en él se introducen todas las opciones de las series y sus datos correspondientes.

Para conseguir almacenar todos los datos de todas las series seleccionadas por el usuario, se realiza una llamada a la API con los identificadores de las series para recuperarlas de la base de datos. Una vez se obtienen las series, se itera sobre ellas para obtener los distintos periodos que han de representarse en el diagrama, los títulos de las series y, por supuesto, los valores numéricos. Por último, se insertan todos estos datos en el atributo *Series* de la siguiente manera:

```
series.push({  
  type: 'line',  
  name: serie.descripcion,  
  data: xData,  
  symbolSize: 6,  
  connectNulls: true  
});
```

Donde “*xData*” se corresponde al conjunto de datos de la serie, “*serie.descripcion*” al título de la serie y “*connectNulls*” es un atributo con el que se especifica que la línea del gráfico ha de conectarse en el caso de que haya valores nulos. Esto es necesario en el caso de que se comparen series de distintos periodos ya que, si comparamos una serie anual y una trimestral, los índices temporales correspondientes a los trimestres estarán asociados a valores nulos en la serie anual y viceversa. Esta opción permite que, a pesar de ser de poca utilidad, se puedan comparar visualmente series de distintos periodos.

Se puede encontrar un listado completo con todas las opciones y sus descripciones completas en la página oficial de [Apache Echarts](#).

Como se puede observar, a pesar de tener un grado de personalización elevado, ECharts no es una librería difícil de usar. Para el requisito de la descarga de imágenes, esta librería da opción de descargar la imagen tal cual se solicita, lo único que hay que hacer es habilitar la característica como se indica en el listado anterior:

```

toolbox: {
  feature: {
    saveAsImage: {
      show: true
    }
    ...
  }
}

```

A continuación, se muestra una imagen resultante de la exportación de un diagrama real de la aplicación usando la funcionalidad *saveAsImage* y otra usando *DataZoom*:

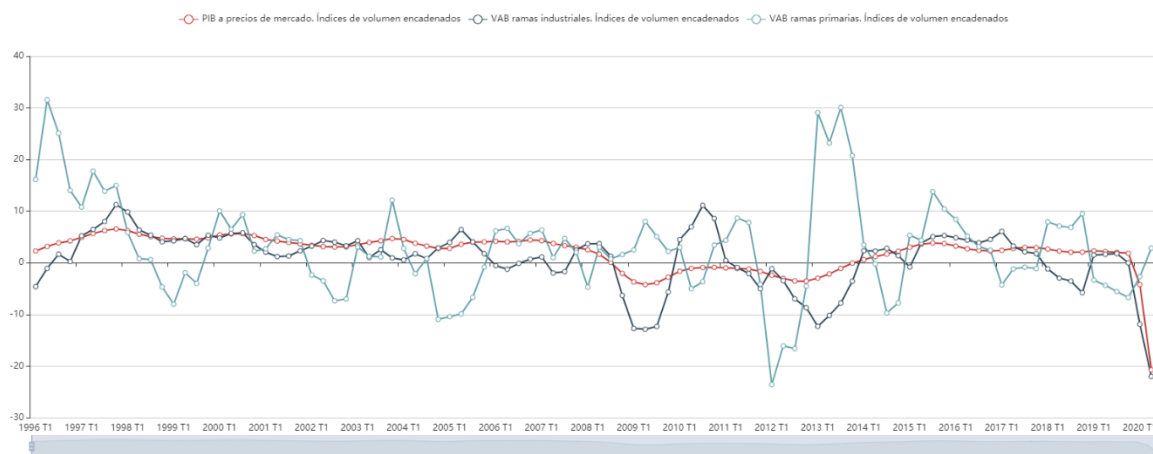


Ilustración 12: Diagrama exportado.

