[PORTADA]

Índice

[1. Resumen 4](#_Toc46054148)

[2. Introducción 4](#_Toc46054149)

[3. Gestión del proyecto 4](#_Toc46054150)

[3.1. Alcance 4](#_Toc46054151)

[3.2. Objetivos 4](#_Toc46054152)

[3.3. Planificación 4](#_Toc46054153)

[3.4. Metodología 8](#_Toc46054154)

[3.5. Herramientas 8](#_Toc46054155)

[3.6. Relación con las asignaturas del Grado 8](#_Toc46054156)

[4. Realización del proyecto 8](#_Toc46054157)

[4.1. Backend 8](#_Toc46054158)

[4.1.1. Análisis 8](#_Toc46054159)

[4.1.2. Diseño 8](#_Toc46054160)

[4.1.3. Implementación 8](#_Toc46054161)

[4.1.4. Pruebas 8](#_Toc46054162)

[4.2. Frontend 8](#_Toc46054163)

[4.2.1. Análisis 8](#_Toc46054164)

[4.2.2. Diseño 9](#_Toc46054165)

[4.2.3. Implementación 9](#_Toc46054166)

[4.2.4. Pruebas 9](#_Toc46054167)

[5. Conclusión 9](#_Toc46054168)

# Resumen

Resumen de todo el documento (1pag aprox)

# Introducción

Problema -> solución

Objetivos

# Gestión del proyecto

## Alcance

## Objetivos

## 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 útima 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. En concreto, estas son las alternativas:

Base de datos: Oracle (sql) / MongoDB (nosql).

Frontend: Svelte / React.

Backend: Spring Boot / Otra.

Segunda fase: Prototipo

El objetivo de esta fase es implementar las funcionalidades principales. Para ello, se creará un scrapper 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 y sus prioridades. 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 |
| **TI.02** | Obtener conocimientos básicos sobre las herramientas elegidas desconocidas. | | | A |
| * Svelte | | | | 3h |
| * MongoDB | | | | 1h |
| **Segunda fase: Prototipo** | | | | |
| **Fecha Inicio: 08/07/2020** | | | **Fecha fin: 28/07/2020** | |
| **TP.01** | Inicialización del proyecto backend | | | A |
| * Crear proyecto base con Spring initializr. | | | | 2h |
| * Instalar dependencias (POM). | | | | 1h |
| **TP.02** | Tabla de series. | | | A |
| * Componente para mostrar las series en una tabla en Svelte. | | | | 5h |
| * Paginación en la tabla. | | | | 5h |
| * Endpoint con paginación en la API. | | | | 1h |
| **TP.03** | Almacenamiento de datos del sistema sobre series y sus datos en MongoDB. | | | A |
| * Dominio java para Serie con la siguiente información: código, periodo, fuente y colección de datos. | | | | 2h |
| * Dominio java para Datos con la siguiente información: momento y valor. | | | | 2h |
| * Repositorios y servicios para Series y Datos | | | | 2h |
| * BDD MongoDB y conectar con los repositorios anteriores | | | | 2h |
| **TP.04** | Scrapper INDEA | | | A |
| * Scrapper que recolecte series de INDEA e inserte las series en el sistema. | | | | 8h |
| **TP.05** | API backend | | | A |
| * Endpoints básicos para enviar las series de datos al frontend. | | | | 3h |
| * CORS. | | | | 2h |
| **TP.06** | Visualización de las series en forma de tabla. | | | A |
| * Componente tabla de datos Svelte. | | | | 5h |
| **TP.07** | Implementación de e-chart para la visualización de series. | | | A |
| * Componente e-chart en Svelte. | | | | 5h |
| * Visualización de varias series en el mismo gráfico. | | | | 5h |
| **TP.08** | Calcular la tasa de variación y visualizarla. | | | M |
| * Calcular tasa de variación en el backend. | | | | 2h |
| * Serie DTO para enviar la tasa con las series al frontend | | | | 3h |
| **TP.09** | Botón de descarga del gráfico. | | | M |
| * Botón para descargar el gráfico que se esté visualizando en el momento. | | | | 2h |
| **TP.10** | Mejorar apartado visual. | | |  |
| * Añadir estilo al frontend | | | | 6h |
| Tercera fase: Finalización | | | | |
| Fecha Inicio: 29/07/2020 | | Fecha fin: 29/08/2020 | | |
| TF.01 | Scrapper INE, EUROSTAT. | | | A |
| * Scrapper para fuente: INE. | | | | 6h |
| * Scrapper para fuente: EUROSTAT. | | | | 6h |
| * Datos de las fuentes coherentes y mostrados correctamente. | | | | 3h |
| TF.02 | Mejorar frontend – tareas extra frontend | | | B |
| * Menú principal. | | | | 5h |
| * Búsqueda y filtrado de series. | | | | 10h |
| * Mejorar estilo, apartado visual del sistema. | | | | 5h |
| * Control de errores. | | | | 3h |
| TF.03 | Mejorar backend – tareas extra backend | | | B |
| * Control de excepciones. | | | | 8h |
| * Documentación Swagger. | | | | 3h |
| * Búsqueda y filtrado de series. | | | | 3h |

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

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](https://www.glassdoor.es/Sueldos/sevilla-programador-junior-sueldo-SRCH_IL.0,7_IM1014_KO8,26.htm) para determinar el sueldo anual de un programador Junior en Sevilla.

La fórmula usada para calcular el coste es:

->

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:

->

Por lo que el coste total del proyecto seria **276.44 €.**

## Metodología

Control de calidad: sonarlint, checkstyle

## Herramientas

Github, toggl,

## Relación con las asignaturas del Grado

AISS, DP, IR, ISPP, Practicas?, …

# Realización del proyecto

## Backend

### Análisis

Requisitos

Requisitos de información:

Se quiere guardar la siguiente información acerca de cada serie (como mínimo)

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

En cada serie, hay que guardar los datos correspondientes a cada punto en el tiempo, los cuales constan de:

* Momento (en el tiempo)
* Valor

### Diseño

Arquitectura

### Implementación

### Pruebas

Test unitarios

## Frontend

Análisis

Visualización de las series:

Los datos de las series se deben de poder visualizar en forma de tabla (¿) y de gráfica. Además, se deben poder combinar distintas series en la misma gráfica con el fin de poder compararlas.

### Diseño

### Implementación

### Pruebas

# Conclusión

# Anexo I – Glosario de términos.

Frontend

Backend

Serie

API

Scrapper

Endpoint

Framework

# Anexo II – Control de cambios.

27/07/2020 – Añadida tarea control de errores frontend

Se deben de poder

**Tareas**

**Desarrollo**

Importar las series de IPC y EPA de Andalucía.

Completar el back para recoger los datos que falten de las series.

Visualizar tabla de datos en formato Excel y poder descargarlo.

Poder añadir y quitar series del gráfico de forma dinámica.

Calcular la tasa de variación y añadirla al gráfico.

Botón para descargar el gráfico que se está mostrando. (canvas)

Añadir las series de IPC y EPA de España.

Completar con todas las series de Europa, Andalucía y España.

Paginación

Credenciales back

**Fallos encontrados**

Problemas al arrancar svelte

Problema mostrando las series

Credenciales scrapper

**Lecciones aprendidas**

Reactividad svelte