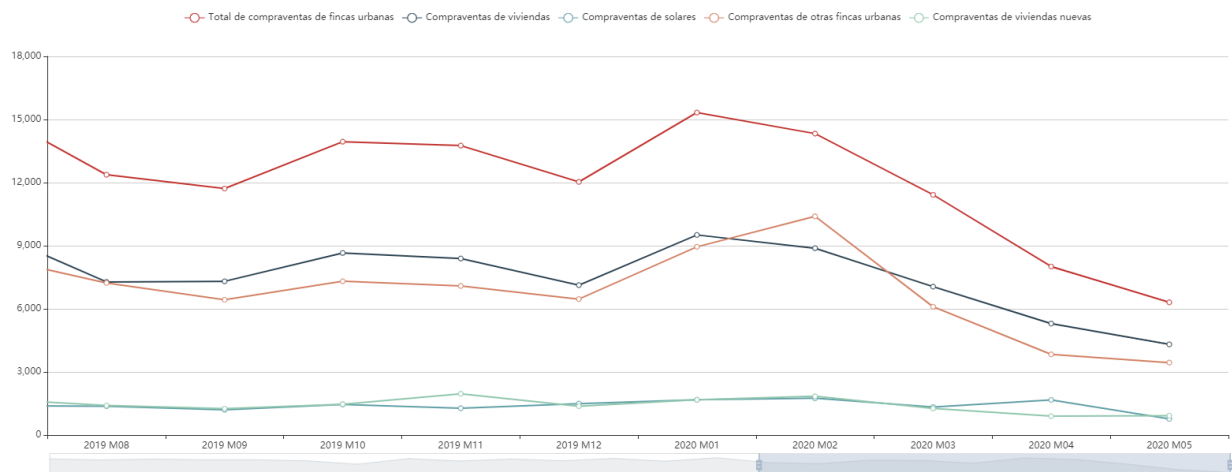


Ilustración 13: Diagrama con zoom.

En esta última imagen podemos observar cómo se amplían los datos, las series mostradas se corresponden a compraventas de distintos tipos de fincas, de hecho, se puede

observar claramente como ha disminuido la compraventa de fincas en los primeros meses de 2020, probablemente debido a la pandemia del COVID-19.

Problemas encontrados

El principal problema encontrado a lo largo del desarrollo del frontend ha sido la falta de documentación, lo cual ha supuesto una carga de trabajo de investigación superior a la esperada. Esta falta de documentación no solo se aplica a los conceptos de Svelte, sino a los distintos componentes usados.

Por otro lado, durante la implementación de la visualización de series con e-charts también se han encontrado problemas. En concreto, a pesar de que todas las demás características funcionasen correctamente, el tooltip no se mostraba con la configuración explicada en el apartado anterior. Para solucionar este problema se intentaron varias alternativas, la primera fue utilizar un componente que sirve de envoltorio para e-charts, este componente genera un Canvas con el componente echart listo para pasarle las opciones. Sin embargo, esto no solucionó el problema ya que es un problema de Svelte en sí mismo, pues a partir de la versión 3.15 de Svelte, el tooltip deja de ser visible. No se ha encontrado el motivo por el cual no se muestra en las versiones más recientes, por lo que la solución a este problema ha sido cambiar a dicha versión de Svelte.

Por último, se han encontrado algunos bugs visuales que han tenido que ser corregidos con CSS. El más importante era producido en los filtros de búsqueda de las series, provocando que los selectores se alineasen de manera incorrecta al seleccionar un valor.



Ilustración 14: Bug visual filtros.

5.3.3. Pruebas

Debido a la implementación del frontend, al no tener una navegabilidad mediante URLs, no se han realizado test.

6. Manual de usuario

Para mostrar el uso de la interfaz de la aplicación desarrollada, se van a desglosar las pantallas que la componen, indicando a qué se corresponde cada parte y qué funcionalidades ofrecen.

Listado de series

Listado de series **1**

Palabra clave Origen Fuente INE Periodicidad Territorio Andalucía **2**

Código	Descripción	Fuente	Nº de Datos	Periodicidad	Datos	Territorio	Selección
ETDPn3345	total de compraventas de fincas urbanas	Instituto Nacional de Estadística (INE)	24	Mensual	BASE (transmisiones)	Andalucía	<input type="checkbox"/>
ETDPn3346	compraventas de viviendas	Instituto Nacional de Estadística (INE)	24	Mensual	BASE (transmisiones)	Andalucía	<input type="checkbox"/>
ETDPn7485	compraventas de solares	Instituto Nacional de Estadística (INE)	24	Mensual	BASE (transmisiones)	Andalucía	<input type="checkbox"/> 4
ETDPn7486	compraventas de otras fincas urbanas	Instituto Nacional de Estadística (INE)	24	Mensual	BASE (transmisiones)	Andalucía	<input checked="" type="checkbox"/>
ETDPn3349	compraventas de viviendas nuevas	Instituto Nacional de Estadística (INE)	24	Mensual	BASE (transmisiones)	Andalucía	<input type="checkbox"/>

Numero de series 5 0 **5**

6

ETDPn7486 **7**

Ilustración 15: Pantalla listado de series

Esta es la pantalla principal del sistema, en la que se encuentra el listado de series. A continuación, se describen los elementos marcados por los números en la imagen:

- 1) Título: Título de la pantalla.
- 2) Filtros: Conjunto de campos donde el usuario podrá introducir los criterios para el filtrado de series: la palabra clave que será buscada en la descripción de la serie, la fuente, el origen, la periodicidad y el territorio.
- 3) Tabla: Tabla que muestra los distintos atributos que identifican a las series: el código, descripción, fuente, periodicidad, número de datos, tipo de datos y territorio.
- 4) Selección: Columna de la tabla destinada a la selección de las series, es decir, marcando el checkbox los usuarios seleccionarán la serie para poder mostrarla en tabla o gráfica (6).
- 5) Paginación: Conjunto de botones que conforman la paginación de la tabla, permitiendo al usuario navegar a las paginas siguiente, anterior, primera y

última. Adicionalmente, se pueden modificar el número de series que muestra la tabla (por defecto 5).

- 6) Botones de visualización: Conjunto de botones para mostrar las series en una tabla o en un diagrama de líneas.

En esta página el usuario podrá navegar por las series, filtrando según los criterios que desee. Adicionalmente, seleccionando las series que considere, el usuario será capaz de acceder a las demás pantallas pulsando los botones de visualización.

Visualización: diagrama de líneas

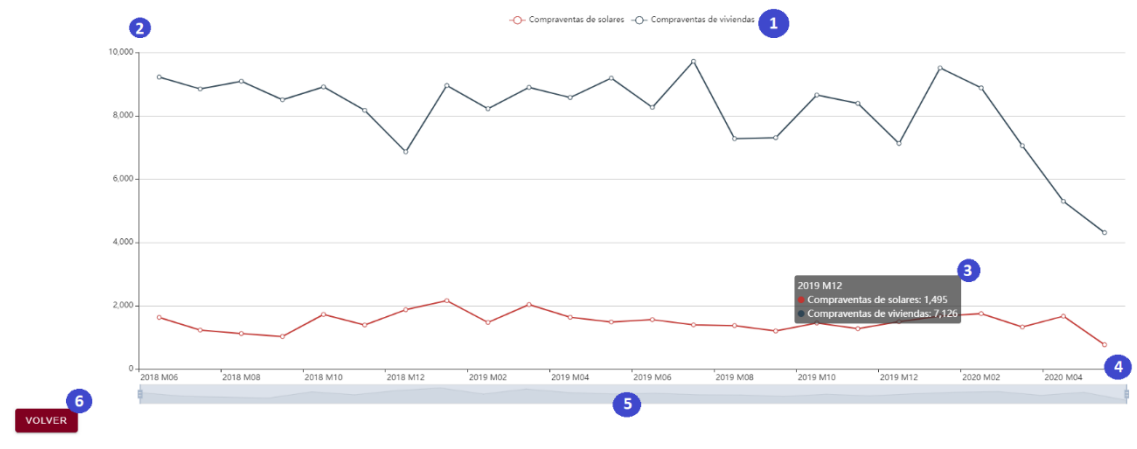


Ilustración 16: Pantalla diagrama de líneas

En esta pantalla se muestra el diagrama de líneas correspondiente a las series de compraventa de solares y de viviendas. A continuación, se describen los elementos marcados por los números en la imagen:

- 1) Leyenda: títulos de las distintas series representadas.
- 2) Eje vertical: Eje de la gráfica donde se observa el índice de valores de las series representadas.
- 3) Tooltip: Cuadro emergente en el que se muestran los valores de las gráficas en el momento correspondiente a la localización del ratón.
- 4) Eje horizontal: Eje de la gráfica donde se observan los distintos momentos en el tiempo.
- 5) Zoom: Herramienta de ECharts que realiza la función de zoom, permitiendo al usuario visualizar solo el fragmento de tiempo que le interese.
- 6) Volver: Botón que permite volver a la pantalla anterior, manteniendo las series seleccionadas.

Visualización: tabla

Datos

DESCARGAR EXCEL

Fecha	Compraventas de solares	Compraventas de viviendas
2018 M06	1630	9223
2018 M07	1230	8847
2018 M08	1120	9090
2018 M09	1028	8505
2018 M10	1724	8912
2018 M11	1391	8172
2018 M12	1873	6862
2019 M01	2160	8957
2019 M02	1472	8222
2019 M03	2037	8897
2019 M04	1634	8576
2019 M05	1486	9190
2019 M06	1555	8888

Ilustración 17: Pantalla tabla Excel

En esta pantalla se muestra la tabla de datos correspondiente a las series de compraventa de solares y de viviendas. A continuación, se describen los elementos marcados por los números en la imagen:

- 1) Volver: Botón que permite volver a la pantalla anterior, manteniendo las series seleccionadas.
- 2) Descarga: Botón que genera un archivo en formato *csv* con los mismos datos mostrados en la tabla.
- 3) Tabla: Tabla de datos en la que se muestran los datos de las distintas series ordenados por fecha.
- 4) Datos: Datos de las series, ordenados por fecha.
- 5) Fecha: Columna que sirve de índice de ordenación para los datos, está compuesta del año y el periodo.

7. Conclusión

Respecto a los objetivos establecidos al comienzo de este documento, podría decirse que se han alcanzado la totalidad de ellos en mayor o menor medida. Se ha desarrollado una aplicación web desde cero, eligiendo las tecnologías y las metodologías a usar, lo que ha supuesto que el alumno repase todo el conocimiento adquirido en estos campos durante el grado. Además, se han realizado los requisitos para satisfacer al cliente, lo cual, junto con una continuación del producto, puede llegar a suponer una oferta laboral, siendo algo muy importante dado el contexto de ser la última asignatura de la carrera.

A continuación, se encuentra una valoración personal de distintas cuestiones tratadas a lo largo de este proyecto:

¿Es recomendable utilizar un framework nuevo?

La respuesta corta sería no, pero hay distintos aspectos que pueden hacer cambiar esa respuesta. Utilizar un framework nuevo, como se explica varias veces a lo largo del proyecto, implica tratar con tecnologías desconocidas y falta de documentación y soporte. No obstante, también es cierto que un framework nuevo es innovador y seguramente aporte muchas ventajas al sistema que se desarrolle. Por lo tanto, la respuesta larga sería que dependiendo de las ventajas que ofrezca el framework nuevo para el producto que se va a desarrollar sí se podría llegar a utilizar. Siempre que se tenga en cuenta que hay que realizar una formación previa al comienzo del desarrollo.

¿Son útiles los Scrapers dada la posible dificultad de implementación?

Es cierto que dependiendo de lo que se quiera extraer la implementación de un scraper puede llegar a ser muy tediosa y compleja, sin embargo, estos programas se realizan al no haber alternativas para obtener los datos deseados. Por lo que sí, los scrapers son útiles siempre que se implementen correctamente.

¿Qué ventajas ofrece una arquitectura desacoplada?

Desde mi punto de vista personal, la arquitectura es el nuevo estándar de desarrollo de aplicaciones web y es recomendable aplicar esta arquitectura o una arquitectura de

microservicios en la gran mayoría de productos. Esto es debido a que este tipo de arquitectura aumenta el valor del producto sin aumentar el coste, además de permitir versatilidad a la hora de escoger la tecnología para el frontend y el backend.

Siendo más específico, al usar esta arquitectura, normalmente se crea una API para la comunicación con el frontend, lo que supone un valor añadido al producto ya que dicha API podrá ser utilizada por terceros o por aplicaciones que se desarrollen en un futuro.

Anexo I – Control de cambios.

Descripción	Localización	Fecha
Añadir nueva tarea de control de excepciones en el backend.	Planificación.	27-07-2020
Adelantar subtaska de filtrado de series a la fase de prototipo como tarea completa.	Planificación.	28-07-2020
Eliminar tarea de tasa de variación (TP.08).	Planificación.	28-07-2020
Incluir nueva tarea: Scraper Mineco. Movida subtaska “Datos de las fuentes coherentes y mostrados correctamente” a Scraper Mineco.	Planificación.	28/07/202
Incluir nuevo campo en las Series: Origen.	Backend – Diseño.	19/08/2020
Corregir costes.	Planificación – costes.	19/08/2020
Separar de análisis en backend y frontend corregida, todo el análisis se describe en el mismo apartado.	Realización del proyecto.	05/09/2020
Añadir manual de usuario.	Manual de usuario.	05/09/2020
Mover el glosario de términos al inicio del documento.	Glosario de términos.	05/09/2020